## MICROEMACS

**MicroEmacs**, JASSPA Distribution, is defined as follows, refer to **me**(1) for a description of the command line variables.

The following sections describe the topics that are available as part of the on-line **MicroEmacs** manual pages. These pages document release number *20260902*.

Acknowledgments, Copyright, History and Contact Information.
Frequently Asked Questions.

See Help! for some information on using the hypertext manual pages.

Installation Information
Installation Patching Information
Setting Up A User Profile
Setting Up a Company Profile

Top Main Menu
Essential Commands
Help Information
Key Bindings

File Handling Commands
Dialogs and Menus
Cursor Movement Commands
Insertion and Deletion Commands
Paragraph and Text Formatting Commands
Capitalization and Transposition Commands
Searching and Replacing
Macro Commands
Buffer Manipulation Commands
Window Commands

Keyboard Binding Commands
Operating Modes
Shell and Command Controls
Spelling Commands
Hilighting, Color and Screen Appearance
Comparison and Differencing Commands
Short Cuts and Abbreviations
Message Line Commands
Printing Commands
Writing MicroEmacs Macros
Macro Development Commands
Registry Commands
Command Line Filters
Games

### Glossaries

Command Glossary
Variable Glossary
Macro Language Glossary

Global Glossary

### Miscellaneous Information

The following topics provide more in depth information:

Language Templates
File Hooks
Editor Files
Compatibility with the original MicroEMACS
Interfacing to external components
Supported File Types
Client-Server Interface
Regular Expressions
Building the source

Ref: MicroEmacs 2026 File: me.0 Date: 2026/09/23

### Help!

The on-line manual pages are defined as follows:

[logo] **Title**
[link1][link2]...[linkn]

The components of the header are defined as follows:

*[logo]* The MicroEmacs Logo in the top left hand corner of the screen exists on every page and hides a hypertext link. Selecting this item will take you back to the main MicroEmacs page (Only present on HTML and Windows help pages).

**Title** The title identifies the title of the topic that you are viewing.

**[link]** The link line provides quick links to other related material in the on-line manual pages. Use these to take you to a chosen topic. The **[home]** link returns to the contents page of the currently viewed topic, this is equivalent to the **Contents** button when viewing Microsoft Windows help files.

### Section Numbering

The section numbering conventions used in these pages is defined as follows:

(1) - Executable command line.
(2) - Editor built in commands.
(2m) - Editor built in modes.
(3) - Editor commands implemented as macros.
(4) - Editor macro language syntax
(5) - Editor variables
(8) - Editor specific file formats

Ref: Help! File: me.0 Date: 2026/09/23

## ESSENTIAL COMMANDS

The very essential commands which are the most important commands to know include:

**abort-command**(2) (**C-g**) Abort command
**backward-char**(2) (**C-b**) Move the cursor left
**backward-delete-char**(2) (**backspace**) Delete the previous character at the cursor position
**backward-line**(2) (**C-p**) Move the cursor to the previous line
**file-browser**(3) (**f10**) Browse the file system
**file-browser-close**(3) Close the file-browser
**file-browser-swap-buffers**(3) Swap between file-browser windows
**forward-char**(2) (**C-f**) Move the cursor right
**forward-delete-char**(2) (**C-d**) Delete the next character at the cursor position
**forward-line**(2) (**C-n**) Move the cursor to the next line
**isearch-forward**(2) (**C-s**) Search forward incrementally (interactive)
**quick-exit**(2) (**esc z**) Exit the editor writing changes
**save-buffer**(2) (**C-x C-s**) Save contents of changed buffer to file
**save-buffers-exit-emacs**(2) (**esc z**) Exit the editor prompt user to write changes
**undo**(2) (**C-x u**) Undo the last edit

Ref: Essential Commands File: me.0 Date: 2026/09/23

## CURSOR MOVEMENT COMMANDS

The cursor movement commands control how the cursor is moved around the buffer.

### Commands

**backward-char**(2) (**C-b**) Move the cursor left
**backward-line**(2) (**C-p**) Move the cursor to the previous line
**backward-paragraph**(2) (**esc p**) Move the cursor to the previous paragraph
**backward-word**(2) (**esc b**) Move the cursor to the previous word
**beginning-of-buffer**(2) (**esc <**) Move to beginning of buffer/file
**beginning-of-line**(2) (**C-a**) Move to beginning of line
**display-matching-fence**(3) Display the matching bracket
**end-of-buffer**(2) (**esc >**) Move to end of buffer/file
**end-of-line**(2) (**C-e**) Move to end of line
**forward-char**(2) (**C-f**) Move the cursor right
**forward-line**(2) (**C-n**) Move the cursor to the next line
**forward-paragraph**(2) (**esc n**) Move the cursor to the next paragraph
**forward-word**(2) (**esc f**) Move the cursor to the next word
**goto-line**(2) (**esc g**) Move the cursor to specified line
**goto-matching-fence**(2) (**esc C-f**) Move the cursor to matching fence
**goto-position**(2) Restore a stored position
**goto-window**(2) Restore a saved window to the current window (historic)
**list-commands**(2) (**C-h c**) List available commands
**list-variables**(2) (**C-h v**) List defined variables
**page-file-backward-line**(3) (**C-p**) Move backwards one line in paged buffer
**page-file-beginning-of-buffer**(3) Move to the start of the paged file
**page-file-end-of-buffer**(3) Move to the end of the paged file
**page-file-forward-line**(3) (**C-n**) Move forwards one line in paged buffer
**page-file-goto-byte**(3) Page to the buffer byte offset
**page-file-goto-page**(3) Move to page
**recenter**(2) (**C-l**) Recenter the window (refresh the screen)
**set-position**(2) Store the current position
**set-window**(2) Save the current window for restore (historic)
**universal-argument**(2) (**C-u**) Set the command argument count

### Variables

**$fmatchdelay**(5) Fence matching delay time
**$pause-time**(5) Length of time to pause for user

## INSERTION AND DELETION COMMANDS

Commands that initiate insertion or deletion of text include:

### Deletion

**backward-delete-char**(2) (**backspace**) Delete the previous character at the cursor position
**backward-delete-tab**(2) (**S-tab**) Delete white space to previous tab-stop
**backward-kill-word**(2) (**esc backspace**) Delete the previous word at the cursor position
**clean**(3) Remove redundant white spaces from the current buffer
**copy-rectangle**(3) Copy a rectangular region of text
**delete-blank-lines**(2) (**C-x C-o**) Delete blank lines about cursor
**delete-indentation**(3) Join 2 lines deleting white spaces
**fold-kill**(3) (**C-x C-k**) Delete the fold and all its contents
**forward-delete-char**(2) (**C-d**) Delete the next character at the cursor position
**forward-kill-word**(2) (**esc d**) Delete the next word at the cursor position
**kill-line**(2) (**C-k**) Delete all characters to the end of the line
**kill-paragraph**(2) Delete a paragraph
**kill-rectangle**(2) (**esc C-w**) Delete a column of text
**kill-region**(2) (**C-w**) Delete all characters in the marked region
**yank-rectangle**(2) (**esc C-y**) Insert a column of text
**yank-rectangle-overwrite**(3) Insert a column of text with overwrite

### Insertion

**bin-copy-region**(3) Binary supporting copy region
**bin-yank**(3) Binary supporting kill buffer insert
**copy-rectangle**(3) Copy a rectangular region of text
**insert-newline**(2) (**C-o**) Insert new line at cursor position
**insert-tab**(2) (**C-i**) Insert tab(s) into current buffer
**kill-rectangle**(2) (**esc C-w**) Delete a column of text
**normal-tab**(3) Insert a normal tab
**open-line**(3) (**C-o**) Open new line at cursor position
**quote-char**(2) (**C-q**) Insert literal character
**reyank**(2) (**esc y**) Restore next yank buffer
**tab**(2) (**tab**) Handle the tab key
**yank**(2) (**C-y**) Paste (copy) kill buffer contents into buffer
**yank-rectangle**(2) (**esc C-y**) Insert a column of text
**yank-rectangle-overwrite**(3) Insert a column of text with overwrite

### Variables

**$buffer-tab-width**(5) Buffer tab character interval
**$tab-width**(5) Global displayable tab character width

## PARAGRAPH AND TEXT FORMATTING COMMANDS

Commands that operate on paragraphs, and the layout of paragraphs:

### Paragraph

Paragraphs are separated by blank lines. A single paragraph is defined as all of the text enclosed between two blank lines, with no intervening blank lines.

**backward-paragraph**(2) (**esc p**) Move the cursor to the previous paragraph
**fill-paragraph**(2) (**esc o**) Format a paragraph
**forward-paragraph**(2) (**esc n**) Move the cursor to the next paragraph
**ifill-paragraph**(3) (**esc q**) Format a paragraph
**kill-paragraph**(2) Delete a paragraph
**paragraph-to-line**(3) Convert a paragraph to a single line
**wrap-word**(2) Wrap word onto next line

### Regions and Marks

A region is the text located between the **point** (the current cursor position) and the **mark** defined by set-mark.

**bin-copy-region**(3) Binary supporting copy region
**bin-yank**(3) Binary supporting kill buffer insert
**copy-rectangle**(3) Copy a rectangular region of text
**copy-region**(2) (**esc w**) Copy a region of the buffer
**count-words**(2) (**esc C-c**) Count the number of words in a region
**exchange-point-and-mark**(2) (**C-x C-x**) Exchange the cursor and marked position
**goto-alpha-mark**(2) (**C-x a**) Move the cursor to a alpha marked location
**kill-rectangle**(2) (**esc C-w**) Delete a column of text
**kill-region**(2) (**C-w**) Delete all characters in the marked region
**set-alpha-mark**(2) (**C-x C-a**) Place an alphabetic marker in the buffer
**set-mark**(2) (**esc space**) Set starting point of region
**write-region**(3) Write contents of region to named (new) file
**yank-rectangle**(2) (**esc C-y**) Insert a column of text
**yank-rectangle-overwrite**(3) Insert a column of text with overwrite

### Variables

**$buffer-fill-col**(5) Buffer paragraph mode; right fill column
**$buffer-fill-mode**(5) Buffer paragraph mode; justification method
**$fill-bullet**(5) Paragraph filling bullet character set
**$fill-bullet-len**(5) Paragraph filling bullet search depth
**$fill-col**(5) Global paragraph mode; right fill column
**$fill-eos**(5) Paragraph filling; end of sentence fill characters
**$fill-eos-len**(5) Paragraph filling; end of sentence padding length
**$fill-ignore**(5) Ignore paragraph filling character(s)
**$fill-mode**(5) Global paragraph mode; justification method

Ref: Paragraph and Text Formatting Commands File: me.0 Date: 2026/09/23

## SHORT CUTS

Automatic commands, history and automatic formatting modes such as **indent**(2).

### Commands

**abbrev-list**(3) Create a list of abbreviations
**buffer-abbrev-file**(2) Set buffers' abbreviation file
**comment-end**(3) End the current comment
**comment-line**(3) Comment out the current line
**comment-restyle**(3) Reformat the current comment
**comment-start**(3) Start a new comment
**comment-to-end-of-line**(3) Extend comment to end of line
**encase**(3) Encase a region with start/end string
**expand-abbrev**(2) Expand an abbreviation
**expand-abbrev-handle**(3) (**esc esc**) Expand an abbreviation handler
**expand-iso-accents**(3) Expand an ISO accent
**expand-look-back**(3) Complete a word by looking back for a similar word
**expand-word**(3) Complete a word by invocation of the speller
**find-tag**(2) (**esc t**) Find tag, auto-load file and move to tag position
**generate-password**(3) Generate a password with given criteria
**generate-tags-file**(3) Generate a tags file
**global-abbrev-file**(2) Set global abbreviation file
**indent**(2) Manage the auto-indentation methods
**insert-password**(3) Create and insert a new password string
**iso-accents-mode**(3) ISO accent expansion short-cut mode
**number-items**(3) Renumber a vertical list of items
**password-to-phonic**(3) Convert a password to phonic string
**read-session**(3) Read in session information
**save-session**(3) Write session information to session file
**session-setup**(3) Configure the data written to a session file
**uncomment-line**(3) Uncomment current line

### Variables

**%tag-file**(5) Tag file name
**%tag-option**(5) Tag file search option
**%tag-template**(5) Tag file search string

## REGISTRY

The registry commands provide an interface to manage the registry files defined by **erf**(8). The registry is a mechanism which allows the binding of information to a hierarchical tree node, using a file system metaphor to access the data. **MicroEmacs** uses a reserved root node `history` to save session information (see **save-session**(3)).

### Commands

**copy-registry**(3) Recursive copy of registry subtree
**delete-registry**(2) Delete a registry tree
**find-registry**(2) Index search of a registry sub-tree
**list-registry**(2) Display the registry in a buffer
**mark-registry**(2) Modify the operating mode of a registry node
**read-registry**(2) Read in a registry definition file
**save-registry**(2) Write a registry definition file
**set-registry**(2) Modify a node value in the registry

### Macro Functions

**&reg**(4) Retrieve a registry value (with default)

Ref: Registry File: me.0 Date: 2026/09/23

## COMMAND LINE FILTERS

MicroEmacs may be invoked from the command line to perform a specific set of filtering tasks, under control of a dedicated start up macro, see **me**(1) and **start-up**(3). A number of standard macros are provided, most of which are invoked automatically from the editor itself when requested by the user. Having said that, it has not been unknown for a colleague of mine to use the editor as a replacement for a more intelligent **sed**(1) filter, with 12 hours to go and a huge ugly 3-D geometric database to convert, what better way than run it through a set of MicroEmacs macros to turn it into another database format that can be handled - probably not for the uninitiated, but that person did pull it off and went home for tea !!

### Macro Command Line Filters

**2dos**(3f) Convert text files to DOS format files.
**2mac**(3f) Convert text files to Macintosh format files
**2unix**(3f) Convert text files to UNIX format files
**2win**(3f) Convert text files to Windows format files.
**benchmrk**(3f) Benchmark MicroEmacs macro processor speed
**ctags**(3f) Generate a C tags file
**dos2unix**(3f) Convert DOS format files to UNIX format files
**ehftools**(3f) Generate a MicroEmacs help file
**emftags**(3f) Generate a MicroEmacs macro tags file
**filetype**(3f) Determine file type UNIX, DOS or Windows
**gdiff**(3f) Command line graphical file difference
**info**(3f) Display Info pages in MicroEmacs.
**javatags**(3f) Generate a C tags file from Java sources
**jstags**(3f) Generate a Javascript tags file
**luatags**(3f) Generate a Lua tags file
**mdview**(3f) Browse Markdown files in MicroEmacs.
**ntags**(3f) Generate a nroff tags file
**perltags**(3f) Generate a Perl tags file
**phptags**(3f) Generate a PHP tags file
**printall**(3f) Formatted print job
**pydoc**(3f) Display Python nodule documentation in MicroEmacs.
**rdoc**(3f) Display R help pages in MicroEmacs.
**tcltags**(3f) Generate a Tcl/Tk tags file
**textags**(3f) Generate a LaTeX/BibTeX tags file
**unix2dos**(3f) Convert UNIX format files to DOS format files.
**unix2win**(3f) Convert UNIX format files to Windows format files.
**vbtags**(3f) Generate a Visual Basic tags file

### Macro Functions

**shut-down**(3) Editor exit callback command
**start-up**(3) Editor startup callback command

## Macro Variables

Ref: Command Line Filters File: me.0 Date: 2026/09/23

## SEARCHING AND REPLACING

Text searching and replacing commands:

**file-list**(3) Generate list of matching file names
**filemask-to-regex**(3) Convert a file mask to a regex
**find**(3) Execute built-in find command
**find-setup**(3) Setup the built-in find and grep commands
**grep**(3) Search a file for a pattern
**grep-for-region**(3) Grep current directory for current region
**hunt-backward**(2) (**C-x C-h**) Resume previous search in backward direction
**hunt-forward**(2) (**C-x h**) Resume previous search in forward direction
**isearch-backward**(2) (**C-r**) Search backwards incrementally (interactive)
**isearch-forward**(2) (**C-s**) Search forward incrementally (interactive)
**item-list**(3) (**F7**) Abbreviated search and list buffer contents
**item-list-close**(3) (**esc F7**) Close the item list
**item-list-find**(3) Find the selected item in the item list
**narrow-search-backward**(3) (**C-c r**) Search for a string in the backward direction
**narrow-search-forward**(3) (**C-c s**) Search for a string in the forward direction
**occur**(3) Regular expression search for occurrences
**osd-search**(3) Search and annotate the current buffer
**page-file-isearch-backward**(3) (**C-r**) Page search backwards incrementally (interactive)
**page-file-isearch-forward**(3) (**C-s**) Page search forward incrementally (interactive)
**page-file-search-backward**(3) (**C-x r**) Page search for a string in the backward direction
**page-file-search-buffer**(3) Search the page file buffer
**page-file-search-forward**(3) (**C-x s**) Page search for a string in the forward direction
**query-replace-all-string**(3) Query replace string in a list of files
**query-replace-string**(2) (**esc C-r**) Search and replace a string - with query
**query-replace-string-in-files**(3) Conditionally search and replace strings across files
**RegularExpressions**(2) Regular Expressions
**replace-all-pairs**(3) Replace string pairs in a list of files
**replace-all-string**(3) Replace string with new string in a list of files
**replace-string**(2) (**esc r**) Replace string with new string
**replace-string-in-files**(3) Search and replace strings across files
**search-backward**(2) (**C-x r**) Search for a string in the backward direction
**search-buffer**(2) Search the current buffer for a string
**search-forward**(2) (**C-x s**) Search for a string in the forward direction
**str-to-regex**(3) Generate regex safe variant of a string

### Comparison and Differencing

Commands that perform comparisons and differences:

### Commands

**compare-windows**(2) Compare buffer windows, ignore whitespace
**compare-windows-exact**(3) Compare buffer windows, with whitespace
**diff**(3) Difference files or directories
**diff-changes**(3) Find the differences from a previous edit session
**diff-lines**(3) Highlight the differences between two lines
**gdiff**(3) Graphical file difference
**rdiff**(3) Recursively compare files or directories
**xdiff**(3) External graphical file difference

### Variables

**%diff-com**(5) Diff command line
**%gdiff-com**(5) Gdiff command line
**%xdiff-com**(5) xdiff command line

## HILIGHTING, COLOR AND SCREEN APPEARANCE

Commands that change the hilighting, screen color and screen appearance:

### Commands

**add-color**(2) Create a new color
**add-color-scheme**(2) Create a new color scheme
**change-font**(2) Change the screen font
**change-font-size**(3) Incrementally change font size
**change-frame-depth**(2) Change the number of lines on the current frame
**change-frame-width**(2) Change the number of columns on the current frame
**change-scheme**(3) Change the current highlight scheme
**delete-hilight-lines**(3) Delete hilight lines
**hilight**(2) Manage the buffer hilighting schemes
**indent**(2) Manage the auto-indentation methods
**indent-decrease**(3) Decrease the line indentation
**indent-increase**(3) Increase the line indentation
**line-scheme**(3) Apply line annotation in current buffer
**osd-search**(3) Search and annotate the current buffer
**print-color**(2) Create a new printer color
**print-scheme**(2) Create a new printer color and font scheme
**restyle-buffer**(3) Automatically reformat a buffer's indentation
**restyle-region**(3) Automatically reformat a regions indentation
**scheme-editor**(3) Color Scheme Editor
**show-region**(2) Show the current copy region

### Variables

**$box-chars**(5) Characters used to draw lines
**$buffer-hilight**(5) Define current buffer hilighting scheme
**$buffer-indent-width**(5) Buffer indentation width in characters
**$buffer-scheme**(5) Buffer color scheme
**$cursor-blink**(5) Cursor blink rate
**$cursor-color**(5) Cursor foreground color
**$frame-depth**(5) Number of lines on the current frame canvas
**$frame-width**(5) Number of columns on the current frame canvas
**$global-scheme**(5) Global buffer color scheme
**$indent-width**(5) Global indentation width in characters
**$line-scheme**(5) Set the current line color scheme
**$mode-line**(5) Mode line format
**$mode-line-scheme**(5) Mode line color scheme
**$mouse-pos**(5) Mouse position information
**$screen-depth**(5) Number of character lines on the screen canvas
**$screen-width**(5) Number of character columns on the screen canvas
**$scroll-bar**(5) Scroll bar configuration
**$scroll-bar-scheme**(5) Scroll bar color scheme
**$show-modes**(5) Select buffer modes to display
**$show-region**(5) Enable the hilighting of regions

**$system**(5) System configuration variable
**$trunc-scheme**(5) Truncation color scheme
**$window-chars**(5) Character set used to render the windows

## KEYBOARD BINDING COMMANDS

Keyboard binding allows key strokes to be associated with commands and macros such that when a bound key stroke sequence is recognized its associated (or bound) command is invoked, thereby controlling the editor. A set of Default Bindings exist for MicroEmacs which may be altered using the binding commands. There are three types of key bindings:

**Global** Associates a key-stroke with a command for all buffers. Used to establish the standard keyboard controls i.e. cursor movement, search, replace etc.

**Local** Associates a key-stroke with a command for a specified buffer only, i.e. a binding local to the buffer. Local bindings allow macro accelerators to be bound to keys without affecting other buffers containing different types of data. Local bindings are used extensively in the buffer hook commands.

**Message Line**
Associates a key binding for use on the command line only, allowing command completion to be diverted etc.

To bind a command to a key, the command and key names must be known, see Command Glossary for a complete list of commands and Key Names for a complete list of key names.

The binding related commands are defined as follows:

### Commands

**buffer-bind-key**(2) Create local key binding for current buffer
**buffer-unbind-key**(2) Remove local key binding for current buffer
**command-apropos**(2) (**C-h a**) List commands involving a concept
**describe-bindings**(2) (**C-h b**) Show current command/key binding
**describe-key**(2) (**C-x ?**) Report keyboard key name and binding
**expand-iso-accents**(3) Expand an ISO accent
**global-bind-key**(2) (**esc k**) Bind a key to a named command or macro
**global-unbind-key**(2) (**esc C-k**) Unbind a key from a named command or macro
**iso-accents-mode**(3) ISO accent expansion short-cut mode
**ml-bind-key**(2) Create key binding for message line
**ml-unbind-key**(2) Remove key binding from message line
**osd-bind-key**(2) Create key binding for OSD dialog
**osd-unbind-key**(2) Remove key binding from OSD dialog
**set-char-mask**(2) Set character word mask
**translate-key**(2) Translate key

### Variables

#### Alt Key

The **Alt Key** has special binding priorities defined as follows:

- Direct key binding (e.g. **A-b** executes file-browser)
- Main menu hot key (e.g. **A-f** opens the File menu)

- Meta key binding (e.g. **A-space** -> **esc space** -> set-mark)

If the ALT key is to be used strictly as the Emacs Meta key then the bindings for the menu should be over-ridden by *Direct Key Bindings* from the user configuration file i.e. to re-map the default MicroEmacs Alt key to equivalent esc keys then the following keys should be re-bound.

```
global-bind-key forward-word "A-f"      ; Over-ride File menu binding
:                                       ; For all of the other menu items.
:
global-bind-key backward-word "A-b"     ; Over-ride the file browser.
global-bind-key replace-string "A-r"    ; Over-ride tools binding.
```

This creates a higher priority binding which overrides the underlying default. The commands that are displaced would have to be re-bound to different keys if required.

## BUFFER MANIPULATION COMMANDS

A buffer is where MicroEmacs stores text. Normally text is read from a file and is visible in an editing window. The name, associated file and operating modes of the buffer, are generally shown in the mode line.

Commands that deal with buffers include:

### Commands

**buffer-info**(2) (**C-x =**) Status information on current buffer position
**buffer-major-mode**(3) Change the operating mode of a buffer
**change-buffer-name**(2) (**esc C-n**) Change name of current buffer
**change-file-name**(2) (**C-x n**) Change the file name of the current buffer
**delete-buffer**(2) (**C-x k**) Delete a buffer
**delete-some-buffers**(2) Delete buffers with query
**execute-buffer**(2) Execute script lines from a buffer
**execute-line**(2) Execute a typed in script line
**find-buffer**(2) (**C-x b**) Switch to a named buffer
**insert-file-name**(2) (**C-x C-y**) Insert filename into current buffer
**list-buffers**(2) (**C-x C-b**) List all buffers and show their status
**narrow-buffer**(2) Hide buffer lines
**next-buffer**(2) (**C-x x**) Switch to the next buffer
**save-all**(3) Save all modified files (with query)
**save-some-buffers**(2) Save contents of all changed buffers to file (with query)

### Variables

**$MEBACKUPPATH**(5) Backup file location
**$MEBACKUPSUB**(5) Backup file name modifier
**$buffer-backup**(5) Buffer backup file name
**$buffer-bname**(5) Name of the current buffer
**$buffer-fmod**(5) Buffer file modes (or attributes)
**$buffer-fname**(5) Name of the current buffer's file name
**$buffer-indent-width**(5) Buffer indentation width in characters
**$buffer-mask**(5) Current buffer word class mask
**$buffer-mode-line**(5) Buffer mode line string
**$buffer-names**(5) Filtered buffer name list
**$buffer-xlate**(5) Buffer on-save translation configuration
**$file-names**(5) Filtered file name list
**$file-size-prompt**(5) Large file size warning prompt
**$global-fmod**(5) Global file modes (or attributes)
**$indent-width**(5) Global indentation width in characters
**$major-mode-ids**(5) Filtered major-mode ID list
**$mode-chars**(5) Mode characters
**$mode-line**(5) Mode line format
**$mode-line-scheme**(5) Mode line color scheme
**$show-modes**(5) Select buffer modes to display

Ref: Buffer Manipulation Commands File: me.0 Date: 2026/09/23

## CAPITALIZATION AND TRANSPOSITION COMMANDS

Commands to change the capitalization and transposition of text:

**capitalize-word**(2) (**esc c**) Capitalize word
**generate-hash**(2) Generate SHA1 or SHA256 hash or HMAC
**lower-case-region**(2) (**C-x C-l**) Lowercase a region (downcase)
**lower-case-word**(2) (**esc l**) Lowercase word (downcase)
**sort-lines**(2) Alphabetically sort lines
**sort-lines-ignore-case**(3) Alphabetically sort lines ignoring case
**transpose-chars**(2) (**C-t**) Exchange (swap) adjacent characters
**transpose-lines**(2) (**C-x C-t**) Exchange (swap) adjacent lines
**transpose-region-line**(3) Exchange (swap) region with adjacent line
**uniq**(3) Make lines in a sorted list unique
**upper-case-region**(2) (**C-x C-u**) Uppercase a region (upcase)
**upper-case-word**(2) (**esc u**) Uppercase word (upcase)

## FILE HANDLING COMMANDS

Commands to read, write and interact with files:

### Commands

**append-buffer**(2) Write contents of buffer to end of named file
**browse**(3) Browse a file or buffer in view mode
**change-company**(3) Change current company profile
**change-file-name**(2) (**C-x n**) Change the file name of the current buffer
**copy-file**(3) Copy or move files and directories
**delete-file**(3) Delete files and directories
**delete-file-delayed**(3) Delete file after a specified delay
**directory-tree**(2) Draw the file directory tree
**file-attrib**(3) Set the current buffers system file attributes
**file-browser**(3) (**f10**) Browse the file system
**file-browser-close**(3) Close the file-browser
**file-browser-swap-buffers**(3) Swap between file-browser windows
**file-op**(2) File system operations command
**file-tool-setup**(3) Define file browser tools
**find-bfile**(3) (**C-x 9**) Load a file as binary data
**find-cfile**(3) Load an encrypted file
**find-file**(2) (**C-x C-f**) Load a file
**find-hfile**(3) (**C-x C-h**) Load a file using selected handler
**find-lfile**(3) Load a file lossless
**find-rfile**(3) Load a file as binary data (reduced)
**find-ufile**(3) Load a Unicode file
**find-zfile**(3) Compressed file support
**ftp**(3) Initiate an FTP connection
**insert-file**(2) (**C-x C-i**) Insert file into current buffer
**insert-file-name**(2) (**C-x C-y**) Insert filename into current buffer
**insert-template**(3) Insert template file into current buffer
**page-bfile**(3) Load a large binary file as binary data
**page-file**(3) Load a large file in page mode
**page-file-save-buffer**(3) (**C-x C-s**) Save contents of a page mode buffer to file
**read-file**(2) (**C-x C-r**) Find and load file replacing current buffer
**reread-all**(3) Reload all files
**reread-file**(3) Reload the current buffer's file
**save-all**(3) Save all modified files (with query)
**save-buffer**(2) (**C-x C-s**) Save contents of changed buffer to file
**save-some-buffers**(2) Save contents of all changed buffers to file (with query)
**set-encryption-key**(2) (**esc e**) Define the encryption key
**suspend-emacs**(2) Suspend editor and place in background
**translate-buffer**(2) Translate buffer from one format to another
**view-file**(2) (**C-x C-v**) Load a file read only
**watch**(3) Monitor a changing file
**watch-reset**(3) Reset the watch command counters
**watch-time**(3) Change the watch command poll time
**write-buffer**(2) (**C-x C-w**) Write contents of buffer to named (new) file

**write-region**(3) Write contents of region to named (new) file
**zfile-reset**(3) Compressed file support reset
**zfile-setup**(3) Compressed file support setup

### Variables

**$MEINSTALLPATH**(5) MicroEmacs system macros search path
**$MEPATH**(5) MicroEmacs search path
**$MEUSERPATH**(5) MicroEmacs user directory
**$auto-time**(5) Automatic buffer save time
**$file-ignore**(5) File extensions to ignore
**$file-size-prompt**(5) Large file size warning prompt
**$home**(5) Users 'home' directory location
**$kept-versions**(5) Number of backups to be kept
**$progdata**(5) Users program data directory
**$search-path**(5) MicroEmacs search path
**$timestamp**(5) Time stamp string
**$user-name**(5) MicroEmacs user name
**$user-path**(5) MicroEmacs user directory
**%company-name**(5) Name of company for template
**%company-ref**(5) Company reference template
**%company-url**(5) Company URL for template
**%ftp-flags**(5) Configure the FTP console
**%http-cookies**(5) Set HTTP request cookies
**%http-flags**(5) Configure the HTTP console
**%http-header**(5) Set additional HTTP request headers
**%http-post-data**(5) Set HTTP request form data
**%http-post-file**(5) Set HTTP request file to submit
**%http-proxy-addr**(5) Set HTTP proxy server address
**%http-proxy-port**(5) Set HTTP proxy server port
**%os-chars**(5) Characters used by the OS
**%sock-recv-timeout**(5) Set socket receive timeout
**%sock-send-timeout**(5) Set socket send timeout
Dialogs and Menus

## DIALOGS AND MENUS

Menus and dialogs in the system:

### Commands

**alias-path-setup**(3) Define alternative path alias
**buffer-setup**(3) Configure the current buffer settings
**describe-word**(3) Display a dictionary definition of a word
**file-tool-setup**(3) Define file browser tools
**file-type-setup**(3) Define file type set
**find-word**(3) Find a word using spelling dictionaries
**generate-tags-file**(3) Generate a tags file
**insert-symbol**(3) Insert an ASCII character
**line-scheme**(3) Apply line annotation in current buffer
**MainMenu**(3) The top main menu
**major-mode-help**(3) Displays help page for current buffer
**major-mode-setup**(3) Configures the current buffer settings
**notes**(3) Tagged hierarchical database of notes
**notes-close**(3) Close the notes window
**notes-context**(3) Create/Access buffer context note
**notes-file**(3) Open a notes database file
**notes-import-skwyrul**(3) Import a Skwyrul index file (skwyrul_l.htm)
**notes-instant**(3) Create quick note
**organizer**(3) Calendar and address organizer
**osd**(2) Manage the On-Screen Display
**osd-dialog**(3) OSD dialog box
**osd-entry**(3) OSD entry dialog box
**osd-search**(3) Search and annotate the current buffer
**osd-xdialog**(3) OSD Extended dialog box
**print-setup**(3) Configure MicroEmacs's printer interface
**scheme-editor**(3) Color Scheme Editor
**shell-tool-setup**(3) Define shell tools
**spell-buffer**(3) Spell check the current buffer
**spell-edit-word**(3) Edits a spell word entry
**spell-word**(3) (**esc $**) Spell check a single word
**toolbar**(3) Toggle tool bar visibility
**toolbar-close**(3) Close the tool bar
**toolbar-open**(3) Open the tool bar
**toolbar-refresh**(3) Update the tool bar
**user-setup**(3) Configure MicroEmacs for a specific user

### Variables

**$osd-scheme**(5) OSD color scheme

## EDITOR FILE TYPES

Different file types used by MicroEmacs:

**eaf**(8) MicroEmacs abbreviation file format
**edf**(8) MicroEmacs spelling dictionary file
**ehf**(8) MicroEmacs help file
**emf**(8) MicroEmacs macro file
**enf**(8) Notes database file
**erf**(8) MicroEmacs registry file
**esf**(8) MicroEmacs session file
**etf**(8) MicroEmacs template file format

Ref: Editor File Types File: me.0 Date: 2026/09/23

## SPELLING COMMANDS

Commands related to spelling:

### Commands

**add-dictionary**(2) Declare existence of a spelling dictionary
**add-spell-rule**(2) Add a new spelling rule to the dictionary
**auto-spell**(3) Auto-spell support
**auto-spell-buffer**(3) Auto-spell whole buffer
**auto-spell-ignore**(3) Auto-spell ignore current word
**auto-spell-menu**(3) Auto-spell context menu
**auto-spell-reset**(3) Auto-spell hilight reset
**delete-dictionary**(2) Remove a spelling dictionary from memory
**describe-word**(3) Display a dictionary definition of a word
**edit-dictionary**(3) Insert a dictionary in a buffer
**expand-word**(3) Complete a word by invocation of the speller
**find-word**(3) Find a word using spelling dictionaries
**restore-dictionary**(3) Save dictionary user changes
**save-dictionary**(2) Save changed spelling dictionaries
**spell**(2) Spell checker service provider
**spell-add-word**(3) Add a word to the main dictionary
**spell-buffer**(3) Spell check the current buffer
**spell-edit-word**(3) Edits a spell word entry
**spell-word**(3) (**esc $**) Spell check a single word

### Variables

**$find-words**(5) Filtered word list

## MACRO COMMANDS

Everyday macro commands used by the user. See Macro Development Commands for commands related to macro development.

### Commands

**end-kbd-macro**(2) (**C-x )**) Stop recording keyboard macro
**execute-buffer**(2) Execute script lines from a buffer
**execute-file**(2) (**esc /**) Execute script lines from a file
**execute-kbd-macro**(2) (**C-x e**) Execute a keyboard macro
**execute-line**(2) Execute a typed in script line
**execute-named-command**(2) [**esc x** ] Execute a named command
**insert-macro**(2) Insert keyboard macro into buffer
**kbd-macro-query**(2) (**C-x q**) Query termination of keyboard macro
**name-kbd-macro**(2) Assign a name to the last keyboard macro
**start-kbd-macro**(2) (**C-x (**) Start recording keyboard macro

### Variables

**$debug**(5) Macro debugging flag

## MACRO DEVELOPMENT COMMANDS

MicroEmacs is based on a macro language which may be extended, refer to Writing MicroEmacs Macros to get started with the language.

Commands used in macro development, and more specialized commands which are only invoked from macros. Refer to Macro Commands for keyboard macros etc.

An additional set of commands for use with macros is outlined in the Introduction to Variable Functions section. The Macro Language Glossary contains a full list of macro related commands and special variables.

### Commands

**add-file-hook**(2) DEPRICATED: Declare file name context dependent configuration
**ascii-time**(3) Return the current time as a string
**buffer-info**(2) (**C-x =**) Status information on current buffer position
**buffer-init**(3) Initialize a buffer language template
**buffer-init-fhook**(3) Initialize the buffer file hooks
**buffer-init-hooks**(3) Initialize the buffer language template hooks
**command-wait**(2) Conditional wait command
**copy-file**(3) Copy or move files and directories
**copy-registry**(3) Recursive copy of registry subtree
**create-callback**(2) Create a timer callback
**create-frame**(2) (**C-x 5 1**, **C-x 5 2**) Create a new frame
**define-macro-file**(2) Define macro file location
**delete-file**(3) Delete files and directories
**delete-file-delayed**(3) Delete file after a specified delay
**delete-frame**(2) (**C-x 5 0**) Delete the current frame
**directory-tree**(2) Draw the file directory tree
**execute-string**(2) Execute a string as a command
**file-list**(3) Generate list of matching file names
**file-op**(2) File system operations command
**fileHooks**(2) File Hooks
**filemask-to-regex**(3) Convert a file mask to a regex
**goto-position**(2) Restore a stored position
**goto-window**(2) Restore a saved window to the current window (historic)
**hilight**(2) Manage the buffer hilighting schemes
**insert-space**(2) Insert space(s) into current buffer
**insert-string**(2) Insert character string into current buffer
**languageTemplates**(2) File Language Templates
**localeSupport**(2) Locale Support
**major-mode**(2) Declare or set a buffer major mode
**newline**(2) (**return**) Insert a new line
**next-frame**(2) (**C-x 5 o**) Change the focus to the next frame
**osd-dialog**(3) OSD dialog box
**osd-entry**(3) OSD entry dialog box
**osd-xdialog**(3) OSD Extended dialog box
**page-bfile**(3) Load a large binary file as binary data
**page-file**(3) Load a large file in page mode
**page-file-backward-line**(3) (**C-p**) Move backwards one line in paged buffer

**page-file-beginning-of-buffer**(3) Move to the start of the paged file
**page-file-end-of-buffer**(3) Move to the end of the paged file
**page-file-forward-line**(3) (**C-n**) Move forwards one line in paged buffer
**page-file-goto-byte**(3) Page to the buffer byte offset
**page-file-goto-page**(3) Move to page
**page-file-isearch-backward**(3) (**C-r**) Page search backwards incrementally (interactive)
**page-file-isearch-forward**(3) (**C-s**) Page search forward incrementally (interactive)
**page-file-save-buffer**(3) (**C-x C-s**) Save contents of a page mode buffer to file
**page-file-scroll-down**(3) (**C-n**) Scroll paged buffer down
**page-file-scroll-up**(3) (**C-p**) Scroll paged buffer up
**page-file-search-backward**(3) (**C-x r**) Page search for a string in the backward direction
**page-file-search-buffer**(3) Search the page file buffer
**page-file-search-forward**(3) (**C-x s**) Page search for a string in the forward direction
**popup-window**(2) Pop-up a window on the screen
**screen-poke**(2) Immediate write string to the screen
**screen-update**(2) (**redraw**) Force screen update
**search-buffer**(2) Search the current buffer for a string
**set-cursor-to-mouse**(2) Move the cursor to the current mouse position
**set-position**(2) Store the current position
**set-scroll-with-mouse**(2) Scroll the window with the mouse
**set-variable**(2) (**C-x v**) Assign a new value to a variable
**set-window**(2) Save the current window for restore (historic)
**shut-down**(3) Editor exit callback command
**spell**(2) Spell checker service provider
**start-up**(3) Editor startup callback command
**str-to-regex**(3) Generate regex safe variant of a string
**toolbar-refresh**(3) Update the tool bar
**unset-variable**(2) Delete a variable
**void**(2) Null command

### Variables

**$MEBACKUPPATH**(5) Backup file location
**$MEBACKUPSUB**(5) Backup file name modifier
**$buffer-backup**(5) Buffer backup file name
**$buffer-bhook**(5) Buffer macro hook command name (buffer current)
**$buffer-dhook**(5) Buffer macro hook command name (buffer deletion)
**$buffer-ehook**(5) Buffer macro hook command name (buffer swapped)
**$buffer-fhook**(5) Buffer macro hook command name (buffer creation)
**$buffer-fmod**(5) Buffer file modes (or attributes)
**$buffer-indent**(5) Current buffer indentation scheme
**$buffer-input**(5) Divert buffer input through macro
**$buffer-ipipe**(5) Divert buffer incremental pipe input through macro
**$buffer-ipipe-flags**(5) Incremental pipe creation flags
**$buffer-major-mode**(5) Define buffer major-modem
**$buffer-names**(5) Filtered buffer name list
**$buffer-xlate**(5) Buffer on-save translation configuration
**$build**(5) MicroEmacs build information
**$command-names**(5) Filtered command name list

**$cursor-x**(5) Mouse X (horizontal) position
**$cursor-y**(5) Mouse Y (vertical) position
**$debug**(5) Macro debugging flag
**$file-names**(5) Filtered file name list
**$find-words**(5) Filtered word list
**$global-fmod**(5) Global file modes (or attributes)
**$line-flags**(5) Current buffer line flags
**$major-mode-ids**(5) Filtered major-mode ID list
**$mode-names**(5) Filtered mode name list
**$mouse**(5) Mouse configuration variable
**$mouse-x**(5) Mouse X (horizontal) position
**$mouse-y**(5) Mouse Y (vertical) position
**$platform**(5) MicroEmacs host platform identifier
**$progname**(5) Program file name
**$random**(5) Generate a random number
**$result**(5) Various command return values
**$status**(5) Macro command execution status
**$system**(5) System configuration variable
**$temp-name**(5) Temporary file name
**$variable-names**(5) Filtered variable name list
**$version**(5) MicroEmacs version date-code
**$window-flags**(5) Current window setup flags
**$window-mode-line**(5) Window mode line position
**$window-scroll-bar**(5) Window scroll bar (or separator) position
*%***ascii-time**(5) Set the default format of the date string
**.about.arg0**(5) First command line option
**.about.arg1**(5) Second command line option
**.about.arg2**(5) Third command line option
**.about.argc**(5) Command line argument count
**.calc.answer**(5) Last successful calc answer
**.calc.result**(5) Last calc calculation result

## HELP INFORMATION

Commands to retrieve on-line help information and status.

**about**(2) Information About MicroEmacs
**command-apropos**(2) (**C-h a**) List commands involving a concept
**describe-key**(2) (**C-x ?**) Report keyboard key name and binding
**describe-variable**(2) (**C-h v**) Describe current setting of a variable
**help**(2) (**esc ?**) Help; high level introduction to help
**help-command**(2) (**C-h C-c**) Help; command information
**help-item**(2) (**C-h C-i**) Help; item information
**help-variable**(2) (**C-h C-v**) Help; variable information
**info**(3) Display a GNU Info database
**info-goto-link**(3) Display Info on a given link
**info-on**(3) Display Info on a given topic
**list-buffers**(2) (**C-x C-b**) List all buffers and show their status
**osd-help**(3) GUI based on-line help

Ref: Help Information File: me.0 Date: 2026/09/23

## OPERATING MODES

**Modes** are switches (or states) that may be applied globally or on a per buffer basis whose settings determine how MicroEmacs operates. Modes affect operations within a buffer, global modes determine the modes of newly created buffers.

Commands to alter the operating state:

**add-global-mode**(3) Set a global buffer mode
**add-mode**(3) Set a local buffer mode
**buffer-major-mode**(3) Change the operating mode of a buffer
**buffer-mode**(2) (**C-x m**) Change a local buffer mode
**delete-global-mode**(3) Remove a global buffer mode
**delete-mode**(3) Remove a local buffer mode
**global-mode**(2) (**esc m**) Change a global buffer mode
**named-buffer-mode**(2) Change a named buffer mode
**unmark-buffer**(3) Remove buffer edited flag

### Modes

The operating modes are defined as follows:

**auto**(2m) Automatic source file line type detection
**autosv**(2m) Automatic file save
**backup**(2m) Automatic file backup of last edit
**binary**(2m) Binary editor mode
**cr**(2m) File's carriage return (CR) style
**crypt**(2m) Encrypted file mode
**ctrlz**(2m) File's termination style
**cursor**(2m) Buffer's cursor visibility
**edit**(2m) Buffer has be changed
**exact**(2m) Searching and sorting case sensitivity
**fence**(2m) Auto fence matching mode
**hide**(2m) Hide buffer
**indent**(2m) Automatic indentation
**justify**(2m) Justification Mode
**lf**(2m) File's line feed style
**lock**(2m) Pipe cursor position lock
**magic**(2m) Regular expression search
**nact**(2m) Buffer not active
**narrow**(2m) Buffer contains a narrow
**over**(2m) Over-strike Mode
**pipe**(2m) Incremental Pipe running
**rbin**(2m) Reduced binary editor mode
**tab**(2m) Tabulation mode
**time**(2m) File time stamping
**undo**(2m) Retain edit modifications
**view**(2m) Read only
**wrap**(2m) Line wrap entered text

**xlate**(2m) Translated buffer editor mode

### Mode Line

The buffer modes may be shown on the mode line as single letter mnemonics as follows:

**A**uto, **a**utosv, **B**ackup, **b**inary, **c**rlf, cr**Y**pt, ctrl**z**, **d**el, **D**ir, **e**dit, **E**xact, **f**ence, **H**ide, **I**ndent, **J**ustify, loc**k**, **M**agic, **n**act, **N**arrow, **O**ver, **P**ipe, **Q**uiet, **r**bin, **S**ave, **T**ab, **t**ime, **U**ndo, usr**1**, usr**2**, usr**3**, usr**4**, usr**5**, usr**6**, usr**7**, usr**8**, **V**iew, **W**rap.

## SHELL AND COMMAND CONTROLS

Operating system and external system call invocations:

### Commands

**add-next-line**(2) Define the searching behavior of command output
**ClearCase**(3) MicroEmacs ClearCase interface
**cc-add**(3) MicroEmacs ClearCase interface - add file
**cc-checkin**(3) MicroEmacs ClearCase interface - checkin files
**cc-checkout**(3) MicroEmacs ClearCase interface - checkout files
**cc-delete**(3) MicroEmacs ClearCase interface - delete files
**cc-diff**(3) MicroEmacs ClearCase interface - diff changes
**cc-setup**(3) MicroEmacs ClearCase interface setup
**cc-state**(3) MicroEmacs ClearCase interface - list state of directory files
**cc-update**(3) MicroEmacs ClearCase interface - update directory files
**compile**(3) Start a compilation process
**cvs**(3) MicroEmacs CVS interface
**cvs-add**(3) MicroEmacs CVS interface - add file
**cvs-annotate**(3) MicroEmacs CVS interface - annotate file
**cvs-cat-file**(3) MicroEmacs CVS interface - output specific revision of a file
**cvs-checkout**(3) MicroEmacs CVS interface - checkout files
**cvs-commit**(3) MicroEmacs CVS interface - commit changes
**cvs-diff**(3) MicroEmacs CVS interface - diff changes
**cvs-file-status**(3) MicroEmacs CVS interface - status of files
**cvs-gdiff**(3) MicroEmacs CVS interface - graphical diff changes
**cvs-log-file**(3) MicroEmacs CVS interface - log changes
**cvs-remove**(3) MicroEmacs CVS interface - remove file
**cvs-resolve**(3) MicroEmacs CVS interface - resolve conflicts
**cvs-resolve-conflicts**(3) MicroEmacs CVS interface - resolve conflicts
**cvs-setup**(3) MicroEmacs CVS interface setup
**cvs-status**(3) MicroEmacs CVS interface - list state of directory files
**cvs-tag**(3) MicroEmacs CVS interface - tag files
**cvs-update**(3) MicroEmacs CVS interface - update directory files
**cygwin**(3) Open a Cygwin BASH window
**dbx**(3) UNIX Debugger
**diff**(3) Difference files or directories
**diff-changes**(3) Find the differences from a previous edit session
**execute-tool**(3) Execute a user defined shell tool
**filter-buffer**(2) (**C-x #**) Filter the current buffer through an O/S command
**gdb**(3) GNU Debugger
**gdiff**(3) Graphical file difference
**generate-tags-file**(3) Generate a tags file
**get-next-line**(2) (**C-x ‘**) Find the next command line
**git**(3) MicroEmacs GIT interface
**git-add**(3) MicroEmacs GIT interface - add items to the index
**git-blame**(3) MicroEmacs GIT interface - annotate a file
**git-cat-file**(3) MicroEmacs GIT interface - output specific commit of a file
**git-commit**(3) MicroEmacs GIT interface - commit changes to repository

**git-diff**(3) MicroEmacs GIT interface - diff changes
**git-fetch**(3) MicroEmacs GIT interface - download from another repository
**git-gdiff**(3) MicroEmacs GIT interface - graphical diff changes
**git-log-commit**(3) MicroEmacs GIT interface - show information about a specific commit
**git-log-file**(3) MicroEmacs GIT interface - list log of changes
**git-move**(3) MicroEmacs GIT interface - move an item
**git-pull**(3) MicroEmacs GIT interface - download and integrate another repository
**git-push**(3) MicroEmacs GIT interface - update remote repository
**git-remove**(3) MicroEmacs GIT interface - remove items
**git-revert**(3) MicroEmacs GIT interface - revert local changes
**git-setup**(3) MicroEmacs GIT interface setup
**git-status**(3) MicroEmacs GIT interface - status of items
**git-switch**(3) MicroEmacs GIT interface - switch or create branches
**git-unstage**(3) MicroEmacs GIT interface - unstage items
**ipipe-kill**(2) Kill a incremental pipe
**ipipe-shell-command**(2) (**esc backslash**) Incremental pipe (non-suspending system call)
**ipipe-write**(2) Write a string to an incremental pipe
**ishell**(3) Open an interactive shell window
**item-list**(3) (**F7**) Abbreviated search and list buffer contents
**item-list-close**(3) (**esc F7**) Close the item list
**item-list-find**(3) Find the selected item in the item list
**keep-buffer**(3) Rename the current buffer
**occur**(3) Regular expression search for occurrences
**perldb**(3) Perl Debugger
**pipe-shell-command**(2) (**esc @**) Execute a single operating system command
**rcs-file**(2) (**C-x C-q**) Handle Revision Control System (RCS) files
**rdiff**(3) Recursively compare files or directories
**recompile**(3) Rerun compilation process
**rgrep**(3) Execute recursive grep command
**shell**(2) [**C-x c**] Create a new command processor or shell
**shell-command**(2) Perform an operating system command
**shell-tool-setup**(3) Define shell tools
**svn**(3) MicroEmacs Subversion interface
**svn-add**(3) MicroEmacs SVN interface - add items
**svn-blame**(3) MicroEmacs SVN interface - annotate a file
**svn-cat-file**(3) MicroEmacs SVN interface - output specific version of a file
**svn-checkout**(3) MicroEmacs SVN interface - checkout a working copy
**svn-commit**(3) MicroEmacs SVN interface - commit changes back
**svn-copy**(3) MicroEmacs SVN interface - copy items
**svn-delete**(3) MicroEmacs SVN interface - delete items
**svn-diff**(3) MicroEmacs SVN interface - diff changes
**svn-gdiff**(3) MicroEmacs SVN interface - graphical diff changes
**svn-ignore**(3) MicroEmacs SVN interface - edit directory item ignore list
**svn-info**(3) MicroEmacs SVN interface - display info about an item
**svn-log-commit**(3) MicroEmacs SVN interface - get log for a commit
**svn-log-file**(3) MicroEmacs SVN interface - list log of changes
**svn-move**(3) MicroEmacs SVN interface - move an item
**svn-resolve**(3) MicroEmacs SVN interface - resolve conflicts
**svn-revert**(3) MicroEmacs SVN interface - revert local changes

**svn-server-copy**(3) MicroEmacs SVN interface - server based copy items
**svn-server-move**(3) MicroEmacs SVN interface - server based move items
**svn-setup**(3) MicroEmacs SVN interface setup
**svn-status**(3) MicroEmacs SVN interface - status of items
**svn-update**(3) MicroEmacs SVN interface - update directory files
**svn-xdiff**(3) MicroEmacs SVN interface - External diff changes
**vcs**(3) MicroEmacs VCS interface - VCS-agnostic front-end
**vcs-add**(3) MicroEmacs VCS interface - add items
**vcs-annotate**(3) MicroEmacs VCS interface - annotate a file
**vcs-blame**(3) MicroEmacs VCS interface - annotate a file
**vcs-cat-file**(3) MicroEmacs VCS interface - output a specific revision/commit of a file
**vcs-checkout**(3) MicroEmacs VCS interface - checkout (create) a working copy
**vcs-coci**(3) MicroEmacs VCS interface - check-out/check-in a file
**vcs-commit**(3) MicroEmacs VCS interface - commit changes to repository
**vcs-delete**(3) MicroEmacs VCS interface - delete/remove items
**vcs-diff**(3) MicroEmacs VCS interface - diff changes
**vcs-fetch**(3) MicroEmacs VCS interface - download from another repository
**vcs-gdiff**(3) MicroEmacs VCS interface - graphical diff changes
**vcs-id**(3) MicroEmacs VCS interface - identify the VCS controlling a file/directory
**vcs-log-commit**(3) MicroEmacs VCS interface - show log for a commit
**vcs-log-file**(3) MicroEmacs VCS interface - list log of changes
**vcs-move**(3) MicroEmacs VCS interface - move an item
**vcs-pull**(3) MicroEmacs VCS interface - download and integrate another repository
**vcs-push**(3) MicroEmacs VCS interface - update remote repository
**vcs-remove**(3) MicroEmacs VCS interface - remove items
**vcs-resolve**(3) MicroEmacs VCS interface - resolve conflicts
**vcs-revert**(3) MicroEmacs VCS interface - revert local changes
**vcs-setup**(3) MicroEmacs VCS interface - setup the underlying VCS
**vcs-status**(3) MicroEmacs VCS interface - status of items
**vcs-switch**(3) MicroEmacs VCS interface - switch or create branches
**vcs-unstage**(3) MicroEmacs VCS interface - unstage items
**vcs-update**(3) MicroEmacs VCS interface - update directory files
**xdiff**(3) External graphical file difference
**xfind**(3) Execute find command
**xgrep**(3) Execute grep command
**xrdb-scheme**(3) Load a XRDB scheme file from the user's config folder

### Variables

**$ME_PIPE_STDERR**(5) Command line diversion to stderr symbol
**$buffer-ipipe**(5) Divert buffer incremental pipe input through macro
**$buffer-ipipe-flags**(5) Incremental pipe creation flags
**$file-template**(5) Regular expression file search string
**$line-template**(5) Command line regular expression search string
**$rcs-ci-com**(5) RCS (and SCCS) check in command
**$rcs-cif-com**(5) RCS (and SCCS) check in first command
**$rcs-co-com**(5) RCS (and SCCS) check out command
**$rcs-cou-com**(5) RCS (and SCCS) check out unlock command
**$rcs-file**(5) RCS (and SCCS) file name

**$rcs-ue-com**(5) RCS (and SCCS) unedit file command
**$result**(5) Various command return values
**%compile-com**(5) Default system compile command line
**%cvs-com**(5) CVS Command line
**%cygwin-hilight**(5) Cygwin shell hilight enable flag
**%cygwin-path**(5) Cygwin root directory
**%cygwin-prompt**(5) Cygwin shell prompt
**%diff-com**(5) Diff command line
**%gdiff-com**(5) Gdiff command line
**%ishell**(5) User defined ishell shell
**%man-com**(5) Man Command line
**%os-chars**(5) Characters used by the OS
**%xdiff-com**(5) xdiff command line
**%xfind-com**(5) Find command line
**%xgrep-com**(5) Grep command line

## MESSAGE LINE COMMANDS

The message line appears at the bottom of the screen and is used for the input of commands and also to receive errors and information (see also Mode Line).

Commands and variables that interact with the message line include:

### Commands

**ml-bind-key**(2) Create key binding for message line
**ml-unbind-key**(2) Remove key binding from message line
**ml-write**(2) Write message on message line
**osd-bind-key**(2) Create key binding for OSD dialog
**osd-unbind-key**(2) Remove key binding from OSD dialog

### Variables

**$ml-scheme**(5) Message line color scheme

Ref: Message Line Commands File: me.0 Date: 2026/09/23

## WINDOW COMMANDS

MicroEmacs uses windows to display and allow you to edit the contents of buffers. Multiple windows may be present on the screen at once, each is separated by a mode line which describes the contents of the window above it.

You can scroll text vertically and horizontally within a window by using the cursor commands. Note that if a line of text extends beyond the boundary of a window, a dollar "**$**" sign is displayed instead of the first/last visible character.

Commands that operate on windows are defined as follows:

### Commands

**change-window-depth**(2) Change the depth of the current window
**change-window-width**(2) Change the width of the current window
**compare-windows**(2) Compare buffer windows, ignore whitespace
**compare-windows-exact**(3) Compare buffer windows, with whitespace
**create-frame**(2) (**C-x 5 1**, **C-x 5 2**) Create a new frame
**delete-frame**(2) (**C-x 5 0**) Delete the current frame
**delete-other-windows**(2) (**C-x 1**) Delete other windows
**delete-window**(2) (**C-x 0**) Delete current window
**grow-window-horizontally**(2) Enlarge current window horizontally (relative)
**grow-window-vertically**(2) Enlarge the current window (relative change)
**next-frame**(2) (**C-x 5 o**) Change the focus to the next frame
**next-window**(2) (**C-x o**) Move the cursor to the next window
**next-window-find-buffer**(2) [] Split the current window and show new buffer
**next-window-find-file**(2) (**C-x 4**) Split the current window and find file
**page-file-scroll-down**(3) (**C-n**) Scroll paged buffer down
**page-file-scroll-up**(3) (**C-p**) Scroll paged buffer up
**previous-window**(2) (**C-x p**) Move the cursor to the previous window
**resize-all-windows**(2) Resize all windows (automatic change)
**resize-window-horizontally**(2) Resize current window horizontally (absolute)
**resize-window-vertically**(2) Resize the current window (absolute change)
**scroll-down**(2) (**C-n**) Move the window down (scrolling)
**scroll-left**(2) (**C-x <**) Move the window left (scrolling)
**scroll-next-window-down**(2) (**esc C-v**) Scroll next window down
**scroll-next-window-up**(2) (**esc C-z**) Scroll next window up
**scroll-right**(2) (**C-x >**) Move the window right (scrolling)
**scroll-up**(2) (**C-p**) Move the window up (scrolling)
**set-title**(3) Set frame title prefix
**shrink-window-horizontally**(2) Shrink current window horizontally (relative)
**shrink-window-vertically**(2) Shrink the current window (relative change)
**split-window-horizontally**(2) (**C-x 3**) Split current window into two (horizontally)
**split-window-vertically**(2) (**C-x 2**) Split the current window into two

### Variables

**$frame-id**(5) Current frame identity
**$frame-title**(5) Frame title prefix

**$scroll**(5) Screen scroll control
**$scroll-bar**(5) Scroll bar configuration
**$window-acol**(5) Window cursor actual column
**$window-aline**(5) Window cursor actual line
**$window-col**(5) Window cursor column (no expansion)
**$window-depth**(5) Number of text lines in a window
**$window-eline**(5) Window end line
**$window-id**(5) Current window identity
**$window-line**(5) Window cursor line
**$window-mark-col**(5) Window mark column
**$window-mark-line**(5) Window mark line
**$window-mode-line**(5) Window mode line position
**$window-pid**(5) Previous window identity
**$window-scroll-bar**(5) Window scroll bar (or separator) position
**$window-wcol**(5) Window cursor column (historic)
**$window-width**(5) Number of character columns in a window
**$window-wline**(5) Window cursor line (historic)
**$window-x-scroll**(5) Current window X scroll
**$window-xcl-scroll**(5) Current window current line X scroll
**$window-y-scroll**(5) Current window Y scroll

## PRINTING COMMANDS

Printing within MicroEmacs is fairly restrictive, the following commands are used in conjunction with the print facility.

**print-buffer**(2) Print buffer, with formatting
**print-color**(2) Create a new printer color
**print-region**(2) Print region, with formatting
**print-scheme**(2) Create a new printer color and font scheme
**print-setup**(3) Configure MicroEmacs's printer interface

## VARIABLE FUNCTIONS

Functions can be used to manipulate variables in various ways. Functions have in general one, two, or three arguments. These arguments will always be placed after the function on the current command line. For example, if we wanted to increase the current fill column by two, using MicroEmacs's **set-variable**(2) (C-x v) command, we would write:

```
set-variable $fill-col &add $fill-col 2
     \             \      \        \     \____second operand
      \             \      \        \_________first operand
       \             \      \__________________function to execute
        \             \_____________________variable to set
         \___________________________________set-variable (C-x v) command
```

Functions always begin with an ampersand (**&**) character, and are only significant to the first 3 characters after the ampersand. Functions will normally expect one of three types of arguments, and will automatically convert types when needed.

**<num>**
: An ASCII string of digits which is interpreted as a numeric value. Any string which does not start with a digit or a minus sign (-) will be considered zero.

**<str>** An arbitrary string of characters. At the moment, strings are limited to 128 characters in length.

**<log>** A logical value consisting of a non-zero number for **TRUE** (e.g. "`1`"), anything else is considered **FALSE** (e.g. "`0`" or "`foo`"). This differs from earlier versions of MicroEmacs where the strings "`TRUE`" and "`FALSE`" were used.

A list of the currently available functions follows:

**Numeric Functions** (returns <*num*>)

**&abs**(4) Absolute value of a number
**&add**(4) Add two numbers
**&dec**(4) Pre-decrement variable
**&divide**(4) Division of two numbers
**&fabs**(4) Absolute value of a floating point number
**&fadd**(4) Floating point add two numbers
**&fdivide**(4) Floating point division of two numbers
**&fmultiply**(4) Floating point multiply two numbers
**&fsqrt**(4) Floating point suqare root of a number
**&fsub**(4) Floating point subtract two numbers
**&inc**(4) Pre-increment variable
**&mod**(4) Modulus of two numbers
**&multiply**(4) Multiply two numbers
**&negate**(4) Negation of two numbers
**&pdec**(4) Post-decrement variable
**&pinc**(4) Post-increment variable
**&sub**(4) Subtract two numbers

**Bitwise Functions** (returns <*num*>)

**&band**(4) Bitwise AND operator
**&bnot**(4) Bitwise NOT operator

**&bor**(4) Bitwise OR operator
**&bxor**(4) Bitwise XOR operator

**String manipulation functions** (returns <*str*>)

**&cat**(4) Concatenate two strings together
**&left**(4) Return the left most characters from a string
**&len**(4) Return the length of a string
**&mid**(4) Return a portion (middle) of a string
**&right**(4) Return the right most characters from a string
**&slower**(4) Return the string converted to lower case
**&sprintf**(4) Formatted string construction
**&supper**(4) Return the string converted to upper case
**&trboth**(4) Return string trimmed of white chars on both sides
**&trleft**(4) Return string trimmed of white chars on left side
**&trright**(4) Return string trimmed of white chars on right side

**List manipulation functions** (returns <*str*>)

**&ldel**(4) Delete list item
**&lfind**(4) Find list item
**&lget**(4) Get list item
**&linsert**(4) Insert list item
**&llength**(4) Get list item count
**&lset**(4) Set list item

**Logical testing functions** (returns <*log*>)

**&and**(4) Logical AND operator
**&equal**(4) Numerical equivalence operator
**&great**(4) Numerical greater than operator
**&isequal**(4) Case insensitive String equivalence operator
**&less**(4) Numerical less than operator
**&not**(4) Logical NOT operator
**&or**(4) Logical OR operator
**&sequal**(4) String equivalence operator
**&sgreat**(4) String greater than operator
**&sless**(4) String less than operator
**&xisequal**(4) Case insensitive regex String equivalence operator
**&xsequal**(4) Regex string equivalence operator

**Miscellaneous functions**

**&atoi**(4) ASCII to integer conversion
**&bmode**(4) Determine buffer mode
**&bstat**(4) Retrieve current buffer statistic
**&cbind**(4) Return the command a key is bound to
**&cond**(4) Conditional expression operator
**&exist**(4) Test if a variable or command exists
**&find**(4) Find a file on the search path
**&gmode**(4) Determine global mode

**&indirect**(4) Evaluate a string as a variable
**&inword**(4) Test for a word character
**&irep**(4) Case insensitive replace string in string
**&isin**(4) Case insensitive test for string in string
**&itoa**(4) Integer to ASCII conversion
**&kbind**(4) Return the key a command is bound to
**&nbind**(4) Return the numerial argument of a binding
**&nbmode**(4) Determine named buffer mode
**&opt**(4) MicroEmacs optional feature test
**&rep**(4) Replace string in string
**&risin**(4) Recursive case insensitive test for string in string
**&rsin**(4) Recursively test for string in string
**&set**(4) In-line macro variable assignment
**&sin**(4) Test for string in string
**&stat**(4) Retrieve a file statistic
**&uci**(4) Charater-set character Unicode info
**&ufi**(4) UTF-8 character Unicode info
**&uni**(4) Unicode character information
**&which**(4) Find a program on the path
**&xirep**(4) Regex case insensitive Replace string in string
**&xrep**(4) Regex replace string in string

## SEE ALSO

Directives

## MODE LINE

The **modeLine** is the line of text shown at the bottom of each window identifying the window being displayed. The mode line typically contains the name of the buffer, the file and operating mode associated with the buffer displayed in the window. e.g.

```
#*-me (BEI) - 250 foo (foo.txt) ----------------
```

**Note** that only a sub-set of the modes are visible on the mode line, this reduces the cluttering of the mode line. Use the **about**(2) command to determine the exact modes that are active.

The text content and color of the mode line is fully configurable and may be defined in the users start-up script. The currently active buffer is generally indicated by the state of the mode line, e.g. highlighted color (typically red) and/or double line separator characters **===**; conversely inactive buffers are shown in a dull color with single line separator characters **—**.

The mode line may be configured or interrogated using the following variables:

**$buffer-mode-line**(5) Buffer mode line string
**$mode-chars**(5) Mode characters
**$mode-line**(5) Mode line format
**$mode-line-scheme**(5) Mode line color scheme
**$show-modes**(5) Select buffer modes to display
**$window-mode-line**(5) Window mode line position
**$window-scroll-bar**(5) Window scroll bar (or separator) position

Ref: Mode Line File: me.0 Date: 2026/09/23

## GAMES

The following is a list of all of the games provided by **MicroEmacs**:

**Arkanoid**(3) MicroEmacs version of the bat and ball wall game
**Balls**(3) MicroEmacs bouncing ball playground
**Hearts**(3) MicroEmacs version of the card game Hearts
**Mahjongg**(3) MicroEmacs version of the solitaire Mah Jongg game
**Match-It**(3) MicroEmacs version of the Match-It game
**Metris**(3) MicroEmacs version of the falling blocks game
**Patience**(3) MicroEmacs version of Patience (or Solitaire)
**Triangle**(3) MicroEmacs version of Triangle patience game

Ref: MicroEmacs - Games File: me.0 Date: 2026/09/23

## SUPPORTED FILE TYPES

The file types currently supported by MicroEmacs are defined in the following list. Other file types may be supported by definition of an appropriate hook function to handle the file, see **fileHooks**(2).

**0-9**(9) UNIX t/nroff file
**ada**(9) Ada programming language
**adb**(9) Ada programming language
**adoc**(9) AsciiDoc Markup File
**ads**(9) Ada programming language
**apache**(9) Apache httpd.conf configuration file
**asciidoc**(9) AsciiDoc Markup File
**asm**(9) Assembler File
**asn.1**(9) ASN.1 file
**asp**(9) Active Server Pages
**au3**(9) Autoit 3 Scripting Language File
**autoit**(9) Autoit 3 Scripting Language File
**awk**(9) AWK File
**bas**(9) Visual Basic
**bat**(9) MS-DOS Batch File
**bib**(9) TeX Bibliography File
**bnf**(9) Backus-Naur Form
**btm**(9) 4-DOS Batch File
**C#**(9) C# programming language
**C3**(9) C3 programming language
**c**(9) C programming language
**cbl**(9) Cobol (85) File
**cc**(9) C++ programming language
**cfc**(9) ColdFusion Markup Language
**cfm**(9) ColdFusion Markup Language
**cfml**(9) Adobe Macromedia ColdFusion Markup Language
**cls**(9) Visual Basic
**cpp**(9) C++ programming language
**cs**(9) C# programming language
**csh**(9) C-Shell file
**dart**(9) Dart programming language
**def**(9) C or C++ definition file
**doc**(9) ASCII plain text document file
**doctools**(9) Tcl Doctools Markup File
**dot**(9) Graphviz Dot Diagram Language File
**e**(9) Euphoria File
**ehf**(9) MicroEmacs help file
**emf**(9) MicroEmacs Macro File
**erf**(9) MicroEmacs registry file
**erl**(9) Erlang programming language
**euphoria**(9) Euphoria File
**f**(9) Fortran File
**f77**(9) Fortran 77 File
**f90**(9) Fortran 90 File

**fu**(9) Fusion transpiler programming language
**fvwm**(9) FVWM configuration file
**fvwmrc**(9) FVWM configuration file
**gawk**(9) GNU AWK File
**go**(9) Go programming language
**graphviz**(9) Graphviz Dot Diagram Language File
**groovy**(9) Groovy programming language
**h**(9) C programming language header
**haskell**(9) Haskell programming language
**hpj**(9) MS-Windows Help Project File
**hs**(9) Haskell programming language
**htm**(9) HyperText Markup Language File
**html**(9) HyperText Markup Language File
**httpd.conf**(9) Apache configuration file
**hx**(9) Haxe transpiler programming language
**i**(9) C/C++ preprocessor output file
**imakefile**(9) Make file
**info**(9) GNU Info file
**ini**(9) MS-Windows Initialization File
**jav**(9) Java programming language
**java**(9) Java programming language
**javascript**(9) Javascript programming language
**jl**(9) Julia File
**js**(9) Javascript programming language
**jscript**(9) Javascript programming language
**jst**(9) JASSPA Structured Text
**julia**(9) Julia File
**kotlin**(9) Kotlin programming language
**ksh**(9) Korn shell file
**kt**(9) Kotlin programming language
**ktm**(9) Kotlin programming language
**kts**(9) Kotlin programming language
**l**(9) LEX programming language
**latex**(9) TeX Documentation
**login**(9) Shell user login file
**lua**(9) Lua File
**MetaFont**(9) MetaFont/MetaPost File
**m**(9) GNU Octave / Matlab File
**m4**(9) M4 Macro Processor
**makefile**(9) Make file
**man**(9) UNIX Manual Page
**markdown**(9) Markdown Markup File
**matlab**(9) GNU Octave / Matlab File
**md**(9) Markdown Markup File
**mf**(9) MetaFont File
**mp**(9) MetaPost File
**nawk**(9) New AWK File
**nim**(9) Nim programming language
**nroff**(9) UNIX nroff file

**octave**(9) GNU Octave / Matlab File
**PowerShell**(9) PowerShell Programming Language
**p**(9) Pascal File
**pas**(9) Pascal File
**perl**(9) Practical Extraction and Report Language File
**php**(9) PHP: Hypertext Preprocessor Language
**php3**(9) PHP: Hypertext Preprocessor Language
**php4**(9) PHP: Hypertext Preprocessor Language
**pl**(9) Practical Extraction and Report Language File
**plantuml**(9) PlantUML Diagram Language File
**pm**(9) Practical Extraction and Report Language File
**pml**(9) PlantUML Diagram Language File
**profile**(9) Shell user profile
**ps1**(9) PowerShell File
**py**(9) Python Language File
**python**(9) Python Language File
**qd**(9) Quarkdown Markup File
**quarkdown**(9) Quarkdown Markup File
**r**(9) R Language File
**rb**(9) Ruby programming language
**rc**(9) Microsoft Developer resource file
**reg**(9) Registry file
**rgy**(9) Registry file
**rmarkdown**(9) R-Markdown Markup File
**rmd**(9) R-Markdown Markup File
**rs**(9) Rust programming language
**ruby**(9) Ruby programming language
**rul**(9) Install Shield Rules
**rust**(9) Rust programming language
**s**(9) Assembler File
**samba**(9) Samba smb.conf configuration file
**scala**(9) Scala programming language
**sch**(9) Scheme File
**scheme**(9) Scheme File
**scm**(9) Scheme File
**sh**(9) Bourne shell file
**smb.conf**(9) Samba configuration file
**so**(9) UNIX t/nroff include file
**sql**(9) SQL File
**sty**(9) TeX Style File
**swift**(9) Swift programming language
**tcl**(9) TCL programming language
**tcshrc**(9) T-Shell start up file
**tex**(9) TeX Documentation
**texi**(9) GNU Texinfo documentation file
**texinfo**(9) GNU Texinfo documentation file
**tk**(9) TK programming language
**tman**(9) Tcl Doctools Markup File
**tni**(9) UNIX t/nroff include file

**toml**(9) Tom's obvious minimal language file
**troff**(9) UNIX troff file
**ts**(9) Typescript programming language
**txt**(9) ASCII plain text file
**typ**(9) Typst Markup File
**typescript**(9) Typescript programming language
**typst**(9) Typst Markup File
**v**(9) Verilog hardware description language
**vala**(9) Vala programming language
**vb**(9) Visual Basic
**verilog**(9) Verilog hardware description language
**vhdl**(9) VHDL hardware simulation File
**vlang**(9) V programming language
**vrml**(9) VRML File
**wiki**(9) GNU Wiki File
**wish**(9) TCL shell file
**x86**(9) Intel .x86 Assembler File
**xml**(9) Extensible Markup Language
**xsl**(9) Extensible Stylesheet Language
**xul**(9) XML User Interface Language
**y**(9) YACC programming language
**zig**(9) Zig programming language
**zsh**(9) Z-Shell file

## COMMAND GLOSSARY

The following is a list of all of the commands (built-in and macro) provided by **MicroEmacs** [See split listing]:

**Arkanoid**(3) MicroEmacs version of the bat and ball wall game
**abbrev-list**(3) Create a list of abbreviations
**abort-command**(2) (**C-g**) Abort command
**about**(2) Information About MicroEmacs
**add-color**(2) Create a new color
**add-color-scheme**(2) Create a new color scheme
**add-dictionary**(2) Declare existence of a spelling dictionary
**add-file-hook**(2) DEPRICATED: Declare file name context dependent configuration
**add-global-mode**(3) Set a global buffer mode
**add-mode**(3) Set a local buffer mode
**add-next-line**(2) Define the searching behavior of command output
**add-spell-rule**(2) Add a new spelling rule to the dictionary
**alarm**(3) Set an alarm
**alias-path-setup**(3) Define alternative path alias
**aman**(3) Compile an nroff file into a buffer (UNIX)
**append-buffer**(2) Write contents of buffer to end of named file
**ascii-time**(3) Return the current time as a string
**auto-spell**(3) Auto-spell support
**auto-spell-buffer**(3) Auto-spell whole buffer
**auto-spell-ignore**(3) Auto-spell ignore current word
**auto-spell-menu**(3) Auto-spell context menu
**auto-spell-reset**(3) Auto-spell hilight reset
**Balls**(3) MicroEmacs bouncing ball playground
**backward-char**(2) (**C-b**) Move the cursor left
**backward-delete-char**(2) (**backspace**) Delete the previous character at the cursor position
**backward-delete-tab**(2) (**S-tab**) Delete white space to previous tab-stop
**backward-kill-word**(2) (**esc backspace**) Delete the previous word at the cursor position
**backward-line**(2) (**C-p**) Move the cursor to the previous line
**backward-paragraph**(2) (**esc p**) Move the cursor to the previous paragraph
**backward-word**(2) (**esc b**) Move the cursor to the previous word
**beginning-of-buffer**(2) (**esc <**) Move to beginning of buffer/file
**beginning-of-line**(2) (**C-a**) Move to beginning of line
**bin-copy-region**(3) Binary supporting copy region
**bin-yank**(3) Binary supporting kill buffer insert
**browse**(3) Browse a file or buffer in view mode
**buffer-abbrev-file**(2) Set buffers' abbreviation file
**buffer-bind-key**(2) Create local key binding for current buffer
**buffer-info**(2) (**C-x =**) Status information on current buffer position
**buffer-init**(3) Initialize a buffer language template
**buffer-init-fhook**(3) Initialize the buffer file hooks
**buffer-init-hooks**(3) Initialize the buffer language template hooks
**buffer-is-utf8**(3) Detect if the current buffer is UTF-8 encoded
**buffer-major-mode**(3) Change the operating mode of a buffer
**buffer-mode**(2) (**C-x m**) Change a local buffer mode
**buffer-setup**(3) Configure the current buffer settings

**buffer-unbind-key**(2) Remove local key binding for current buffer
**ClearCase**(3) MicroEmacs ClearCase interface
**c-deslashify**(3) C language, remove backslash to end of lines of region
**c-hash-define**(3) Set a C/C++ #define
**c-hash-eval**(3) Evaluate C/C++ #defines
**c-hash-undef**(3) Unset a C/C++ #define
**c-hash-undo**(3) Remove C/C++ #define evaluation
**c-slashify**(3) C language, append backslash to end of lines of region
**c-to-cpp-convert**(3) Convert a C file to C++
**calc**(3) Calculator
**calc-op**(3) Low level calculator command
**capitalize-word**(2) (**esc c**) Capitalize word
**cc-add**(3) MicroEmacs ClearCase interface - add file
**cc-checkin**(3) MicroEmacs ClearCase interface - checkin files
**cc-checkout**(3) MicroEmacs ClearCase interface - checkout files
**cc-delete**(3) MicroEmacs ClearCase interface - delete files
**cc-diff**(3) MicroEmacs ClearCase interface - diff changes
**cc-setup**(3) MicroEmacs ClearCase interface setup
**cc-state**(3) MicroEmacs ClearCase interface - list state of directory files
**cc-update**(3) MicroEmacs ClearCase interface - update directory files
**change-buffer-charset**(3) Convert buffer between two character sets
**change-buffer-name**(2) (**esc C-n**) Change name of current buffer
**change-company**(3) Change current company profile
**change-file-name**(2) (**C-x n**) Change the file name of the current buffer
**change-font**(2) Change the screen font
**change-font-size**(3) Incrementally change font size
**change-frame-depth**(2) Change the number of lines on the current frame
**change-frame-width**(2) Change the number of columns on the current frame
**change-scheme**(3) Change the current highlight scheme
**change-window-depth**(2) Change the depth of the current window
**change-window-width**(2) Change the width of the current window
**check-line-length**(3) Check the length of text lines are valid
**clean**(3) Remove redundant white spaces from the current buffer
**collapse-all**(3) (**f3**) (Un)Fold all regions in the current buffer
**collapse-current**(3) (**f2**) (un)Fold a region in the current buffer
**command-apropos**(2) (**C-h a**) List commands involving a concept
**command-wait**(2) Conditional wait command
**compare-windows**(2) Compare buffer windows, ignore whitespace
**compare-windows-exact**(3) Compare buffer windows, with whitespace
**compile**(3) Start a compilation process
**copy-file**(3) Copy or move files and directories
**copy-rectangle**(3) Copy a rectangular region of text
**copy-region**(2) (**esc w**) Copy a region of the buffer
**copy-registry**(3) Recursive copy of registry subtree
**count-words**(2) (**esc C-c**) Count the number of words in a region
**create-callback**(2) Create a timer callback
**create-frame**(2) (**C-x 5 1**, **C-x 5 2**) Create a new frame
**cvs**(3) MicroEmacs CVS interface
**cvs-add**(3) MicroEmacs CVS interface - add file

**cvs-annotate**(3) MicroEmacs CVS interface - annotate file
**cvs-cat-file**(3) MicroEmacs CVS interface - output specific revision of a file
**cvs-checkout**(3) MicroEmacs CVS interface - checkout files
**cvs-commit**(3) MicroEmacs CVS interface - commit changes
**cvs-diff**(3) MicroEmacs CVS interface - diff changes
**cvs-file-status**(3) MicroEmacs CVS interface - status of files
**cvs-gdiff**(3) MicroEmacs CVS interface - graphical diff changes
**cvs-log-file**(3) MicroEmacs CVS interface - log changes
**cvs-remove**(3) MicroEmacs CVS interface - remove file
**cvs-resolve**(3) MicroEmacs CVS interface - resolve conflicts
**cvs-resolve-conflicts**(3) MicroEmacs CVS interface - resolve conflicts
**cvs-setup**(3) MicroEmacs CVS interface setup
**cvs-status**(3) MicroEmacs CVS interface - list state of directory files
**cvs-tag**(3) MicroEmacs CVS interface - tag files
**cvs-update**(3) MicroEmacs CVS interface - update directory files
**cygwin**(3) Open a Cygwin BASH window
**define-help**(2) Define help information
**define-macro**(2) Define a new macro
**define-macro-file**(2) Define macro file location
**delete-blank-lines**(2) (**C-x C-o**) Delete blank lines about cursor
**delete-buffer**(2) (**C-x k**) Delete a buffer
**delete-dictionary**(2) Remove a spelling dictionary from memory
**delete-file**(3) Delete files and directories
**delete-file-delayed**(3) Delete file after a specified delay
**delete-frame**(2) (**C-x 5 0**) Delete the current frame
**delete-global-mode**(3) Remove a global buffer mode
**delete-hilight-lines**(3) Delete hilight lines
**delete-indentation**(3) Join 2 lines deleting white spaces
**delete-mode**(3) Remove a local buffer mode
**delete-other-windows**(2) (**C-x 1**) Delete other windows
**delete-registry**(2) Delete a registry tree
**delete-some-buffers**(2) Delete buffers with query
**delete-window**(2) (**C-x 0**) Delete current window
**describe-bindings**(2) (**C-h b**) Show current command/key binding
**describe-key**(2) (**C-x ?**) Report keyboard key name and binding
**describe-variable**(2) (**C-h v**) Describe current setting of a variable
**describe-word**(3) Display a dictionary definition of a word
**diff**(3) Difference files or directories
**diff-changes**(3) Find the differences from a previous edit session
**diff-lines**(3) Highlight the differences between two lines
**directory-tree**(2) Draw the file directory tree
**display-white-chars**(3) Toggle the displaying of white characters
**display-whitespaces**(3) Toggle the displaying of white characters
**draw**(3) Simple line drawing utility
**edit-dictionary**(3) Insert a dictionary in a buffer
**end-kbd-macro**(2) (**C-x )**) Stop recording keyboard macro
**end-of-buffer**(2) (**esc >**) Move to end of buffer/file
**end-of-line**(2) (**C-e**) Move to end of line
**exchange-point-and-mark**(2) (**C-x C-x**) Exchange the cursor and marked position

**execute-buffer**(2) Execute script lines from a buffer
**execute-file**(2) (**esc /**) Execute script lines from a file
**execute-kbd-macro**(2) (**C-x e**) Execute a keyboard macro
**execute-line**(2) Execute a typed in script line
**execute-named-command**(2) [**esc x**] Execute a named command
**execute-string**(2) Execute a string as a command
**execute-tool**(3) Execute a user defined shell tool
**exit-emacs**(2) Exit MicroEmacs
**expand-abbrev**(2) Expand an abbreviation
**expand-abbrev-handle**(3) (**esc esc**) Expand an abbreviation handler
**expand-look-back**(3) Complete a word by looking back for a similar word
**expand-word**(3) Complete a word by invocation of the speller
**file-attrib**(3) Set the current buffers system file attributes
**file-browser**(3) (**f10**) Browse the file system
**file-browser-close**(3) Close the file-browser
**file-browser-swap-buffers**(3) Swap between file-browser windows
**file-exec**(3) Executes the current buffer with compiler/interpreter
**file-format**(3) Format the current buffer using the a formatter
**file-lint**(3) check the current buffer for problems using a linter
**file-list**(3) Generate list of matching file names
**file-op**(2) File system operations command
**file-tool-setup**(3) Define file browser tools
**file-type-setup**(3) Define file type set
**filemask-to-regex**(3) Convert a file mask to a regex
**fill-paragraph**(2) (**esc o**) Format a paragraph
**filter-buffer**(2) (**C-x #**) Filter the current buffer through an O/S command
**find**(3) Execute built-in find command
**find-bfile**(3) (**C-x 9**) Load a file as binary data
**find-buffer**(2) (**C-x b**) Switch to a named buffer
**find-cfile**(3) Load an encrypted file
**find-file**(2) (**C-x C-f**) Load a file
**find-hfile**(3) (**C-x C-h**) Load a file using selected handler
**find-lfile**(3) Load a file lossless
**find-registry**(2) Index search of a registry sub-tree
**find-rfile**(3) Load a file as binary data (reduced)
**find-setup**(3) Setup the built-in find and grep commands
**find-tag**(2) (**esc t**) Find tag, auto-load file and move to tag position
**find-ufile**(3) Load a Unicode file
**find-word**(3) Find a word using spelling dictionaries
**find-zfile**(3) Compressed file support
**fold-close**(3) (**C-f2**) Close an open fold (in-line)
**fold-create**(3) (**C-f2**) Create a new fold
**fold-enter**(3) (**f3**) Enter fold
**fold-exit**(3) (**esc f3**) Exit a fold
**fold-exit-all**(3) (**esc C-f3**) Exit all folds
**fold-kill**(3) (**C-x C-k**) Delete the fold and all its contents
**fold-open**(3) (**f2**) Open fold (in-line)
**fold-remove**(3) (**esc C-f3**) Remove the fold, maintaining content
**forward-char**(2) (**C-f**) Move the cursor right

**forward-delete-char**(2) (**C-d**) Delete the next character at the cursor position
**forward-kill-word**(2) (**esc d**) Delete the next word at the cursor position
**forward-line**(2) (**C-n**) Move the cursor to the next line
**forward-paragraph**(2) (**esc n**) Move the cursor to the next paragraph
**forward-word**(2) (**esc f**) Move the cursor to the next word
**ftp**(3) Initiate an FTP connection
**gdiff**(3) Graphical file difference
**generate-hash**(2) Generate SHA1 or SHA256 hash or HMAC
**generate-password**(3) Generate a password with given criteria
**generate-tags-file**(3) Generate a tags file
**get-next-line**(2) (**C-x ‘**) Find the next command line
**git**(3) MicroEmacs GIT interface
**git-add**(3) MicroEmacs GIT interface - add items to the index
**git-blame**(3) MicroEmacs GIT interface - annotate a file
**git-cat-file**(3) MicroEmacs GIT interface - output specific commit of a file
**git-commit**(3) MicroEmacs GIT interface - commit changes to repository
**git-diff**(3) MicroEmacs GIT interface - diff changes
**git-fetch**(3) MicroEmacs GIT interface - download from another repository
**git-gdiff**(3) MicroEmacs GIT interface - graphical diff changes
**git-log-commit**(3) MicroEmacs GIT interface - show information about a specific commit
**git-log-file**(3) MicroEmacs GIT interface - list log of changes
**git-move**(3) MicroEmacs GIT interface - move an item
**git-pull**(3) MicroEmacs GIT interface - download and integrate another repository
**git-push**(3) MicroEmacs GIT interface - update remote repository
**git-remove**(3) MicroEmacs GIT interface - remove items
**git-revert**(3) MicroEmacs GIT interface - revert local changes
**git-setup**(3) MicroEmacs GIT interface setup
**git-status**(3) MicroEmacs GIT interface - status of items
**git-switch**(3) MicroEmacs GIT interface - switch or create branches
**git-unstage**(3) MicroEmacs GIT interface - unstage items
**global-abbrev-file**(2) Set global abbreviation file
**global-bind-key**(2) (**esc k**) Bind a key to a named command or macro
**global-mode**(2) (**esc m**) Change a global buffer mode
**global-unbind-key**(2) (**esc C-k**) Unbind a key from a named command or macro
**goto-alpha-mark**(2) (**C-x a**) Move the cursor to a alpha marked location
**goto-line**(2) (**esc g**) Move the cursor to specified line
**goto-matching-fence**(2) (**esc C-f**) Move the cursor to matching fence
**goto-position**(2) Restore a stored position
**goto-window**(2) Restore a saved window to the current window (historic)
**grep**(3) Search a file for a pattern
**grep-for-region**(3) Grep current directory for current region
**grow-window-horizontally**(2) Enlarge current window horizontally (relative)
**grow-window-vertically**(2) Enlarge the current window (relative change)
**Hearts**(3) MicroEmacs version of the card game Hearts
**help**(2) (**esc ?**) Help; high level introduction to help
**help-command**(2) (**C-h C-c**) Help; command information
**help-item**(2) (**C-h C-i**) Help; item information
**help-variable**(2) (**C-h C-v**) Help; variable information
**hilight**(2) Manage the buffer hilighting schemes

**hunt-backward**(2) (**C-x C-h**) Resume previous search in backward direction
**hunt-forward**(2) (**C-x h**) Resume previous search in forward direction
**ifill-paragraph**(3) (**esc q**) Format a paragraph
**indent**(2) Manage the auto-indentation methods
**indent-decrease**(3) Decrease the line indentation
**indent-increase**(3) Increase the line indentation
**info**(3) Display a GNU Info database
**info-goto-link**(3) Display Info on a given link
**info-on**(3) Display Info on a given topic
**insert-calc**(3) Calculator with answer insertion
**insert-file**(2) (**C-x C-i**) Insert file into current buffer
**insert-file-name**(2) (**C-x C-y**) Insert filename into current buffer
**insert-macro**(2) Insert keyboard macro into buffer
**insert-newline**(2) (**C-o**) Insert new line at cursor position
**insert-password**(3) Create and insert a new password string
**insert-space**(2) Insert space(s) into current buffer
**insert-string**(2) Insert character string into current buffer
**insert-symbol**(3) Insert an ASCII character
**insert-tab**(2) (**C-i**) Insert tab(s) into current buffer
**insert-template**(3) Insert template file into current buffer
**ipipe-kill**(2) Kill a incremental pipe
**ipipe-shell-command**(2) (**esc backslash**) Incremental pipe (non-suspending system call)
**ipipe-write**(2) Write a string to an incremental pipe
**isearch-backward**(2) (**C-r**) Search backwards incrementally (interactive)
**isearch-forward**(2) (**C-s**) Search forward incrementally (interactive)
**ishell**(3) Open an interactive shell window
**kbd-macro-query**(2) (**C-x q**) Query termination of keyboard macro
**keep-buffer**(3) Rename the current buffer
**kill-line**(2) (**C-k**) Delete all characters to the end of the line
**kill-paragraph**(2) Delete a paragraph
**kill-rectangle**(2) (**esc C-w**) Delete a column of text
**kill-region**(2) (**C-w**) Delete all characters in the marked region
**line-scheme**(3) Apply line annotation in current buffer
**list-buffers**(2) (**C-x C-b**) List all buffers and show their status
**list-commands**(2) (**C-h c**) List available commands
**list-registry**(2) Display the registry in a buffer
**list-variables**(2) (**C-h v**) List defined variables
**lower-case-region**(2) (**C-x C-l**) Lowercase a region (downcase)
**lower-case-word**(2) (**esc l**) Lowercase word (downcase)
**Mahjongg**(3) MicroEmacs version of the solitaire Mah Jongg game
**MainMenu**(3) The top main menu
**Match-It**(3) MicroEmacs version of the Match-It game
**Metris**(3) MicroEmacs version of the falling blocks game
**mail**(3) Compose and send an email
**mail-check**(3) Check for new email
**mail-setup**(3) Email setup dialog
**major-mode**(2) Declare or set a buffer major mode
**major-mode-help**(3) Displays help page for current buffer
**major-mode-setup**(3) Configures the current buffer settings

**major-mode-tools-setup**(3) File major-mode tool setup
**man**(3) UNIX manual page viewer
**man-clean**(3) Clean UNIX manual page
**mark-registry**(2) Modify the operating mode of a registry node
**ml-bind-key**(2) Create key binding for message line
**ml-unbind-key**(2) Remove key binding from message line
**ml-write**(2) Write message on message line
**name-kbd-macro**(2) Assign a name to the last keyboard macro
**named-buffer-mode**(2) Change a named buffer mode
**narrow-buffer**(2) Hide buffer lines
**narrow-search-backward**(3) (**C-c r**) Search for a string in the backward direction
**narrow-search-forward**(3) (**C-c s**) Search for a string in the forward direction
**newline**(2) (**return**) Insert a new line
**next-buffer**(2) (**C-x x**) Switch to the next buffer
**next-frame**(2) (**C-x 5 o**) Change the focus to the next frame
**next-window**(2) (**C-x o**) Move the cursor to the next window
**next-window-find-buffer**(2) [] Split the current window and show new buffer
**next-window-find-file**(2) (**C-x 4**) Split the current window and find file
**normal-tab**(3) Insert a normal tab
**notes**(3) Tagged hierarchical database of notes
**notes-close**(3) Close the notes window
**notes-context**(3) Create/Access buffer context note
**notes-file**(3) Open a notes database file
**notes-import-skwyrul**(3) Import a Skwyrul index file (skwyrul_l.htm)
**notes-instant**(3) Create quick note
**open-line**(3) (**C-o**) Open new line at cursor position
**organizer**(3) Calendar and address organizer
**osd**(2) Manage the On-Screen Display
**osd-bind-key**(2) Create key binding for OSD dialog
**osd-dialog**(3) OSD dialog box
**osd-entry**(3) OSD entry dialog box
**osd-help**(3) GUI based on-line help
**osd-search**(3) Search and annotate the current buffer
**osd-unbind-key**(2) Remove key binding from OSD dialog
**osd-xdialog**(3) OSD Extended dialog box
**Patience**(3) MicroEmacs version of Patience (or Solitaire)
**page-bfile**(3) Load a large binary file as binary data
**page-file**(3) Load a large file in page mode
**page-file-backward-line**(3) (**C-p**) Move backwards one line in paged buffer
**page-file-beginning-of-buffer**(3) Move to the start of the paged file
**page-file-end-of-buffer**(3) Move to the end of the paged file
**page-file-forward-line**(3) (**C-n**) Move forwards one line in paged buffer
**page-file-goto-byte**(3) Page to the buffer byte offset
**page-file-goto-page**(3) Move to page
**page-file-isearch-backward**(3) (**C-r**) Page search backwards incrementally (interactive)
**page-file-isearch-forward**(3) (**C-s**) Page search forward incrementally (interactive)
**page-file-save-buffer**(3) (**C-x C-s**) Save contents of a page mode buffer to file
**page-file-scroll-down**(3) (**C-n**) Scroll paged buffer down
**page-file-scroll-up**(3) (**C-p**) Scroll paged buffer up

**page-file-search-backward**(3) (**C-x r**) Page search for a string in the backward direction
**page-file-search-buffer**(3) Search the page file buffer
**page-file-search-forward**(3) (**C-x s**) Page search for a string in the forward direction
**paragraph-to-line**(3) Convert a paragraph to a single line
**password-to-phonic**(3) Convert a password to phonic string
**pipe-shell-command**(2) (**esc @**) Execute a single operating system command
**popup-window**(2) Pop-up a window on the screen
**prefix**(2) Key prefix command
**previous-window**(2) (**C-x p**) Move the cursor to the previous window
**print-buffer**(2) Print buffer, with formatting
**print-color**(2) Create a new printer color
**print-region**(2) Print region, with formatting
**print-scheme**(2) Create a new printer color and font scheme
**print-setup**(3) Configure MicroEmacs's printer interface
**query-replace-all-string**(3) Query replace string in a list of files
**query-replace-string**(2) (**esc C-r**) Search and replace a string - with query
**query-replace-string-in-files**(3) Conditionally search and replace strings across files
**quick-exit**(2) (**esc z**) Exit the editor writing changes
**quote-char**(2) (**C-q**) Insert literal character
**rcs-file**(2) (**C-x C-q**) Handle Revision Control System (RCS) files
**rdiff**(3) Recursively compare files or directories
**read-file**(2) (**C-x C-r**) Find and load file replacing current buffer
**read-registry**(2) Read in a registry definition file
**read-session**(3) Read in session information
**recenter**(2) (**C-l**) Recenter the window (refresh the screen)
**recompile**(3) Rerun compilation process
**replace-all-pairs**(3) Replace string pairs in a list of files
**replace-all-string**(3) Replace string with new string in a list of files
**replace-string**(2) (**esc r**) Replace string with new string
**replace-string-in-files**(3) Search and replace strings across files
**reread-all**(3) Reload all files
**reread-file**(3) Reload the current buffer's file
**resize-all-windows**(2) Resize all windows (automatic change)
**resize-window-horizontally**(2) Resize current window horizontally (absolute)
**resize-window-vertically**(2) Resize the current window (absolute change)
**restore-dictionary**(3) Save dictionary user changes
**restyle-buffer**(3) Automatically reformat a buffer's indentation
**restyle-region**(3) Automatically reformat a regions indentation
**reyank**(2) (**esc y**) Restore next yank buffer
**rgrep**(3) Execute recursive grep command
**save-all**(3) Save all modified files (with query)
**save-buffer**(2) (**C-x C-s**) Save contents of changed buffer to file
**save-buffers-exit-emacs**(2) (**esc z**) Exit the editor prompt user to write changes
**save-dictionary**(2) Save changed spelling dictionaries
**save-registry**(2) Write a registry definition file
**save-session**(3) Write session information to session file
**save-some-buffers**(2) Save contents of all changed buffers to file (with query)
**scheme-editor**(3) Color Scheme Editor
**screen-poke**(2) Immediate write string to the screen

**screen-update**(2) (**redraw**) Force screen update
**scroll-down**(2) (**C-n**) Move the window down (scrolling)
**scroll-left**(2) (**C-x <**) Move the window left (scrolling)
**scroll-next-window-down**(2) (**esc C-v**) Scroll next window down
**scroll-next-window-up**(2) (**esc C-z**) Scroll next window up
**scroll-right**(2) (**C-x >**) Move the window right (scrolling)
**scroll-up**(2) (**C-p**) Move the window up (scrolling)
**search-backward**(2) (**C-x r**) Search for a string in the backward direction
**search-buffer**(2) Search the current buffer for a string
**search-forward**(2) (**C-x s**) Search for a string in the forward direction
**session-setup**(3) Configure the data written to a session file
**set-alpha-mark**(2) (**C-x C-a**) Place an alphabetic marker in the buffer
**set-char-mask**(2) Set character word mask
**set-cursor-to-mouse**(2) Move the cursor to the current mouse position
**set-encryption-key**(2) (**esc e**) Define the encryption key
**set-mark**(2) (**esc space**) Set starting point of region
**set-position**(2) Store the current position
**set-registry**(2) Modify a node value in the registry
**set-scroll-with-mouse**(2) Scroll the window with the mouse
**set-variable**(2) (**C-x v**) Assign a new value to a variable
**set-window**(2) Save the current window for restore (historic)
**shell**(2) [**C-x c**] Create a new command processor or shell
**shell-command**(2) Perform an operating system command
**shell-tool-setup**(3) Define shell tools
**show-region**(2) Show the current copy region
**shrink-window-horizontally**(2) Shrink current window horizontally (relative)
**shrink-window-vertically**(2) Shrink the current window (relative change)
**shut-down**(3) Editor exit callback command
**sort-lines**(2) Alphabetically sort lines
**sort-lines-ignore-case**(3) Alphabetically sort lines ignoring case
**space-rectangle**(3) Adds space(s) vertically
**spell**(2) Spell checker service provider
**spell-add-word**(3) Add a word to the main dictionary
**spell-buffer**(3) Spell check the current buffer
**spell-edit-word**(3) Edits a spell word entry
**spell-word**(3) (**esc $**) Spell check a single word
**split-window-horizontally**(2) (**C-x 3**) Split current window into two (horizontally)
**split-window-vertically**(2) (**C-x 2**) Split the current window into two
**sql-format**(3) SQL language, format SQL query
**start-kbd-macro**(2) (**C-x (**) Start recording keyboard macro
**start-up**(3) Editor startup callback command
**stop-mail-check**(3) Disable the check for new email
**str-to-regex**(3) Generate regex safe variant of a string
**string-rectangle**(3) Add a string vertically
**suspend-emacs**(2) Suspend editor and place in background
**svn**(3) MicroEmacs Subversion interface
**svn-add**(3) MicroEmacs SVN interface - add items
**svn-blame**(3) MicroEmacs SVN interface - annotate a file
**svn-cat-file**(3) MicroEmacs SVN interface - output specific version of a file

**svn-checkout**(3) MicroEmacs SVN interface - checkout a working copy
**svn-commit**(3) MicroEmacs SVN interface - commit changes back
**svn-copy**(3) MicroEmacs SVN interface - copy items
**svn-delete**(3) MicroEmacs SVN interface - delete items
**svn-diff**(3) MicroEmacs SVN interface - diff changes
**svn-gdiff**(3) MicroEmacs SVN interface - graphical diff changes
**svn-ignore**(3) MicroEmacs SVN interface - edit directory item ignore list
**svn-info**(3) MicroEmacs SVN interface - display info about an item
**svn-log-commit**(3) MicroEmacs SVN interface - get log for a commit
**svn-log-file**(3) MicroEmacs SVN interface - list log of changes
**svn-move**(3) MicroEmacs SVN interface - move an item
**svn-resolve**(3) MicroEmacs SVN interface - resolve conflicts
**svn-revert**(3) MicroEmacs SVN interface - revert local changes
**svn-server-copy**(3) MicroEmacs SVN interface - server based copy items
**svn-server-move**(3) MicroEmacs SVN interface - server based move items
**svn-setup**(3) MicroEmacs SVN interface setup
**svn-status**(3) MicroEmacs SVN interface - status of items
**svn-update**(3) MicroEmacs SVN interface - update directory files
**svn-xdiff**(3) MicroEmacs SVN interface - External diff changes
**Triangle**(3) MicroEmacs version of Triangle patience game
**tab**(2) (**tab**) Handle the tab key
**tabs-to-spaces**(3) Converts all tabs to spaces
**tex2nr**(3) Convert a Latex file into nroff
**time**(3) Command time evaluator
**toolbar**(3) Toggle tool bar visibility
**toolbar-close**(3) Close the tool bar
**toolbar-open**(3) Open the tool bar
**toolbar-refresh**(3) Update the tool bar
**translate-buffer**(2) Translate buffer from one format to another
**translate-key**(2) Translate key
**transpose-chars**(2) (**C-t**) Exchange (swap) adjacent characters
**transpose-lines**(2) (**C-x C-t**) Exchange (swap) adjacent lines
**transpose-region-line**(3) Exchange (swap) region with adjacent line
**undo**(2) (**C-x u**) Undo the last edit
**uniq**(3) Make lines in a sorted list unique
**universal-argument**(2) (**C-u**) Set the command argument count
**unmark-buffer**(3) Remove buffer edited flag
**unset-variable**(2) Delete a variable
**upper-case-region**(2) (**C-x C-u**) Uppercase a region (upcase)
**upper-case-word**(2) (**esc u**) Uppercase word (upcase)
**user-setup**(3) Configure MicroEmacs for a specific user
**vcs**(3) MicroEmacs VCS interface - VCS-agnostic front-end
**vcs-add**(3) MicroEmacs VCS interface - add items
**vcs-annotate**(3) MicroEmacs VCS interface - annotate a file
**vcs-blame**(3) MicroEmacs VCS interface - annotate a file
**vcs-cat-file**(3) MicroEmacs VCS interface - output a specific revision/commit of a file
**vcs-checkout**(3) MicroEmacs VCS interface - checkout (create) a working copy
**vcs-coci**(3) MicroEmacs VCS interface - check-out/check-in a file
**vcs-commit**(3) MicroEmacs VCS interface - commit changes to repository

**vcs-delete**(3) MicroEmacs VCS interface - delete/remove items
**vcs-diff**(3) MicroEmacs VCS interface - diff changes
**vcs-fetch**(3) MicroEmacs VCS interface - download from another repository
**vcs-gdiff**(3) MicroEmacs VCS interface - graphical diff changes
**vcs-id**(3) MicroEmacs VCS interface - identify the VCS controlling a file/directory
**vcs-log-commit**(3) MicroEmacs VCS interface - show log for a commit
**vcs-log-file**(3) MicroEmacs VCS interface - list log of changes
**vcs-move**(3) MicroEmacs VCS interface - move an item
**vcs-pull**(3) MicroEmacs VCS interface - download and integrate another repository
**vcs-push**(3) MicroEmacs VCS interface - update remote repository
**vcs-remove**(3) MicroEmacs VCS interface - remove items
**vcs-resolve**(3) MicroEmacs VCS interface - resolve conflicts
**vcs-revert**(3) MicroEmacs VCS interface - revert local changes
**vcs-setup**(3) MicroEmacs VCS interface - setup the underlying VCS
**vcs-status**(3) MicroEmacs VCS interface - status of items
**vcs-switch**(3) MicroEmacs VCS interface - switch or create branches
**vcs-unstage**(3) MicroEmacs VCS interface - unstage items
**vcs-update**(3) MicroEmacs VCS interface - update directory files
**view-file**(2) (**C-x C-v**) Load a file read only
**vm**(3) Email viewer
**void**(2) Null command
**watch**(3) Monitor a changing file
**watch-reset**(3) Reset the watch command counters
**watch-time**(3) Change the watch command poll time
**which**(3) Program finder
**wrap-word**(2) Wrap word onto next line
**write-buffer**(2) (**C-x C-w**) Write contents of buffer to named (new) file
**write-region**(3) Write contents of region to named (new) file
**xdiff**(3) External graphical file difference
**xfind**(3) Execute find command
**xgrep**(3) Execute grep command
**xrdb-scheme**(3) Load a XRDB scheme file from the user's config folder
**yank**(2) (**C-y**) Paste (copy) kill buffer contents into buffer
**yank-rectangle**(2) (**esc C-y**) Insert a column of text
**yank-rectangle-overwrite**(3) Insert a column of text with overwrite
**zfile-reset**(3) Compressed file support reset
**zfile-setup**(3) Compressed file support setup

## SPLIT COMMAND GLOSSARY

The following is a list of all of the built in commands provided by **MicroEmacs** [See mixed listing]:

**abort-command**(2) (**C-g**) Abort command
**about**(2) Information About MicroEmacs
**add-color**(2) Create a new color
**add-color-scheme**(2) Create a new color scheme
**add-dictionary**(2) Declare existence of a spelling dictionary
**add-file-hook**(2) DEPRICATED: Declare file name context dependent configuration
**add-next-line**(2) Define the searching behavior of command output
**add-spell-rule**(2) Add a new spelling rule to the dictionary
**append-buffer**(2) Write contents of buffer to end of named file
**backward-char**(2) (**C-b**) Move the cursor left
**backward-delete-char**(2) (**backspace**) Delete the previous character at the cursor position
**backward-delete-tab**(2) (**S-tab**) Delete white space to previous tab-stop
**backward-kill-word**(2) (**esc backspace**) Delete the previous word at the cursor position
**backward-line**(2) (**C-p**) Move the cursor to the previous line
**backward-paragraph**(2) (**esc p**) Move the cursor to the previous paragraph
**backward-word**(2) (**esc b**) Move the cursor to the previous word
**beginning-of-buffer**(2) (**esc <**) Move to beginning of buffer/file
**beginning-of-line**(2) (**C-a**) Move to beginning of line
**buffer-abbrev-file**(2) Set buffers' abbreviation file
**buffer-bind-key**(2) Create local key binding for current buffer
**buffer-info**(2) (**C-x =**) Status information on current buffer position
**buffer-mode**(2) (**C-x m**) Change a local buffer mode
**buffer-unbind-key**(2) Remove local key binding for current buffer
**capitalize-word**(2) (**esc c**) Capitalize word
**change-buffer-name**(2) (**esc C-n**) Change name of current buffer
**change-file-name**(2) (**C-x n**) Change the file name of the current buffer
**change-font**(2) Change the screen font
**change-frame-depth**(2) Change the number of lines on the current frame
**change-frame-width**(2) Change the number of columns on the current frame
**change-window-depth**(2) Change the depth of the current window
**change-window-width**(2) Change the width of the current window
**command-apropos**(2) (**C-h a**) List commands involving a concept
**command-wait**(2) Conditional wait command
**compare-windows**(2) Compare buffer windows, ignore whitespace
**copy-region**(2) (**esc w**) Copy a region of the buffer
**count-words**(2) (**esc C-c**) Count the number of words in a region
**create-callback**(2) Create a timer callback
**create-frame**(2) (**C-x 5 1**, **C-x 5 2**) Create a new frame
**define-help**(2) Define help information
**define-macro**(2) Define a new macro
**define-macro-file**(2) Define macro file location
**delete-blank-lines**(2) (**C-x C-o**) Delete blank lines about cursor
**delete-buffer**(2) (**C-x k**) Delete a buffer
**delete-dictionary**(2) Remove a spelling dictionary from memory
**delete-frame**(2) (**C-x 5 0**) Delete the current frame

**delete-other-windows**(2) (**C-x 1**) Delete other windows
**delete-registry**(2) Delete a registry tree
**delete-some-buffers**(2) Delete buffers with query
**delete-window**(2) (**C-x 0**) Delete current window
**describe-bindings**(2) (**C-h b**) Show current command/key binding
**describe-key**(2) (**C-x ?**) Report keyboard key name and binding
**describe-variable**(2) (**C-h v**) Describe current setting of a variable
**directory-tree**(2) Draw the file directory tree
**end-kbd-macro**(2) (**C-x )**) Stop recording keyboard macro
**end-of-buffer**(2) (**esc >**) Move to end of buffer/file
**end-of-line**(2) (**C-e**) Move to end of line
**exchange-point-and-mark**(2) (**C-x C-x**) Exchange the cursor and marked position
**execute-buffer**(2) Execute script lines from a buffer
**execute-file**(2) (**esc /**) Execute script lines from a file
**execute-kbd-macro**(2) (**C-x e**) Execute a keyboard macro
**execute-line**(2) Execute a typed in script line
**execute-named-command**(2) [**esc x**] Execute a named command
**execute-string**(2) Execute a string as a command
**exit-emacs**(2) Exit MicroEmacs
**expand-abbrev**(2) Expand an abbreviation
**file-op**(2) File system operations command
**fill-paragraph**(2) (**esc o**) Format a paragraph
**filter-buffer**(2) (**C-x #**) Filter the current buffer through an O/S command
**find-buffer**(2) (**C-x b**) Switch to a named buffer
**find-file**(2) (**C-x C-f**) Load a file
**find-registry**(2) Index search of a registry sub-tree
**find-tag**(2) (**esc t**) Find tag, auto-load file and move to tag position
**forward-char**(2) (**C-f**) Move the cursor right
**forward-delete-char**(2) (**C-d**) Delete the next character at the cursor position
**forward-kill-word**(2) (**esc d**) Delete the next word at the cursor position
**forward-line**(2) (**C-n**) Move the cursor to the next line
**forward-paragraph**(2) (**esc n**) Move the cursor to the next paragraph
**forward-word**(2) (**esc f**) Move the cursor to the next word
**generate-hash**(2) Generate SHA1 or SHA256 hash or HMAC
**get-next-line**(2) (**C-x ‘**) Find the next command line
**global-abbrev-file**(2) Set global abbreviation file
**global-bind-key**(2) (**esc k**) Bind a key to a named command or macro
**global-mode**(2) (**esc m**) Change a global buffer mode
**global-unbind-key**(2) (**esc C-k**) Unbind a key from a named command or macro
**goto-alpha-mark**(2) (**C-x a**) Move the cursor to a alpha marked location
**goto-line**(2) (**esc g**) Move the cursor to specified line
**goto-matching-fence**(2) (**esc C-f**) Move the cursor to matching fence
**goto-position**(2) Restore a stored position
**goto-window**(2) Restore a saved window to the current window (historic)
**grow-window-horizontally**(2) Enlarge current window horizontally (relative)
**grow-window-vertically**(2) Enlarge the current window (relative change)
**help**(2) (**esc ?**) Help; high level introduction to help
**help-command**(2) (**C-h C-c**) Help; command information
**help-item**(2) (**C-h C-i**) Help; item information

**help-variable**(2) (**C-h C-v**) Help; variable information
**hilight**(2) Manage the buffer hilighting schemes
**hunt-backward**(2) (**C-x C-h**) Resume previous search in backward direction
**hunt-forward**(2) (**C-x h**) Resume previous search in forward direction
**indent**(2) Manage the auto-indentation methods
**insert-file**(2) (**C-x C-i**) Insert file into current buffer
**insert-file-name**(2) (**C-x C-y**) Insert filename into current buffer
**insert-macro**(2) Insert keyboard macro into buffer
**insert-newline**(2) (**C-o**) Insert new line at cursor position
**insert-space**(2) Insert space(s) into current buffer
**insert-string**(2) Insert character string into current buffer
**insert-tab**(2) (**C-i**) Insert tab(s) into current buffer
**ipipe-kill**(2) Kill a incremental pipe
**ipipe-shell-command**(2) (**esc backslash**) Incremental pipe (non-suspending system call)
**ipipe-write**(2) Write a string to an incremental pipe
**isearch-backward**(2) (**C-r**) Search backwards incrementally (interactive)
**isearch-forward**(2) (**C-s**) Search forward incrementally (interactive)
**kbd-macro-query**(2) (**C-x q**) Query termination of keyboard macro
**kill-line**(2) (**C-k**) Delete all characters to the end of the line
**kill-paragraph**(2) Delete a paragraph
**kill-rectangle**(2) (**esc C-w**) Delete a column of text
**kill-region**(2) (**C-w**) Delete all characters in the marked region
**list-buffers**(2) (**C-x C-b**) List all buffers and show their status
**list-commands**(2) (**C-h c**) List available commands
**list-registry**(2) Display the registry in a buffer
**list-variables**(2) (**C-h v**) List defined variables
**lower-case-region**(2) (**C-x C-l**) Lowercase a region (downcase)
**lower-case-word**(2) (**esc l**) Lowercase word (downcase)
**major-mode**(2) Declare or set a buffer major mode
**mark-registry**(2) Modify the operating mode of a registry node
**ml-bind-key**(2) Create key binding for message line
**ml-unbind-key**(2) Remove key binding from message line
**ml-write**(2) Write message on message line
**name-kbd-macro**(2) Assign a name to the last keyboard macro
**named-buffer-mode**(2) Change a named buffer mode
**narrow-buffer**(2) Hide buffer lines
**newline**(2) (**return**) Insert a new line
**next-buffer**(2) (**C-x x**) Switch to the next buffer
**next-frame**(2) (**C-x 5 o**) Change the focus to the next frame
**next-window**(2) (**C-x o**) Move the cursor to the next window
**next-window-find-buffer**(2) [] Split the current window and show new buffer
**next-window-find-file**(2) (**C-x 4**) Split the current window and find file
**osd**(2) Manage the On-Screen Display
**osd-bind-key**(2) Create key binding for OSD dialog
**osd-unbind-key**(2) Remove key binding from OSD dialog
**pipe-shell-command**(2) (**esc @**) Execute a single operating system command
**popup-window**(2) Pop-up a window on the screen
**prefix**(2) Key prefix command
**previous-window**(2) (**C-x p**) Move the cursor to the previous window

**print-buffer**(2) Print buffer, with formatting
**print-color**(2) Create a new printer color
**print-region**(2) Print region, with formatting
**print-scheme**(2) Create a new printer color and font scheme
**query-replace-string**(2) (**esc C-r**) Search and replace a string - with query
**quick-exit**(2) (**esc z**) Exit the editor writing changes
**quote-char**(2) (**C-q**) Insert literal character
**rcs-file**(2) (**C-x C-q**) Handle Revision Control System (RCS) files
**read-file**(2) (**C-x C-r**) Find and load file replacing current buffer
**read-registry**(2) Read in a registry definition file
**recenter**(2) (**C-l**) Recenter the window (refresh the screen)
**replace-string**(2) (**esc r**) Replace string with new string
**resize-all-windows**(2) Resize all windows (automatic change)
**resize-window-horizontally**(2) Resize current window horizontally (absolute)
**resize-window-vertically**(2) Resize the current window (absolute change)
**reyank**(2) (**esc y**) Restore next yank buffer
**save-buffer**(2) (**C-x C-s**) Save contents of changed buffer to file
**save-buffers-exit-emacs**(2) (**esc z**) Exit the editor prompt user to write changes
**save-dictionary**(2) Save changed spelling dictionaries
**save-registry**(2) Write a registry definition file
**save-some-buffers**(2) Save contents of all changed buffers to file (with query)
**screen-poke**(2) Immediate write string to the screen
**screen-update**(2) (**redraw**) Force screen update
**scroll-down**(2) (**C-n**) Move the window down (scrolling)
**scroll-left**(2) (**C-x <**) Move the window left (scrolling)
**scroll-next-window-down**(2) (**esc C-v**) Scroll next window down
**scroll-next-window-up**(2) (**esc C-z**) Scroll next window up
**scroll-right**(2) (**C-x >**) Move the window right (scrolling)
**scroll-up**(2) (**C-p**) Move the window up (scrolling)
**search-backward**(2) (**C-x r**) Search for a string in the backward direction
**search-buffer**(2) Search the current buffer for a string
**search-forward**(2) (**C-x s**) Search for a string in the forward direction
**set-alpha-mark**(2) (**C-x C-a**) Place an alphabetic marker in the buffer
**set-char-mask**(2) Set character word mask
**set-cursor-to-mouse**(2) Move the cursor to the current mouse position
**set-encryption-key**(2) (**esc e**) Define the encryption key
**set-mark**(2) (**esc space**) Set starting point of region
**set-position**(2) Store the current position
**set-registry**(2) Modify a node value in the registry
**set-scroll-with-mouse**(2) Scroll the window with the mouse
**set-variable**(2) (**C-x v**) Assign a new value to a variable
**set-window**(2) Save the current window for restore (historic)
**shell**(2) [**C-x c**] Create a new command processor or shell
**shell-command**(2) Perform an operating system command
**show-region**(2) Show the current copy region
**shrink-window-horizontally**(2) Shrink current window horizontally (relative)
**shrink-window-vertically**(2) Shrink the current window (relative change)
**sort-lines**(2) Alphabetically sort lines
**spell**(2) Spell checker service provider

**split-window-horizontally**(2) (**C-x 3**) Split current window into two (horizontally)
**split-window-vertically**(2) (**C-x 2**) Split the current window into two
**start-kbd-macro**(2) (**C-x (**) Start recording keyboard macro
**suspend-emacs**(2) Suspend editor and place in background
**tab**(2) (**tab**) Handle the tab key
**translate-buffer**(2) Translate buffer from one format to another
**translate-key**(2) Translate key
**transpose-chars**(2) (**C-t**) Exchange (swap) adjacent characters
**transpose-lines**(2) (**C-x C-t**) Exchange (swap) adjacent lines
**undo**(2) (**C-x u**) Undo the last edit
**universal-argument**(2) (**C-u**) Set the command argument count
**unset-variable**(2) Delete a variable
**upper-case-region**(2) (**C-x C-u**) Uppercase a region (upcase)
**upper-case-word**(2) (**esc u**) Uppercase word (upcase)
**view-file**(2) (**C-x C-v**) Load a file read only
**void**(2) Null command
**wrap-word**(2) Wrap word onto next line
**write-buffer**(2) (**C-x C-w**) Write contents of buffer to named (new) file
**yank**(2) (**C-y**) Paste (copy) kill buffer contents into buffer
**yank-rectangle**(2) (**esc C-y**) Insert a column of text

The following is a list of documented macro commands provided by **MicroEmacs**:

**Arkanoid**(3) MicroEmacs version of the bat and ball wall game
**abbrev-list**(3) Create a list of abbreviations
**add-global-mode**(3) Set a global buffer mode
**add-mode**(3) Set a local buffer mode
**alarm**(3) Set an alarm
**alias-path-setup**(3) Define alternative path alias
**aman**(3) Compile an nroff file into a buffer (UNIX)
**ascii-time**(3) Return the current time as a string
**auto-spell**(3) Auto-spell support
**auto-spell-buffer**(3) Auto-spell whole buffer
**auto-spell-ignore**(3) Auto-spell ignore current word
**auto-spell-menu**(3) Auto-spell context menu
**auto-spell-reset**(3) Auto-spell hilight reset
**Balls**(3) MicroEmacs bouncing ball playground
**bin-copy-region**(3) Binary supporting copy region
**bin-yank**(3) Binary supporting kill buffer insert
**browse**(3) Browse a file or buffer in view mode
**buffer-init**(3) Initialize a buffer language template
**buffer-init-fhook**(3) Initialize the buffer file hooks
**buffer-init-hooks**(3) Initialize the buffer language template hooks
**buffer-is-utf8**(3) Detect if the current buffer is UTF-8 encoded
**buffer-major-mode**(3) Change the operating mode of a buffer
**buffer-setup**(3) Configure the current buffer settings
**ClearCase**(3) MicroEmacs ClearCase interface
**c-deslashify**(3) C language, remove backslash to end of lines of region
**c-hash-define**(3) Set a C/C++ #define
**c-hash-eval**(3) Evaluate C/C++ #defines

**c-hash-undef**(3) Unset a C/C++ #define
**c-hash-undo**(3) Remove C/C++ #define evaluation
**c-slashify**(3) C language, append backslash to end of lines of region
**c-to-cpp-convert**(3) Convert a C file to C++
**calc**(3) Calculator
**calc-op**(3) Low level calculator command
**cc-add**(3) MicroEmacs ClearCase interface - add file
**cc-checkin**(3) MicroEmacs ClearCase interface - checkin files
**cc-checkout**(3) MicroEmacs ClearCase interface - checkout files
**cc-delete**(3) MicroEmacs ClearCase interface - delete files
**cc-diff**(3) MicroEmacs ClearCase interface - diff changes
**cc-setup**(3) MicroEmacs ClearCase interface setup
**cc-state**(3) MicroEmacs ClearCase interface - list state of directory files
**cc-update**(3) MicroEmacs ClearCase interface - update directory files
**change-buffer-charset**(3) Convert buffer between two character sets
**change-company**(3) Change current company profile
**change-font-size**(3) Incrementally change font size
**change-scheme**(3) Change the current highlight scheme
**check-line-length**(3) Check the length of text lines are valid
**clean**(3) Remove redundant white spaces from the current buffer
**collapse-all**(3) (**f3**) (Un)Fold all regions in the current buffer
**collapse-current**(3) (**f2**) (un)Fold a region in the current buffer
**compare-windows-exact**(3) Compare buffer windows, with whitespace
**compile**(3) Start a compilation process
**copy-file**(3) Copy or move files and directories
**copy-rectangle**(3) Copy a rectangular region of text
**copy-registry**(3) Recursive copy of registry subtree
**cvs**(3) MicroEmacs CVS interface
**cvs-add**(3) MicroEmacs CVS interface - add file
**cvs-annotate**(3) MicroEmacs CVS interface - annotate file
**cvs-cat-file**(3) MicroEmacs CVS interface - output specific revision of a file
**cvs-checkout**(3) MicroEmacs CVS interface - checkout files
**cvs-commit**(3) MicroEmacs CVS interface - commit changes
**cvs-diff**(3) MicroEmacs CVS interface - diff changes
**cvs-file-status**(3) MicroEmacs CVS interface - status of files
**cvs-gdiff**(3) MicroEmacs CVS interface - graphical diff changes
**cvs-log-file**(3) MicroEmacs CVS interface - log changes
**cvs-remove**(3) MicroEmacs CVS interface - remove file
**cvs-resolve**(3) MicroEmacs CVS interface - resolve conflicts
**cvs-resolve-conflicts**(3) MicroEmacs CVS interface - resolve conflicts
**cvs-setup**(3) MicroEmacs CVS interface setup
**cvs-status**(3) MicroEmacs CVS interface - list state of directory files
**cvs-tag**(3) MicroEmacs CVS interface - tag files
**cvs-update**(3) MicroEmacs CVS interface - update directory files
**cygwin**(3) Open a Cygwin BASH window
**delete-file**(3) Delete files and directories
**delete-file-delayed**(3) Delete file after a specified delay
**delete-global-mode**(3) Remove a global buffer mode
**delete-hilight-lines**(3) Delete hilight lines

**delete-indentation**(3) Join 2 lines deleting white spaces
**delete-mode**(3) Remove a local buffer mode
**describe-word**(3) Display a dictionary definition of a word
**diff**(3) Difference files or directories
**diff-changes**(3) Find the differences from a previous edit session
**diff-lines**(3) Highlight the differences between two lines
**display-white-chars**(3) Toggle the displaying of white characters
**display-whitespaces**(3) Toggle the displaying of white characters
**draw**(3) Simple line drawing utility
**edit-dictionary**(3) Insert a dictionary in a buffer
**execute-tool**(3) Execute a user defined shell tool
**expand-abbrev-handle**(3) (**esc esc**) Expand an abbreviation handler
**expand-look-back**(3) Complete a word by looking back for a similar word
**expand-word**(3) Complete a word by invocation of the speller
**file-attrib**(3) Set the current buffers system file attributes
**file-browser**(3) (**f10**) Browse the file system
**file-browser-close**(3) Close the file-browser
**file-browser-swap-buffers**(3) Swap between file-browser windows
**file-exec**(3) Executes the current buffer with compiler/interpreter
**file-format**(3) Format the current buffer using the a formatter
**file-lint**(3) check the current buffer for problems using a linter
**file-list**(3) Generate list of matching file names
**file-tool-setup**(3) Define file browser tools
**file-type-setup**(3) Define file type set
**filemask-to-regex**(3) Convert a file mask to a regex
**find**(3) Execute built-in find command
**find-bfile**(3) (**C-x 9**) Load a file as binary data
**find-cfile**(3) Load an encrypted file
**find-hfile**(3) (**C-x C-h**) Load a file using selected handler
**find-lfile**(3) Load a file lossless
**find-rfile**(3) Load a file as binary data (reduced)
**find-setup**(3) Setup the built-in find and grep commands
**find-ufile**(3) Load a Unicode file
**find-word**(3) Find a word using spelling dictionaries
**find-zfile**(3) Compressed file support
**fold-close**(3) (**C-f2**) Close an open fold (in-line)
**fold-create**(3) (**C-f2**) Create a new fold
**fold-enter**(3) (**f3**) Enter fold
**fold-exit**(3) (**esc f3**) Exit a fold
**fold-exit-all**(3) (**esc C-f3**) Exit all folds
**fold-kill**(3) (**C-x C-k**) Delete the fold and all its contents
**fold-open**(3) (**f2**) Open fold (in-line)
**fold-remove**(3) (**esc C-f3**) Remove the fold, maintaining content
**ftp**(3) Initiate an FTP connection
**gdiff**(3) Graphical file difference
**generate-password**(3) Generate a password with given criteria
**generate-tags-file**(3) Generate a tags file
**git**(3) MicroEmacs GIT interface
**git-add**(3) MicroEmacs GIT interface - add items to the index

**git-blame**(3) MicroEmacs GIT interface - annotate a file
**git-cat-file**(3) MicroEmacs GIT interface - output specific commit of a file
**git-commit**(3) MicroEmacs GIT interface - commit changes to repository
**git-diff**(3) MicroEmacs GIT interface - diff changes
**git-fetch**(3) MicroEmacs GIT interface - download from another repository
**git-gdiff**(3) MicroEmacs GIT interface - graphical diff changes
**git-log-commit**(3) MicroEmacs GIT interface - show information about a specific commit
**git-log-file**(3) MicroEmacs GIT interface - list log of changes
**git-move**(3) MicroEmacs GIT interface - move an item
**git-pull**(3) MicroEmacs GIT interface - download and integrate another repository
**git-push**(3) MicroEmacs GIT interface - update remote repository
**git-remove**(3) MicroEmacs GIT interface - remove items
**git-revert**(3) MicroEmacs GIT interface - revert local changes
**git-setup**(3) MicroEmacs GIT interface setup
**git-status**(3) MicroEmacs GIT interface - status of items
**git-switch**(3) MicroEmacs GIT interface - switch or create branches
**git-unstage**(3) MicroEmacs GIT interface - unstage items
**grep**(3) Search a file for a pattern
**grep-for-region**(3) Grep current directory for current region
**Hearts**(3) MicroEmacs version of the card game Hearts
**ifill-paragraph**(3) (**esc q**) Format a paragraph
**indent-decrease**(3) Decrease the line indentation
**indent-increase**(3) Increase the line indentation
**info**(3) Display a GNU Info database
**info-goto-link**(3) Display Info on a given link
**info-on**(3) Display Info on a given topic
**insert-calc**(3) Calculator with answer insertion
**insert-password**(3) Create and insert a new password string
**insert-symbol**(3) Insert an ASCII character
**insert-template**(3) Insert template file into current buffer
**ishell**(3) Open an interactive shell window
**keep-buffer**(3) Rename the current buffer
**line-scheme**(3) Apply line annotation in current buffer
**Mahjongg**(3) MicroEmacs version of the solitaire Mah Jongg game
**MainMenu**(3) The top main menu
**Match-It**(3) MicroEmacs version of the Match-It game
**Metris**(3) MicroEmacs version of the falling blocks game
**mail**(3) Compose and send an email
**mail-check**(3) Check for new email
**mail-setup**(3) Email setup dialog
**major-mode-help**(3) Displays help page for current buffer
**major-mode-setup**(3) Configures the current buffer settings
**major-mode-tools-setup**(3) File major-mode tool setup
**man**(3) UNIX manual page viewer
**man-clean**(3) Clean UNIX manual page
**narrow-search-backward**(3) (**C-c r**) Search for a string in the backward direction
**narrow-search-forward**(3) (**C-c s**) Search for a string in the forward direction
**normal-tab**(3) Insert a normal tab
**notes**(3) Tagged hierarchical database of notes

**notes-close**(3) Close the notes window
**notes-context**(3) Create/Access buffer context note
**notes-file**(3) Open a notes database file
**notes-import-skwyrul**(3) Import a Skwyrul index file (skwyrul_l.htm)
**notes-instant**(3) Create quick note
**open-line**(3) (**C-o**) Open new line at cursor position
**organizer**(3) Calendar and address organizer
**osd-dialog**(3) OSD dialog box
**osd-entry**(3) OSD entry dialog box
**osd-help**(3) GUI based on-line help
**osd-search**(3) Search and annotate the current buffer
**osd-xdialog**(3) OSD Extended dialog box
**Patience**(3) MicroEmacs version of Patience (or Solitaire)
**page-bfile**(3) Load a large binary file as binary data
**page-file**(3) Load a large file in page mode
**page-file-backward-line**(3) (**C-p**) Move backwards one line in paged buffer
**page-file-beginning-of-buffer**(3) Move to the start of the paged file
**page-file-end-of-buffer**(3) Move to the end of the paged file
**page-file-forward-line**(3) (**C-n**) Move forwards one line in paged buffer
**page-file-goto-byte**(3) Page to the buffer byte offset
**page-file-goto-page**(3) Move to page
**page-file-isearch-backward**(3) (**C-r**) Page search backwards incrementally (interactive)
**page-file-isearch-forward**(3) (**C-s**) Page search forward incrementally (interactive)
**page-file-save-buffer**(3) (**C-x C-s**) Save contents of a page mode buffer to file
**page-file-scroll-down**(3) (**C-n**) Scroll paged buffer down
**page-file-scroll-up**(3) (**C-p**) Scroll paged buffer up
**page-file-search-backward**(3) (**C-x r**) Page search for a string in the backward direction
**page-file-search-buffer**(3) Search the page file buffer
**page-file-search-forward**(3) (**C-x s**) Page search for a string in the forward direction
**paragraph-to-line**(3) Convert a paragraph to a single line
**password-to-phonic**(3) Convert a password to phonic string
**print-setup**(3) Configure MicroEmacs's printer interface
**query-replace-all-string**(3) Query replace string in a list of files
**query-replace-string-in-files**(3) Conditionally search and replace strings across files
**rdiff**(3) Recursively compare files or directories
**read-session**(3) Read in session information
**recompile**(3) Rerun compilation process
**replace-all-pairs**(3) Replace string pairs in a list of files
**replace-all-string**(3) Replace string with new string in a list of files
**replace-string-in-files**(3) Search and replace strings across files
**reread-all**(3) Reload all files
**reread-file**(3) Reload the current buffer's file
**restore-dictionary**(3) Save dictionary user changes
**restyle-buffer**(3) Automatically reformat a buffer's indentation
**restyle-region**(3) Automatically reformat a regions indentation
**rgrep**(3) Execute recursive grep command
**save-all**(3) Save all modified files (with query)
**save-session**(3) Write session information to session file
**scheme-editor**(3) Color Scheme Editor

**session-setup**(3) Configure the data written to a session file
**shell-tool-setup**(3) Define shell tools
**shut-down**(3) Editor exit callback command
**sort-lines-ignore-case**(3) Alphabetically sort lines ignoring case
**space-rectangle**(3) Adds space(s) vertically
**spell-add-word**(3) Add a word to the main dictionary
**spell-buffer**(3) Spell check the current buffer
**spell-edit-word**(3) Edits a spell word entry
**spell-word**(3) (**esc $**) Spell check a single word
**sql-format**(3) SQL language, format SQL query
**start-up**(3) Editor startup callback command
**stop-mail-check**(3) Disable the check for new email
**str-to-regex**(3) Generate regex safe variant of a string
**string-rectangle**(3) Add a string vertically
**svn**(3) MicroEmacs Subversion interface
**svn-add**(3) MicroEmacs SVN interface - add items
**svn-blame**(3) MicroEmacs SVN interface - annotate a file
**svn-cat-file**(3) MicroEmacs SVN interface - output specific version of a file
**svn-checkout**(3) MicroEmacs SVN interface - checkout a working copy
**svn-commit**(3) MicroEmacs SVN interface - commit changes back
**svn-copy**(3) MicroEmacs SVN interface - copy items
**svn-delete**(3) MicroEmacs SVN interface - delete items
**svn-diff**(3) MicroEmacs SVN interface - diff changes
**svn-gdiff**(3) MicroEmacs SVN interface - graphical diff changes
**svn-ignore**(3) MicroEmacs SVN interface - edit directory item ignore list
**svn-info**(3) MicroEmacs SVN interface - display info about an item
**svn-log-commit**(3) MicroEmacs SVN interface - get log for a commit
**svn-log-file**(3) MicroEmacs SVN interface - list log of changes
**svn-move**(3) MicroEmacs SVN interface - move an item
**svn-resolve**(3) MicroEmacs SVN interface - resolve conflicts
**svn-revert**(3) MicroEmacs SVN interface - revert local changes
**svn-server-copy**(3) MicroEmacs SVN interface - server based copy items
**svn-server-move**(3) MicroEmacs SVN interface - server based move items
**svn-setup**(3) MicroEmacs SVN interface setup
**svn-status**(3) MicroEmacs SVN interface - status of items
**svn-update**(3) MicroEmacs SVN interface - update directory files
**svn-xdiff**(3) MicroEmacs SVN interface - External diff changes
**Triangle**(3) MicroEmacs version of Triangle patience game
**tabs-to-spaces**(3) Converts all tabs to spaces
**tex2nr**(3) Convert a Latex file into nroff
**time**(3) Command time evaluator
**toolbar**(3) Toggle tool bar visibility
**toolbar-close**(3) Close the tool bar
**toolbar-open**(3) Open the tool bar
**toolbar-refresh**(3) Update the tool bar
**transpose-region-line**(3) Exchange (swap) region with adjacent line
**uniq**(3) Make lines in a sorted list unique
**unmark-buffer**(3) Remove buffer edited flag
**user-setup**(3) Configure MicroEmacs for a specific user

**vcs**(3) MicroEmacs VCS interface - VCS-agnostic front-end
**vcs-add**(3) MicroEmacs VCS interface - add items
**vcs-annotate**(3) MicroEmacs VCS interface - annotate a file
**vcs-blame**(3) MicroEmacs VCS interface - annotate a file
**vcs-cat-file**(3) MicroEmacs VCS interface - output a specific revision/commit of a file
**vcs-checkout**(3) MicroEmacs VCS interface - checkout (create) a working copy
**vcs-coci**(3) MicroEmacs VCS interface - check-out/check-in a file
**vcs-commit**(3) MicroEmacs VCS interface - commit changes to repository
**vcs-delete**(3) MicroEmacs VCS interface - delete/remove items
**vcs-diff**(3) MicroEmacs VCS interface - diff changes
**vcs-fetch**(3) MicroEmacs VCS interface - download from another repository
**vcs-gdiff**(3) MicroEmacs VCS interface - graphical diff changes
**vcs-id**(3) MicroEmacs VCS interface - identify the VCS controlling a file/directory
**vcs-log-commit**(3) MicroEmacs VCS interface - show log for a commit
**vcs-log-file**(3) MicroEmacs VCS interface - list log of changes
**vcs-move**(3) MicroEmacs VCS interface - move an item
**vcs-pull**(3) MicroEmacs VCS interface - download and integrate another repository
**vcs-push**(3) MicroEmacs VCS interface - update remote repository
**vcs-remove**(3) MicroEmacs VCS interface - remove items
**vcs-resolve**(3) MicroEmacs VCS interface - resolve conflicts
**vcs-revert**(3) MicroEmacs VCS interface - revert local changes
**vcs-setup**(3) MicroEmacs VCS interface - setup the underlying VCS
**vcs-status**(3) MicroEmacs VCS interface - status of items
**vcs-switch**(3) MicroEmacs VCS interface - switch or create branches
**vcs-unstage**(3) MicroEmacs VCS interface - unstage items
**vcs-update**(3) MicroEmacs VCS interface - update directory files
**vm**(3) Email viewer
**watch**(3) Monitor a changing file
**watch-reset**(3) Reset the watch command counters
**watch-time**(3) Change the watch command poll time
**which**(3) Program finder
**write-region**(3) Write contents of region to named (new) file
**xdiff**(3) External graphical file difference
**xfind**(3) Execute find command
**xgrep**(3) Execute grep command
**xrdb-scheme**(3) Load a XRDB scheme file from the user's config folder
**yank-rectangle-overwrite**(3) Insert a column of text with overwrite
**zfile-reset**(3) Compressed file support reset
**zfile-setup**(3) Compressed file support setup

## VARIABLE GLOSSARY

The following is an alphabetic list of **MicroEmacs** variables:

**$INFOPATH**(5) GNU info files base directory
**$LOGNAME**(5) System user name (UNIX)
**$MEBACKUPPATH**(5) Backup file location
**$MEBACKUPSUB**(5) Backup file name modifier
**$MEINSTALLPATH**(5) MicroEmacs system macros search path
**$MENAME**(5) MicroEmacs user name
**$MEPATH**(5) MicroEmacs search path
**$MEUSERPATH**(5) MicroEmacs user directory
**$ME_PIPE_STDERR**(5) Command line diversion to stderr symbol
**$auto-time**(5) Automatic buffer save time
**$box-chars**(5) Characters used to draw lines
**$buffer-backup**(5) Buffer backup file name
**$buffer-bhook**(5) Buffer macro hook command name (buffer current)
**$buffer-bname**(5) Name of the current buffer
**$buffer-dhook**(5) Buffer macro hook command name (buffer deletion)
**$buffer-ehook**(5) Buffer macro hook command name (buffer swapped)
**$buffer-fhook**(5) Buffer macro hook command name (buffer creation)
**$buffer-fill-col**(5) Buffer paragraph mode; right fill column
**$buffer-fill-mode**(5) Buffer paragraph mode; justification method
**$buffer-fmod**(5) Buffer file modes (or attributes)
**$buffer-fname**(5) Name of the current buffer's file name
**$buffer-hilight**(5) Define current buffer hilighting scheme
**$buffer-indent**(5) Current buffer indentation scheme
**$buffer-indent-width**(5) Buffer indentation width in characters
**$buffer-input**(5) Divert buffer input through macro
**$buffer-ipipe**(5) Divert buffer incremental pipe input through macro
**$buffer-ipipe-flags**(5) Incremental pipe creation flags
**$buffer-major-mode**(5) Define buffer major-modem
**$buffer-mask**(5) Current buffer word class mask
**$buffer-mode-line**(5) Buffer mode line string
**$buffer-names**(5) Filtered buffer name list
**$buffer-scheme**(5) Buffer color scheme
**$buffer-tab-width**(5) Buffer tab character interval
**$buffer-xlate**(5) Buffer on-save translation configuration
**$build**(5) MicroEmacs build information
**$command-names**(5) Filtered command name list
**$cursor-blink**(5) Cursor blink rate
**$cursor-color**(5) Cursor foreground color
**$cursor-x**(5) Mouse X (horizontal) position
**$cursor-y**(5) Mouse Y (vertical) position
**$debug**(5) Macro debugging flag
**$delay-time**(5) Mouse time event delay time
**$file-ignore**(5) File extensions to ignore
**$file-names**(5) Filtered file name list
**$file-size-prompt**(5) Large file size warning prompt
**$file-template**(5) Regular expression file search string

**$fill-bullet**(5) Paragraph filling bullet character set
**$fill-bullet-len**(5) Paragraph filling bullet search depth
**$fill-col**(5) Global paragraph mode; right fill column
**$fill-eos**(5) Paragraph filling; end of sentence fill characters
**$fill-eos-len**(5) Paragraph filling; end of sentence padding length
**$fill-ignore**(5) Ignore paragraph filling character(s)
**$fill-mode**(5) Global paragraph mode; justification method
**$find-words**(5) Filtered word list
**$fmatchdelay**(5) Fence matching delay time
**$frame-depth**(5) Number of lines on the current frame canvas
**$frame-id**(5) Current frame identity
**$frame-title**(5) Frame title prefix
**$frame-width**(5) Number of columns on the current frame canvas
**$global-fmod**(5) Global file modes (or attributes)
**$global-scheme**(5) Global buffer color scheme
**$home**(5) Users 'home' directory location
**$idle-time**(5) System idle event delay time
**$indent-width**(5) Global indentation width in characters
**$kept-versions**(5) Number of backups to be kept
**$line-flags**(5) Current buffer line flags
**$line-scheme**(5) Set the current line color scheme
**$line-template**(5) Command line regular expression search string
**$major-mode-ids**(5) Filtered major-mode ID list
**$ml-scheme**(5) Message line color scheme
**$mode-chars**(5) Mode characters
**$mode-line**(5) Mode line format
**$mode-line-scheme**(5) Mode line color scheme
**$mode-names**(5) Filtered mode name list
**$mouse**(5) Mouse configuration variable
**$mouse-pos**(5) Mouse position information
**$mouse-x**(5) Mouse X (horizontal) position
**$mouse-y**(5) Mouse Y (vertical) position
**$osd-scheme**(5) OSD color scheme
**$pause-time**(5) Length of time to pause for user
**$platform**(5) MicroEmacs host platform identifier
**$progdata**(5) Users program data directory
**$progname**(5) Program file name
**$quiet**(5) Quiet mode control variable
**$random**(5) Generate a random number
**$rcs-ci-com**(5) RCS (and SCCS) check in command
**$rcs-cif-com**(5) RCS (and SCCS) check in first command
**$rcs-co-com**(5) RCS (and SCCS) check out command
**$rcs-cou-com**(5) RCS (and SCCS) check out unlock command
**$rcs-file**(5) RCS (and SCCS) file name
**$rcs-ue-com**(5) RCS (and SCCS) unedit file command
**$recent-keys**(5) Recent key history
**$repeat-time**(5) Mouse time event repeat time
**$result**(5) Various command return values
**$screen-depth**(5) Number of character lines on the screen canvas

**$screen-width**(5) Number of character columns on the screen canvas
**$scroll**(5) Screen scroll control
**$scroll-bar**(5) Scroll bar configuration
**$scroll-bar-scheme**(5) Scroll bar color scheme
**$search-path**(5) MicroEmacs search path
**$show-modes**(5) Select buffer modes to display
**$show-region**(5) Enable the hilighting of regions
**$status**(5) Macro command execution status
**$system**(5) System configuration variable
**$tab-width**(5) Global displayable tab character width
**$temp-name**(5) Temporary file name
**$time**(5) The current system time
**$timestamp**(5) Time stamp string
**$trunc-scheme**(5) Truncation color scheme
**$unix-time**(5) The current UNIX epoch time
**$user-name**(5) MicroEmacs user name
**$user-path**(5) MicroEmacs user directory
**$variable-names**(5) Filtered variable name list
**$version**(5) MicroEmacs version date-code
**$window-acol**(5) Window cursor actual column
**$window-aline**(5) Window cursor actual line
**$window-chars**(5) Character set used to render the windows
**$window-col**(5) Window cursor column (no expansion)
**$window-depth**(5) Number of text lines in a window
**$window-eline**(5) Window end line
**$window-flags**(5) Current window setup flags
**$window-id**(5) Current window identity
**$window-line**(5) Window cursor line
**$window-mark-col**(5) Window mark column
**$window-mark-line**(5) Window mark line
**$window-mode-line**(5) Window mode line position
**$window-pid**(5) Previous window identity
**$window-scroll-bar**(5) Window scroll bar (or separator) position
**$window-wcol**(5) Window cursor column (historic)
**$window-width**(5) Number of character columns in a window
**$window-wline**(5) Window cursor line (historic)
**$window-x-scroll**(5) Current window X scroll
**$window-xcl-scroll**(5) Current window current line X scroll
**$window-y-scroll**(5) Current window Y scroll
**%compile-com**(5) Default system compile command line
**%cvs-com**(5) CVS Command line
**%cygwin-hilight**(5) Cygwin shell hilight enable flag
**%cygwin-path**(5) Cygwin root directory
**%cygwin-prompt**(5) Cygwin shell prompt
**%diff-com**(5) Diff command line
**%ftp-flags**(5) Configure the FTP console
**%gdiff-com**(5) Gdiff command line
**%http-cookies**(5) Set HTTP request cookies
**%http-flags**(5) Configure the HTTP console

**%http-header**(5) Set additional HTTP request headers
**%http-post-data**(5) Set HTTP request form data
**%http-post-file**(5) Set HTTP request file to submit
**%http-proxy-addr**(5) Set HTTP proxy server address
**%http-proxy-port**(5) Set HTTP proxy server port
**%ishell**(5) User defined ishell shell
**%man-com**(5) Man Command line
**%os-chars**(5) Characters used by the OS
**%sock-recv-timeout**(5) Set socket receive timeout
**%sock-send-timeout**(5) Set socket send timeout
**%tag-file**(5) Tag file name
**%tag-option**(5) Tag file search option
**%tag-template**(5) Tag file search string
**%xdiff-com**(5) xdiff command line
**%xgrep-com**(5) Grep command line
**.about.arg0**(5) First command line option
**.about.arg1**(5) Second command line option
**.about.arg2**(5) Third command line option
**.about.argc**(5) Command line argument count
**.calc.answer**(5) Last successful calc answer
**.calc.result**(5) Last calc calculation result

## MACRO LANGUAGE GLOSSARY

The following is a list of all of the macro language commands available in **MicroEmacs**, refer to Writing MicroEmacs Macros for information on writing macros.

### Functions

All functions are denoted by a **&** prefix as follows:

**&abs**(4) Absolute value of a number
**&add**(4) Add two numbers
**&and**(4) Logical AND operator
**&atoi**(4) ASCII to integer conversion
**&band**(4) Bitwise AND operator
**&bmode**(4) Determine buffer mode
**&bnot**(4) Bitwise NOT operator
**&bor**(4) Bitwise OR operator
**&bstat**(4) Retrieve current buffer statistic
**&bxor**(4) Bitwise XOR operator
**&cat**(4) Concatenate two strings together
**&cbind**(4) Return the command a key is bound to
**&cond**(4) Conditional expression operator
**&dec**(4) Pre-decrement variable
**&divide**(4) Division of two numbers
**&equal**(4) Numerical equivalence operator
**&exist**(4) Test if a variable or command exists
**&fabs**(4) Absolute value of a floating point number
**&fadd**(4) Floating point add two numbers
**&fdivide**(4) Floating point division of two numbers
**&find**(4) Find a file on the search path
**&fmultiply**(4) Floating point multiply two numbers
**&fsqrt**(4) Floating point suqare root of a number
**&fsub**(4) Floating point subtract two numbers
**&gmode**(4) Determine global mode
**&great**(4) Numerical greater than operator
**&inc**(4) Pre-increment variable
**&indirect**(4) Evaluate a string as a variable
**&inword**(4) Test for a word character
**&irep**(4) Case insensitive replace string in string
**&isequal**(4) Case insensitive String equivalence operator
**&isin**(4) Case insensitive test for string in string
**&itoa**(4) Integer to ASCII conversion
**&kbind**(4) Return the key a command is bound to
**&ldel**(4) Delete list item
**&left**(4) Return the left most characters from a string
**&len**(4) Return the length of a string
**&less**(4) Numerical less than operator
**&lfind**(4) Find list item
**&lget**(4) Get list item
**&linsert**(4) Insert list item

**&llength**(4) Get list item count
**&lset**(4) Set list item
**&mid**(4) Return a portion (middle) of a string
**&mod**(4) Modulus of two numbers
**&multiply**(4) Multiply two numbers
**&nbind**(4) Return the numerial argument of a binding
**&nbmode**(4) Determine named buffer mode
**&negate**(4) Negation of two numbers
**&not**(4) Logical NOT operator
**&opt**(4) MicroEmacs optional feature test
**&or**(4) Logical OR operator
**&pdec**(4) Post-decrement variable
**&pinc**(4) Post-increment variable
**&reg**(4) Retrieve a registry value (with default)
**&rep**(4) Replace string in string
**&right**(4) Return the right most characters from a string
**&risin**(4) Recursive case insensitive test for string in string
**&rsin**(4) Recursively test for string in string
**&sequal**(4) String equivalence operator
**&set**(4) In-line macro variable assignment
**&sgreat**(4) String greater than operator
**&sin**(4) Test for string in string
**&sless**(4) String less than operator
**&slower**(4) Return the string converted to lower case
**&sprintf**(4) Formatted string construction
**&stat**(4) Retrieve a file statistic
**&sub**(4) Subtract two numbers
**&supper**(4) Return the string converted to upper case
**&trboth**(4) Return string trimmed of white chars on both sides
**&trleft**(4) Return string trimmed of white chars on left side
**&trright**(4) Return string trimmed of white chars on right side
**&uci**(4) Charater-set character Unicode info
**&ufi**(4) UTF-8 character Unicode info
**&uni**(4) Unicode character information
**&which**(4) Find a program on the path
**&xirep**(4) Regex case insensitive Replace string in string
**&xisequal**(4) Case insensitive regex String equivalence operator
**&xrep**(4) Regex replace string in string
**&xsequal**(4) Regex string equivalence operator

### Directives

The macro directives are denoted by a **!** prefix as follows:

**!abort**(4) Exit macro with a FALSE status
**!bell**(4) Sound audio alarm
**!break**(4) Exit loop
**!continue**(4) Restart loop
**!done**(4) Unconditional loop end
**!ehelp**(4) Terminate a help definition

**!elif**(4) Conditional test statement, continuation
**!else**(4) Conditional alternative statement
**!emacro**(4) Terminate a macro definition
**!endif**(4) Conditional test termination
**!force**(4) Ignore command or macro status
**!goto**(4) Unconditional labeled jump
**!if**(4) Conditional test statement
**!iif**(4) In-line conditional test and body
**!jump**(4) Unconditional jump
**!nmacro**(4) Execute line as if not in a macro
**!repeat**(4) Unconditional loop start
**!return**(4) Exit macro with a TRUE status
**!tgoto**(4) Conditional labeled jump
**!tjump**(4) Conditional relative branch
**!until**(4) Conditional loop end
**!while**(4) Conditional loop start

### Variables

The macro variables are denoted by a **%** for user variables; **#** for a register variable and @ for an interactive variable as follows:

**@#**(4) Macro arguments (numeric argument value)
**@0**(4) Macro arguments (macro name)
**@1**(4) Macro arguments (first argument)
**@2**(4) Macro arguments (second argument)
**@?**(4) Macro arguments (numeric argument given)
**@cc**(4) Current command name
**@cck**(4) Current command key
**@cg**(4) Get a command name from the user
**@cgk**(4) Get a key from the user
**@cl**(4) Last command name
**@clk**(4) Last command key
**@cq**(4) Get a quoted command name from the user
**@cqk**(4) Get a quoted key from the user
**@fs**(4) Frame store variable
**@fss**(4) Frame store scheme variable
**@h0**(4) General history string
**@h1**(4) Buffer history string
**@h2**(4) Command history string
**@h3**(4) File history string
**@h4**(4) Search history string
**@mc**(4) Message line character input request
**@ml**(4) Message line input request
**@mn**(4) Message line input as normal request
**@mna**(4) All input from Message line as normal
**@mx**(4) Message line input by executing command
**@mxa**(4) All input from Message line by executing command
**@p**(4) Macro arguments (calling macro name)
**@s0**(4) Last search's whole match string

**@s1**(4) Last search's first group value
**@s2**(4) Last search's second group value
**@wc**(4) Extract character from the current buffer
**@wl**(4) Extract a line from the current buffer
**@y**(4) Yank buffer variable
**BufferVariables**(4) Buffer variables
**CmdVariables**(4) Command variables
**CommandVariables**(4) Last, current and get a command key/name
**CurrentBufferVariables**(4) Extract information from the current buffer
**HistoryGroupVariables**(4) Last history values
**MacroArguments**(4) Arguments to macros
**MacroNumericArguments**(4) Numeric arguments to macros
**MessageLineVariables**(4) Prompt the user for input on message line
**RegisterVariables**(4) Register variables
**SearchGroups**(4) Last search group values
**Variables**(4) User defined macro variables

## GLOSSARY

The following is a list of all keywords associated with **MicroEmacs**:

**!abort**(4) Exit macro with a FALSE status
**!bell**(4) Sound audio alarm
**!break**(4) Exit loop
**!continue**(4) Restart loop
**!done**(4) Unconditional loop end
**!ehelp**(4) Terminate a help definition
**!elif**(4) Conditional test statement, continuation
**!else**(4) Conditional alternative statement
**!emacro**(4) Terminate a macro definition
**!endif**(4) Conditional test termination
**!force**(4) Ignore command or macro status
**!goto**(4) Unconditional labeled jump
**!if**(4) Conditional test statement
**!iif**(4) In-line conditional test and body
**!jump**(4) Unconditional jump
**!nmacro**(4) Execute line as if not in a macro
**!repeat**(4) Unconditional loop start
**!return**(4) Exit macro with a TRUE status
**!tgoto**(4) Conditional labeled jump
**!tjump**(4) Conditional relative branch
**!until**(4) Conditional loop end
**!while**(4) Conditional loop start
**$INFOPATH**(5) GNU info files base directory
**$LOGNAME**(5) System user name (UNIX)
**$MEBACKUPPATH**(5) Backup file location
**$MEBACKUPSUB**(5) Backup file name modifier
**$MEINSTALLPATH**(5) MicroEmacs system macros search path
**$MENAME**(5) MicroEmacs user name
**$MEPATH**(5) MicroEmacs search path
**$MEUSERPATH**(5) MicroEmacs user directory
**$ME_PIPE_STDERR**(5) Command line diversion to stderr symbol
**$auto-time**(5) Automatic buffer save time
**$box-chars**(5) Characters used to draw lines
**$buffer-backup**(5) Buffer backup file name
**$buffer-bhook**(5) Buffer macro hook command name (buffer current)
**$buffer-bname**(5) Name of the current buffer
**$buffer-dhook**(5) Buffer macro hook command name (buffer deletion)
**$buffer-ehook**(5) Buffer macro hook command name (buffer swapped)
**$buffer-fhook**(5) Buffer macro hook command name (buffer creation)
**$buffer-fill-col**(5) Buffer paragraph mode; right fill column
**$buffer-fill-mode**(5) Buffer paragraph mode; justification method
**$buffer-fmod**(5) Buffer file modes (or attributes)
**$buffer-fname**(5) Name of the current buffer's file name
**$buffer-hilight**(5) Define current buffer hilighting scheme
**$buffer-indent**(5) Current buffer indentation scheme
**$buffer-indent-width**(5) Buffer indentation width in characters

**$buffer-input**(5) Divert buffer input through macro
**$buffer-ipipe**(5) Divert buffer incremental pipe input through macro
**$buffer-ipipe-flags**(5) Incremental pipe creation flags
**$buffer-major-mode**(5) Define buffer major-modem
**$buffer-mask**(5) Current buffer word class mask
**$buffer-mode-line**(5) Buffer mode line string
**$buffer-names**(5) Filtered buffer name list
**$buffer-scheme**(5) Buffer color scheme
**$buffer-tab-width**(5) Buffer tab character interval
**$buffer-xlate**(5) Buffer on-save translation configuration
**$build**(5) MicroEmacs build information
**$command-names**(5) Filtered command name list
**$cursor-blink**(5) Cursor blink rate
**$cursor-color**(5) Cursor foreground color
**$cursor-x**(5) Mouse X (horizontal) position
**$cursor-y**(5) Mouse Y (vertical) position
**$debug**(5) Macro debugging flag
**$delay-time**(5) Mouse time event delay time
**$file-ignore**(5) File extensions to ignore
**$file-names**(5) Filtered file name list
**$file-size-prompt**(5) Large file size warning prompt
**$file-template**(5) Regular expression file search string
**$fill-bullet**(5) Paragraph filling bullet character set
**$fill-bullet-len**(5) Paragraph filling bullet search depth
**$fill-col**(5) Global paragraph mode; right fill column
**$fill-eos**(5) Paragraph filling; end of sentence fill characters
**$fill-eos-len**(5) Paragraph filling; end of sentence padding length
**$fill-ignore**(5) Ignore paragraph filling character(s)
**$fill-mode**(5) Global paragraph mode; justification method
**$find-words**(5) Filtered word list
**$fmatchdelay**(5) Fence matching delay time
**$frame-depth**(5) Number of lines on the current frame canvas
**$frame-id**(5) Current frame identity
**$frame-title**(5) Frame title prefix
**$frame-width**(5) Number of columns on the current frame canvas
**$global-fmod**(5) Global file modes (or attributes)
**$global-scheme**(5) Global buffer color scheme
**$home**(5) Users ‘home’ directory location
**$idle-time**(5) System idle event delay time
**$indent-width**(5) Global indentation width in characters
**$kept-versions**(5) Number of backups to be kept
**$line-flags**(5) Current buffer line flags
**$line-scheme**(5) Set the current line color scheme
**$line-template**(5) Command line regular expression search string
**$major-mode-ids**(5) Filtered major-mode ID list
**$ml-scheme**(5) Message line color scheme
**$mode-chars**(5) Mode characters
**$mode-line**(5) Mode line format
**$mode-line-scheme**(5) Mode line color scheme

**$mode-names**(5) Filtered mode name list
**$mouse**(5) Mouse configuration variable
**$mouse-pos**(5) Mouse position information
**$mouse-x**(5) Mouse X (horizontal) position
**$mouse-y**(5) Mouse Y (vertical) position
**$osd-scheme**(5) OSD color scheme
**$pause-time**(5) Length of time to pause for user
**$platform**(5) MicroEmacs host platform identifier
**$progdata**(5) Users program data directory
**$progname**(5) Program file name
**$quiet**(5) Quiet mode control variable
**$random**(5) Generate a random number
**$rcs-ci-com**(5) RCS (and SCCS) check in command
**$rcs-cif-com**(5) RCS (and SCCS) check in first command
**$rcs-co-com**(5) RCS (and SCCS) check out command
**$rcs-cou-com**(5) RCS (and SCCS) check out unlock command
**$rcs-file**(5) RCS (and SCCS) file name
**$rcs-ue-com**(5) RCS (and SCCS) unedit file command
**$recent-keys**(5) Recent key history
**$repeat-time**(5) Mouse time event repeat time
**$result**(5) Various command return values
**$screen-depth**(5) Number of character lines on the screen canvas
**$screen-width**(5) Number of character columns on the screen canvas
**$scroll**(5) Screen scroll control
**$scroll-bar**(5) Scroll bar configuration
**$scroll-bar-scheme**(5) Scroll bar color scheme
**$search-path**(5) MicroEmacs search path
**$show-modes**(5) Select buffer modes to display
**$show-region**(5) Enable the hilighting of regions
**$status**(5) Macro command execution status
**$system**(5) System configuration variable
**$tab-width**(5) Global displayable tab character width
**$temp-name**(5) Temporary file name
**$time**(5) The current system time
**$timestamp**(5) Time stamp string
**$trunc-scheme**(5) Truncation color scheme
**$unix-time**(5) The current UNIX epoch time
**$user-name**(5) MicroEmacs user name
**$user-path**(5) MicroEmacs user directory
**$variable-names**(5) Filtered variable name list
**$version**(5) MicroEmacs version date-code
**$window-acol**(5) Window cursor actual column
**$window-aline**(5) Window cursor actual line
**$window-chars**(5) Character set used to render the windows
**$window-col**(5) Window cursor column (no expansion)
**$window-depth**(5) Number of text lines in a window
**$window-eline**(5) Window end line
**$window-flags**(5) Current window setup flags
**$window-id**(5) Current window identity

**$window-line**(5) Window cursor line
**$window-mark-col**(5) Window mark column
**$window-mark-line**(5) Window mark line
**$window-mode-line**(5) Window mode line position
**$window-pid**(5) Previous window identity
**$window-scroll-bar**(5) Window scroll bar (or separator) position
**$window-wcol**(5) Window cursor column (historic)
**$window-width**(5) Number of character columns in a window
**$window-wline**(5) Window cursor line (historic)
**$window-x-scroll**(5) Current window X scroll
**$window-xcl-scroll**(5) Current window current line X scroll
**$window-y-scroll**(5) Current window Y scroll
**%ascii-time**(5) Set the default format of the date string
**%company-name**(5) Name of company for template
**%company-ref**(5) Company reference template
**%company-url**(5) Company URL for template
**%compile-com**(5) Default system compile command line
**%cvs-com**(5) CVS Command line
**%cygwin-hilight**(5) Cygwin shell hilight enable flag
**%cygwin-path**(5) Cygwin root directory
**%cygwin-prompt**(5) Cygwin shell prompt
**%diff-com**(5) Diff command line
**%ftp-flags**(5) Configure the FTP console
**%gdiff-com**(5) Gdiff command line
**%http-cookies**(5) Set HTTP request cookies
**%http-flags**(5) Configure the HTTP console
**%http-header**(5) Set additional HTTP request headers
**%http-post-data**(5) Set HTTP request form data
**%http-post-file**(5) Set HTTP request file to submit
**%http-proxy-addr**(5) Set HTTP proxy server address
**%http-proxy-port**(5) Set HTTP proxy server port
**%ishell**(5) User defined ishell shell
**%man-com**(5) Man Command line
**%os-chars**(5) Characters used by the OS
**%sock-recv-timeout**(5) Set socket receive timeout
**%sock-send-timeout**(5) Set socket send timeout
**%tag-file**(5) Tag file name
**%tag-option**(5) Tag file search option
**%tag-template**(5) Tag file search string
**%xdiff-com**(5) xdiff command line
**%xfind-com**(5) Find command line
**%xgrep-com**(5) Grep command line
**&abs**(4) Absolute value of a number
**&add**(4) Add two numbers
**&and**(4) Logical AND operator
**&atoi**(4) ASCII to integer conversion
**&band**(4) Bitwise AND operator
**&bmode**(4) Determine buffer mode
**&bnot**(4) Bitwise NOT operator

**&bor**(4) Bitwise OR operator
**&bstat**(4) Retrieve current buffer statistic
**&bxor**(4) Bitwise XOR operator
**&cat**(4) Concatenate two strings together
**&cbind**(4) Return the command a key is bound to
**&cond**(4) Conditional expression operator
**&dec**(4) Pre-decrement variable
**&divide**(4) Division of two numbers
**&equal**(4) Numerical equivalence operator
**&exist**(4) Test if a variable or command exists
**&fabs**(4) Absolute value of a floating point number
**&fadd**(4) Floating point add two numbers
**&fdivide**(4) Floating point division of two numbers
**&find**(4) Find a file on the search path
**&fmultiply**(4) Floating point multiply two numbers
**&fsqrt**(4) Floating point suqare root of a number
**&fsub**(4) Floating point subtract two numbers
**&gmode**(4) Determine global mode
**&great**(4) Numerical greater than operator
**&inc**(4) Pre-increment variable
**&indirect**(4) Evaluate a string as a variable
**&inword**(4) Test for a word character
**&irep**(4) Case insensitive replace string in string
**&isequal**(4) Case insensitive String equivalence operator
**&isin**(4) Case insensitive test for string in string
**&itoa**(4) Integer to ASCII conversion
**&kbind**(4) Return the key a command is bound to
**&ldel**(4) Delete list item
**&left**(4) Return the left most characters from a string
**&len**(4) Return the length of a string
**&less**(4) Numerical less than operator
**&lfind**(4) Find list item
**&lget**(4) Get list item
**&linsert**(4) Insert list item
**&llength**(4) Get list item count
**&lset**(4) Set list item
**&mid**(4) Return a portion (middle) of a string
**&mod**(4) Modulus of two numbers
**&multiply**(4) Multiply two numbers
**&nbind**(4) Return the numerial argument of a binding
**&nbmode**(4) Determine named buffer mode
**&negate**(4) Negation of two numbers
**&not**(4) Logical NOT operator
**&opt**(4) MicroEmacs optional feature test
**&or**(4) Logical OR operator
**&pdec**(4) Post-decrement variable
**&pinc**(4) Post-increment variable
**&reg**(4) Retrieve a registry value (with default)
**&rep**(4) Replace string in string

**&right**(4) Return the right most characters from a string
**&risin**(4) Recursive case insensitive test for string in string
**&rsin**(4) Recursively test for string in string
**&sequal**(4) String equivalence operator
**&set**(4) In-line macro variable assignment
**&sgreat**(4) String greater than operator
**&sin**(4) Test for string in string
**&sless**(4) String less than operator
**&slower**(4) Return the string converted to lower case
**&sprintf**(4) Formatted string construction
**&stat**(4) Retrieve a file statistic
**&sub**(4) Subtract two numbers
**&supper**(4) Return the string converted to upper case
**&trboth**(4) Return string trimmed of white chars on both sides
**&trleft**(4) Return string trimmed of white chars on left side
**&trright**(4) Return string trimmed of white chars on right side
**&uci**(4) Charater-set character Unicode info
**&ufi**(4) UTF-8 character Unicode info
**&uni**(4) Unicode character information
**&which**(4) Find a program on the path
**&xirep**(4) Regex case insensitive Replace string in string
**&xisequal**(4) Case insensitive regex String equivalence operator
**&xrep**(4) Regex replace string in string
**&xsequal**(4) Regex string equivalence operator
**.Xdefaults**(8) UNIX Window Manager defaults file
**.about.arg0**(5) First command line option
**.about.arg1**(5) Second command line option
**.about.arg2**(5) Third command line option
**.about.argc**(5) Command line argument count
**.calc.answer**(5) Last successful calc answer
**.calc.result**(5) Last calc calculation result
**.which.result**(5) Program path
**0-9**(9) UNIX t/nroff file
**2dos**(3f) Convert text files to DOS format files.
**2mac**(3f) Convert text files to Macintosh format files
**2unix**(3f) Convert text files to UNIX format files
**2win**(3f) Convert text files to Windows format files.
**@#**(4) Macro arguments (numeric argument value)
**@0**(4) Macro arguments (macro name)
**@1**(4) Macro arguments (first argument)
**@2**(4) Macro arguments (second argument)
**@?**(4) Macro arguments (numeric argument given)
**@cc**(4) Current command name
**@cck**(4) Current command key
**@cg**(4) Get a command name from the user
**@cgk**(4) Get a key from the user
**@cl**(4) Last command name
**@clk**(4) Last command key
**@cq**(4) Get a quoted command name from the user

**@cqk**(4) Get a quoted key from the user
**@fs**(4) Frame store variable
**@fss**(4) Frame store scheme variable
**@h0**(4) General history string
**@h1**(4) Buffer history string
**@h2**(4) Command history string
**@h3**(4) File history string
**@h4**(4) Search history string
**@mc**(4) Message line character input request
**@ml**(4) Message line input request
**@mn**(4) Message line input as normal request
**@mna**(4) All input from Message line as normal
**@mx**(4) Message line input by executing command
**@mxa**(4) All input from Message line by executing command
**@p**(4) Macro arguments (calling macro name)
**@s0**(4) Last search's whole match string
**@s1**(4) Last search's first group value
**@s2**(4) Last search's second group value
**@wc**(4) Extract character from the current buffer
**@wl**(4) Extract a line from the current buffer
**@y**(4) Yank buffer variable
**Arkanoid**(3) MicroEmacs version of the bat and ball wall game
**abbrev-list**(3) Create a list of abbreviations
**abort-command**(2) (**C-g**) Abort command
**about**(2) Information About MicroEmacs
**ada**(9) Ada programming language
**adb**(9) Ada programming language
**add-color**(2) Create a new color
**add-color-scheme**(2) Create a new color scheme
**add-dictionary**(2) Declare existence of a spelling dictionary
**add-file-hook**(2) DEPRICATED: Declare file name context dependent configuration
**add-global-mode**(3) Set a global buffer mode
**add-mode**(3) Set a local buffer mode
**add-next-line**(2) Define the searching behavior of command output
**add-spell-rule**(2) Add a new spelling rule to the dictionary
**adoc**(9) AsciiDoc Markup File
**ads**(9) Ada programming language
**alarm**(3) Set an alarm
**alias-path-setup**(3) Define alternative path alias
**aman**(3) Compile an nroff file into a buffer (UNIX)
**apache**(9) Apache httpd.conf configuration file
**append-buffer**(2) Write contents of buffer to end of named file
**ascii-time**(3) Return the current time as a string
**asciidoc**(9) AsciiDoc Markup File
**asm**(9) Assembler File
**asn.1**(9) ASN.1 file
**asp**(9) Active Server Pages
**au3**(9) Autoit 3 Scripting Language File
**auto**(2m) Automatic source file line type detection

**auto-spell**(3) Auto-spell support
**auto-spell-buffer**(3) Auto-spell whole buffer
**auto-spell-ignore**(3) Auto-spell ignore current word
**auto-spell-menu**(3) Auto-spell context menu
**auto-spell-reset**(3) Auto-spell hilight reset
**autoit**(9) Autoit 3 Scripting Language File
**autosv**(2m) Automatic file save
**awk**(9) AWK File
**Balls**(3) MicroEmacs bouncing ball playground
**Bindings**(2) Default Key Bindings
**BufferVariables**(4) Buffer variables
**Build**(2) Build
**backup**(2m) Automatic file backup of last edit
**backward-char**(2) (**C-b**) Move the cursor left
**backward-delete-char**(2) (**backspace**) Delete the previous character at the cursor position
**backward-delete-tab**(2) (**S-tab**) Delete white space to previous tab-stop
**backward-kill-word**(2) (**esc backspace**) Delete the previous word at the cursor position
**backward-line**(2) (**C-p**) Move the cursor to the previous line
**backward-paragraph**(2) (**esc p**) Move the cursor to the previous paragraph
**backward-word**(2) (**esc b**) Move the cursor to the previous word
**bas**(9) Visual Basic
**bat**(9) MS-DOS Batch File
**beginning-of-buffer**(2) (**esc <**) Move to beginning of buffer/file
**beginning-of-line**(2) (**C-a**) Move to beginning of line
**benchmrk**(3f) Benchmark MicroEmacs macro processor speed
**bib**(9) TeX Bibliography File
**bin-copy-region**(3) Binary supporting copy region
**bin-yank**(3) Binary supporting kill buffer insert
**binary**(2m) Binary editor mode
**bnf**(9) Backus-Naur Form
**browse**(3) Browse a file or buffer in view mode
**btm**(9) 4-DOS Batch File
**buffer-abbrev-file**(2) Set buffers' abbreviation file
**buffer-bind-key**(2) Create local key binding for current buffer
**buffer-info**(2) (**C-x =**) Status information on current buffer position
**buffer-init**(3) Initialize a buffer language template
**buffer-init-fhook**(3) Initialize the buffer file hooks
**buffer-init-hooks**(3) Initialize the buffer language template hooks
**buffer-is-utf8**(3) Detect if the current buffer is UTF-8 encoded
**buffer-major-mode**(3) Change the operating mode of a buffer
**buffer-mode**(2) (**C-x m**) Change a local buffer mode
**buffer-setup**(3) Configure the current buffer settings
**buffer-unbind-key**(2) Remove local key binding for current buffer
**C#**(9) C# programming language
**C3**(9) C3 programming language
**ClearCase**(3) MicroEmacs ClearCase interface
**Client-Server**(2) Client-Server Model
**CmdVariables**(4) Command variables
**CommandVariables**(4) Last, current and get a command key/name

**CompanyProfiles**(2) Defining a company profile
**Compatibility**(2) Compatibility with the original MicroEmacs
**CurrentBufferVariables**(4) Extract information from the current buffer
**c**(9) C programming language
**c-deslashify**(3) C language, remove backslash to end of lines of region
**c-hash-define**(3) Set a C/C++ #define
**c-hash-eval**(3) Evaluate C/C++ #defines
**c-hash-undef**(3) Unset a C/C++ #define
**c-hash-undo**(3) Remove C/C++ #define evaluation
**c-slashify**(3) C language, append backslash to end of lines of region
**c-to-cpp-convert**(3) Convert a C file to C++
**calc**(3) Calculator
**calc-op**(3) Low level calculator command
**capitalize-word**(2) (**esc c**) Capitalize word
**cbl**(9) Cobol (85) File
**cc**(9) C++ programming language
**cc-add**(3) MicroEmacs ClearCase interface - add file
**cc-checkin**(3) MicroEmacs ClearCase interface - checkin files
**cc-checkout**(3) MicroEmacs ClearCase interface - checkout files
**cc-delete**(3) MicroEmacs ClearCase interface - delete files
**cc-diff**(3) MicroEmacs ClearCase interface - diff changes
**cc-setup**(3) MicroEmacs ClearCase interface setup
**cc-state**(3) MicroEmacs ClearCase interface - list state of directory files
**cc-update**(3) MicroEmacs ClearCase interface - update directory files
**cfc**(9) ColdFusion Markup Language
**cfm**(9) ColdFusion Markup Language
**cfml**(9) Adobe Macromedia ColdFusion Markup Language
**change-buffer-charset**(3) Convert buffer between two character sets
**change-buffer-name**(2) (**esc C-n**) Change name of current buffer
**change-company**(3) Change current company profile
**change-file-name**(2) (**C-x n**) Change the file name of the current buffer
**change-font**(2) Change the screen font
**change-font-size**(3) Incrementally change font size
**change-frame-depth**(2) Change the number of lines on the current frame
**change-frame-width**(2) Change the number of columns on the current frame
**change-scheme**(3) Change the current highlight scheme
**change-window-depth**(2) Change the depth of the current window
**change-window-width**(2) Change the width of the current window
**check-line-length**(3) Check the length of text lines are valid
**clean**(3) Remove redundant white spaces from the current buffer
**cls**(9) Visual Basic
**collapse-all**(3) (**f3**) (Un)Fold all regions in the current buffer
**collapse-current**(3) (**f2**) (un)Fold a region in the current buffer
**command-apropos**(2) (**C-h a**) List commands involving a concept
**command-wait**(2) Conditional wait command
**comment-end**(3) End the current comment
**comment-line**(3) Comment out the current line
**comment-restyle**(3) Reformat the current comment
**comment-start**(3) Start a new comment

**comment-to-end-of-line**(3) Extend comment to end of line
**compare-windows**(2) Compare buffer windows, ignore whitespace
**compare-windows-exact**(3) Compare buffer windows, with whitespace
**compile**(3) Start a compilation process
**copy-file**(3) Copy or move files and directories
**copy-rectangle**(3) Copy a rectangular region of text
**copy-region**(2) (**esc w**) Copy a region of the buffer
**copy-registry**(3) Recursive copy of registry subtree
**count-words**(2) (**esc C-c**) Count the number of words in a region
**cpp**(9) C++ programming language
**cr**(2m) File's carriage return (CR) style
**create-callback**(2) Create a timer callback
**create-frame**(2) (**C-x 5 1**, **C-x 5 2**) Create a new frame
**crypt**(2m) Encrypted file mode
**cs**(9) C# programming language
**csh**(9) C-Shell file
**ctags**(3f) Generate a C tags file
**ctrlz**(2m) File's termination style
**cursor**(2m) Buffer's cursor visibility
**cvs**(3) MicroEmacs CVS interface
**cvs-add**(3) MicroEmacs CVS interface - add file
**cvs-annotate**(3) MicroEmacs CVS interface - annotate file
**cvs-cat-file**(3) MicroEmacs CVS interface - output specific revision of a file
**cvs-checkout**(3) MicroEmacs CVS interface - checkout files
**cvs-commit**(3) MicroEmacs CVS interface - commit changes
**cvs-diff**(3) MicroEmacs CVS interface - diff changes
**cvs-file-status**(3) MicroEmacs CVS interface - status of files
**cvs-gdiff**(3) MicroEmacs CVS interface - graphical diff changes
**cvs-log-file**(3) MicroEmacs CVS interface - log changes
**cvs-remove**(3) MicroEmacs CVS interface - remove file
**cvs-resolve**(3) MicroEmacs CVS interface - resolve conflicts
**cvs-resolve-conflicts**(3) MicroEmacs CVS interface - resolve conflicts
**cvs-setup**(3) MicroEmacs CVS interface setup
**cvs-status**(3) MicroEmacs CVS interface - list state of directory files
**cvs-tag**(3) MicroEmacs CVS interface - tag files
**cvs-update**(3) MicroEmacs CVS interface - update directory files
**cygwin**(3) Open a Cygwin BASH window
**dart**(9) Dart programming language
**dbx**(3) UNIX Debugger
**def**(9) C or C++ definition file
**define-help**(2) Define help information
**define-macro**(2) Define a new macro
**define-macro-file**(2) Define macro file location
**delete-blank-lines**(2) (**C-x C-o**) Delete blank lines about cursor
**delete-buffer**(2) (**C-x k**) Delete a buffer
**delete-dictionary**(2) Remove a spelling dictionary from memory
**delete-file**(3) Delete files and directories
**delete-file-delayed**(3) Delete file after a specified delay
**delete-frame**(2) (**C-x 5 0**) Delete the current frame

**delete-global-mode**(3) Remove a global buffer mode
**delete-hilight-lines**(3) Delete hilight lines
**delete-indentation**(3) Join 2 lines deleting white spaces
**delete-mode**(3) Remove a local buffer mode
**delete-other-windows**(2) (**C-x 1**) Delete other windows
**delete-registry**(2) Delete a registry tree
**delete-some-buffers**(2) Delete buffers with query
**delete-window**(2) (**C-x 0**) Delete current window
**describe-bindings**(2) (**C-h b**) Show current command/key binding
**describe-key**(2) (**C-x ?**) Report keyboard key name and binding
**describe-variable**(2) (**C-h v**) Describe current setting of a variable
**describe-word**(3) Display a dictionary definition of a word
**diff**(3) Difference files or directories
**diff-changes**(3) Find the differences from a previous edit session
**diff-lines**(3) Highlight the differences between two lines
**directory-tree**(2) Draw the file directory tree
**display-matching-fence**(3) Display the matching bracket
**display-white-chars**(3) Toggle the displaying of white characters
**display-whitespaces**(3) Toggle the displaying of white characters
**doc**(9) ASCII plain text document file
**doctools**(9) Tcl Doctools Markup File
**dos2unix**(3f) Convert DOS format files to UNIX format files
**dot**(9) Graphviz Dot Diagram Language File
**draw**(3) Simple line drawing utility
**e**(9) Euphoria File
**eaf**(8) MicroEmacs abbreviation file format
**edf**(8) MicroEmacs spelling dictionary file
**edit**(2m) Buffer has be changed
**edit-dictionary**(3) Insert a dictionary in a buffer
**ehf**(8) MicroEmacs help file
**ehf**(9) MicroEmacs help file
**ehftools**(3f) Generate a MicroEmacs help file
**emf**(8) MicroEmacs macro file
**emf**(9) MicroEmacs Macro File
**emftags**(3f) Generate a MicroEmacs macro tags file
**encase**(3) Encase a region with start/end string
**end-kbd-macro**(2) (**C-x )**) Stop recording keyboard macro
**end-of-buffer**(2) (**esc >**) Move to end of buffer/file
**end-of-line**(2) (**C-e**) Move to end of line
**enf**(8) Notes database file
**erf**(8) MicroEmacs registry file
**erf**(9) MicroEmacs registry file
**erl**(9) Erlang programming language
**esf**(8) MicroEmacs session file
**etf**(8) MicroEmacs template file format
**euphoria**(9) Euphoria File
**exact**(2m) Searching and sorting case sensitivity
**exchange-point-and-mark**(2) (**C-x C-x**) Exchange the cursor and marked position
**execute-buffer**(2) Execute script lines from a buffer

**execute-file**(2) (**esc /**) Execute script lines from a file
**execute-kbd-macro**(2) (**C-x e**) Execute a keyboard macro
**execute-line**(2) Execute a typed in script line
**execute-named-command**(2) [**esc x**] Execute a named command
**execute-string**(2) Execute a string as a command
**execute-tool**(3) Execute a user defined shell tool
**exit-emacs**(2) Exit MicroEmacs
**expand-abbrev**(2) Expand an abbreviation
**expand-abbrev-handle**(3) (**esc esc**) Expand an abbreviation handler
**expand-iso-accents**(3) Expand an ISO accent
**expand-look-back**(3) Complete a word by looking back for a similar word
**expand-word**(3) Complete a word by invocation of the speller
**f**(9) Fortran File
**f77**(9) Fortran 77 File
**f90**(9) Fortran 90 File
**fence**(2m) Auto fence matching mode
**file-attrib**(3) Set the current buffers system file attributes
**file-browser**(3) (**f10**) Browse the file system
**file-browser-close**(3) Close the file-browser
**file-browser-swap-buffers**(3) Swap between file-browser windows
**file-exec**(3) Executes the current buffer with compiler/interpreter
**file-format**(3) Format the current buffer using the a formatter
**file-lint**(3) check the current buffer for problems using a linter
**file-list**(3) Generate list of matching file names
**file-op**(2) File system operations command
**file-tool-setup**(3) Define file browser tools
**file-type-setup**(3) Define file type set
**fileHooks**(2) File Hooks
**filemask-to-regex**(3) Convert a file mask to a regex
**filetype**(3f) Determine file type UNIX, DOS or Windows
**fill-paragraph**(2) (**esc o**) Format a paragraph
**filter-buffer**(2) (**C-x #**) Filter the current buffer through an O/S command
**find**(3) Execute built-in find command
**find-bfile**(3) (**C-x 9**) Load a file as binary data
**find-buffer**(2) (**C-x b**) Switch to a named buffer
**find-cfile**(3) Load an encrypted file
**find-file**(2) (**C-x C-f**) Load a file
**find-hfile**(3) (**C-x C-h**) Load a file using selected handler
**find-lfile**(3) Load a file lossless
**find-registry**(2) Index search of a registry sub-tree
**find-rfile**(3) Load a file as binary data (reduced)
**find-setup**(3) Setup the built-in find and grep commands
**find-tag**(2) (**esc t**) Find tag, auto-load file and move to tag position
**find-ufile**(3) Load a Unicode file
**find-word**(3) Find a word using spelling dictionaries
**find-zfile**(3) Compressed file support
**fold-close**(3) (**C-f2**) Close an open fold (in-line)
**fold-create**(3) (**C-f2**) Create a new fold
**fold-enter**(3) (**f3**) Enter fold

**fold-exit**(3) (**esc f3**) Exit a fold
**fold-exit-all**(3) (**esc C-f3**) Exit all folds
**fold-kill**(3) (**C-x C-k**) Delete the fold and all its contents
**fold-open**(3) (**f2**) Open fold (in-line)
**fold-remove**(3) (**esc C-f3**) Remove the fold, maintaining content
**forward-char**(2) (**C-f**) Move the cursor right
**forward-delete-char**(2) (**C-d**) Delete the next character at the cursor position
**forward-kill-word**(2) (**esc d**) Delete the next word at the cursor position
**forward-line**(2) (**C-n**) Move the cursor to the next line
**forward-paragraph**(2) (**esc n**) Move the cursor to the next paragraph
**forward-word**(2) (**esc f**) Move the cursor to the next word
**ftp**(3) Initiate an FTP connection
**fu**(9) Fusion transpiler programming language
**fvwm**(9) FVWM configuration file
**fvwmrc**(9) FVWM configuration file
**gawk**(9) GNU AWK File
**gdb**(3) GNU Debugger
**gdiff**(3) Graphical file difference
**gdiff**(3f) Command line graphical file difference
**generate-hash**(2) Generate SHA1 or SHA256 hash or HMAC
**generate-password**(3) Generate a password with given criteria
**generate-tags-file**(3) Generate a tags file
**get-next-line**(2) (**C-x ‘**) Find the next command line
**git**(3) MicroEmacs GIT interface
**git-add**(3) MicroEmacs GIT interface - add items to the index
**git-blame**(3) MicroEmacs GIT interface - annotate a file
**git-cat-file**(3) MicroEmacs GIT interface - output specific commit of a file
**git-commit**(3) MicroEmacs GIT interface - commit changes to repository
**git-diff**(3) MicroEmacs GIT interface - diff changes
**git-fetch**(3) MicroEmacs GIT interface - download from another repository
**git-gdiff**(3) MicroEmacs GIT interface - graphical diff changes
**git-log-commit**(3) MicroEmacs GIT interface - show information about a specific commit
**git-log-file**(3) MicroEmacs GIT interface - list log of changes
**git-move**(3) MicroEmacs GIT interface - move an item
**git-pull**(3) MicroEmacs GIT interface - download and integrate another repository
**git-push**(3) MicroEmacs GIT interface - update remote repository
**git-remove**(3) MicroEmacs GIT interface - remove items
**git-revert**(3) MicroEmacs GIT interface - revert local changes
**git-setup**(3) MicroEmacs GIT interface setup
**git-status**(3) MicroEmacs GIT interface - status of items
**git-switch**(3) MicroEmacs GIT interface - switch or create branches
**git-unstage**(3) MicroEmacs GIT interface - unstage items
**global-abbrev-file**(2) Set global abbreviation file
**global-bind-key**(2) (**esc k**) Bind a key to a named command or macro
**global-mode**(2) (**esc m**) Change a global buffer mode
**global-unbind-key**(2) (**esc C-k**) Unbind a key from a named command or macro
**go**(9) Go programming language
**goto-alpha-mark**(2) (**C-x a**) Move the cursor to a alpha marked location
**goto-line**(2) (**esc g**) Move the cursor to specified line

**goto-matching-fence**(2) (**esc C-f**) Move the cursor to matching fence
**goto-position**(2) Restore a stored position
**goto-window**(2) Restore a saved window to the current window (historic)
**graphviz**(9) Graphviz Dot Diagram Language File
**grep**(3) Search a file for a pattern
**grep-for-region**(3) Grep current directory for current region
**groovy**(9) Groovy programming language
**grow-window-horizontally**(2) Enlarge current window horizontally (relative)
**grow-window-vertically**(2) Enlarge the current window (relative change)
**Hearts**(3) MicroEmacs version of the card game Hearts
**HistoryGroupVariables**(4) Last history values
**h**(9) C programming language header
**haskell**(9) Haskell programming language
**help**(2) (**esc ?**) Help; high level introduction to help
**help-command**(2) (**C-h C-c**) Help; command information
**help-item**(2) (**C-h C-i**) Help; item information
**help-variable**(2) (**C-h C-v**) Help; variable information
**hide**(2m) Hide buffer
**hilight**(2) Manage the buffer hilighting schemes
**hpj**(9) MS-Windows Help Project File
**hs**(9) Haskell programming language
**htm**(9) HyperText Markup Language File
**html**(9) HyperText Markup Language File
**httpd.conf**(9) Apache configuration file
**hunt-backward**(2) (**C-x C-h**) Resume previous search in backward direction
**hunt-forward**(2) (**C-x h**) Resume previous search in forward direction
**hx**(9) Haxe transpiler programming language
**Installation**(1) Installation details for MicroEmacs
**Interfacing**(2) Interfacing to external components
**i**(9) C/C++ preprocessor output file
**ifill-paragraph**(3) (**esc q**) Format a paragraph
**imakefile**(9) Make file
**indent**(2) Manage the auto-indentation methods
**indent**(2m) Automatic indentation
**indent-decrease**(3) Decrease the line indentation
**indent-increase**(3) Increase the line indentation
**info**(3) Display a GNU Info database
**info**(3f) Display Info pages in MicroEmacs.
**info**(9) GNU Info file
**info-goto-link**(3) Display Info on a given link
**info-on**(3) Display Info on a given topic
**ini**(9) MS-Windows Initialization File
**insert-calc**(3) Calculator with answer insertion
**insert-file**(2) (**C-x C-i**) Insert file into current buffer
**insert-file-name**(2) (**C-x C-y**) Insert filename into current buffer
**insert-macro**(2) Insert keyboard macro into buffer
**insert-newline**(2) (**C-o**) Insert new line at cursor position
**insert-password**(3) Create and insert a new password string
**insert-space**(2) Insert space(s) into current buffer

**insert-string**(2) Insert character string into current buffer
**insert-symbol**(3) Insert an ASCII character
**insert-tab**(2) (**C-i**) Insert tab(s) into current buffer
**insert-template**(3) Insert template file into current buffer
**ipipe-kill**(2) Kill a incremental pipe
**ipipe-shell-command**(2) (**esc backslash**) Incremental pipe (non-suspending system call)
**ipipe-write**(2) Write a string to an incremental pipe
**isearch-backward**(2) (**C-r**) Search backwards incrementally (interactive)
**isearch-forward**(2) (**C-s**) Search forward incrementally (interactive)
**ishell**(3) Open an interactive shell window
**iso-accents-mode**(3) ISO accent expansion short-cut mode
**item-list**(3) (**F7**) Abbreviated search and list buffer contents
**item-list-close**(3) (**esc F7**) Close the item list
**item-list-find**(3) Find the selected item in the item list
**jav**(9) Java programming language
**java**(9) Java programming language
**javascript**(9) Javascript programming language
**javatags**(3f) Generate a C tags file from Java sources
**jl**(9) Julia File
**js**(9) Javascript programming language
**jscript**(9) Javascript programming language
**jst**(9) JASSPA Structured Text
**jstags**(3f) Generate a Javascript tags file
**julia**(9) Julia File
**justify**(2m) Justification Mode
**kbd-macro-query**(2) (**C-x q**) Query termination of keyboard macro
**keep-buffer**(3) Rename the current buffer
**keyNames**(2) Key Binding Names
**kill-line**(2) (**C-k**) Delete all characters to the end of the line
**kill-paragraph**(2) Delete a paragraph
**kill-rectangle**(2) (**esc C-w**) Delete a column of text
**kill-region**(2) (**C-w**) Delete all characters in the marked region
**kotlin**(9) Kotlin programming language
**ksh**(9) Korn shell file
**kt**(9) Kotlin programming language
**ktm**(9) Kotlin programming language
**kts**(9) Kotlin programming language
**l**(9) LEX programming language
**languageTemplates**(2) File Language Templates
**latex**(9) TeX Documentation
**lf**(2m) File's line feed style
**line-scheme**(3) Apply line annotation in current buffer
**list-buffers**(2) (**C-x C-b**) List all buffers and show their status
**list-commands**(2) (**C-h c**) List available commands
**list-registry**(2) Display the registry in a buffer
**list-variables**(2) (**C-h v**) List defined variables
**localeSupport**(2) Locale Support
**lock**(2m) Pipe cursor position lock
**login**(9) Shell user login file

**lower-case-region**(2) (**C-x C-l**) Lowercase a region (downcase)
**lower-case-word**(2) (**esc l**) Lowercase word (downcase)
**lua**(9) Lua File
**luatags**(3f) Generate a Lua tags file
**MacroArguments**(4) Arguments to macros
**MacroNumericArguments**(4) Numeric arguments to macros
**Mahjongg**(3) MicroEmacs version of the solitaire Mah Jongg game
**MainMenu**(3) The top main menu
**Match-It**(3) MicroEmacs version of the Match-It game
**MessageLineVariables**(4) Prompt the user for input on message line
**MetaFont**(9) MetaFont/MetaPost File
**Metris**(3) MicroEmacs version of the falling blocks game
**m**(9) GNU Octave / Matlab File
**m4**(9) M4 Macro Processor
**macroDebugging**(4) Macro Debugging - A Crash Course
**macroOverview**(4) MicroEmacs Macro Language at a Glance
**macroWriting**(4) Writing MicroEmacs Macros
**magic**(2m) Regular expression search
**mail**(3) Compose and send an email
**mail-check**(3) Check for new email
**mail-setup**(3) Email setup dialog
**major-mode**(2) Declare or set a buffer major mode
**major-mode-help**(3) Displays help page for current buffer
**major-mode-setup**(3) Configures the current buffer settings
**major-mode-tools-setup**(3) File major-mode tool setup
**makefile**(9) Make file
**man**(3) UNIX manual page viewer
**man**(9) UNIX Manual Page
**man-clean**(3) Clean UNIX manual page
**mark-registry**(2) Modify the operating mode of a registry node
**markdown**(9) Markdown Markup File
**matlab**(9) GNU Octave / Matlab File
**md**(9) Markdown Markup File
**mdview**(3)
**mdview**(3f) Browse Markdown files in MicroEmacs.
**me**(1) MicroEmacs 2026 text editor
**mf**(9) MetaFont File
**ml-bind-key**(2) Create key binding for message line
**ml-unbind-key**(2) Remove key binding from message line
**ml-write**(2) Write message on message line
**mp**(9) MetaPost File
**nact**(2m) Buffer not active
**name-kbd-macro**(2) Assign a name to the last keyboard macro
**named-buffer-mode**(2) Change a named buffer mode
**narrow**(2m) Buffer contains a narrow
**narrow-buffer**(2) Hide buffer lines
**narrow-search-backward**(3) (**C-c r**) Search for a string in the backward direction
**narrow-search-forward**(3) (**C-c s**) Search for a string in the forward direction
**nawk**(9) New AWK File

**ne**(1) NanoEmacs 2026 text editor
**newline**(2) (**return**) Insert a new line
**next-buffer**(2) (**C-x x**) Switch to the next buffer
**next-frame**(2) (**C-x 5 o**) Change the focus to the next frame
**next-window**(2) (**C-x o**) Move the cursor to the next window
**next-window-find-buffer**(2) [] Split the current window and show new buffer
**next-window-find-file**(2) (**C-x 4**) Split the current window and find file
**nim**(9) Nim programming language
**normal-tab**(3) Insert a normal tab
**notes**(3) Tagged hierarchical database of notes
**notes-close**(3) Close the notes window
**notes-context**(3) Create/Access buffer context note
**notes-file**(3) Open a notes database file
**notes-import-skwyrul**(3) Import a Skwyrul index file (skwyrul_l.htm)
**notes-instant**(3) Create quick note
**nroff**(9) UNIX nroff file
**ntags**(3f) Generate a nroff tags file
**number-items**(3) Renumber a vertical list of items
**occur**(3) Regular expression search for occurrences
**octave**(9) GNU Octave / Matlab File
**open-line**(3) (**C-o**) Open new line at cursor position
**organizer**(3) Calendar and address organizer
**osd**(2) Manage the On-Screen Display
**osd-bind-key**(2) Create key binding for OSD dialog
**osd-dialog**(3) OSD dialog box
**osd-entry**(3) OSD entry dialog box
**osd-help**(3) GUI based on-line help
**osd-search**(3) Search and annotate the current buffer
**osd-unbind-key**(2) Remove key binding from OSD dialog
**osd-xdialog**(3) OSD Extended dialog box
**over**(2m) Over-strike Mode
**Patching**(1) Patch a MicroEmacs Installation
**Patience**(3) MicroEmacs version of Patience (or Solitaire)
**PowerShell**(9) PowerShell Programming Language
**p**(9) Pascal File
**page-bfile**(3) Load a large binary file as binary data
**page-file**(3) Load a large file in page mode
**page-file-backward-line**(3) (**C-p**) Move backwards one line in paged buffer
**page-file-beginning-of-buffer**(3) Move to the start of the paged file
**page-file-end-of-buffer**(3) Move to the end of the paged file
**page-file-forward-line**(3) (**C-n**) Move forwards one line in paged buffer
**page-file-goto-byte**(3) Page to the buffer byte offset
**page-file-goto-page**(3) Move to page
**page-file-isearch-backward**(3) (**C-r**) Page search backwards incrementally (interactive)
**page-file-isearch-forward**(3) (**C-s**) Page search forward incrementally (interactive)
**page-file-save-buffer**(3) (**C-x C-s**) Save contents of a page mode buffer to file
**page-file-scroll-down**(3) (**C-n**) Scroll paged buffer down
**page-file-scroll-up**(3) (**C-p**) Scroll paged buffer up
**page-file-search-backward**(3) (**C-x r**) Page search for a string in the backward direction

**page-file-search-buffer**(3) Search the page file buffer
**page-file-search-forward**(3) (**C-x s**) Page search for a string in the forward direction
**paragraph-to-line**(3) Convert a paragraph to a single line
**pas**(9) Pascal File
**password-to-phonic**(3) Convert a password to phonic string
**perl**(9) Practical Extraction and Report Language File
**perldb**(3) Perl Debugger
**perltags**(3f) Generate a Perl tags file
**php**(9) PHP: Hypertext Preprocessor Language
**php3**(9) PHP: Hypertext Preprocessor Language
**php4**(9) PHP: Hypertext Preprocessor Language
**phphtml**(9) PHP: Hypertext Preprocessor Language
**phptags**(3f) Generate a PHP tags file
**pipe**(2m) Incremental Pipe running
**pipe-shell-command**(2) (**esc @**) Execute a single operating system command
**pl**(9) Practical Extraction and Report Language File
**plantuml**(9) PlantUML Diagram Language File
**pm**(9) Practical Extraction and Report Language File
**pml**(9) PlantUML Diagram Language File
**popup-window**(2) Pop-up a window on the screen
**prefix**(2) Key prefix command
**previous-window**(2) (**C-x p**) Move the cursor to the previous window
**print-buffer**(2) Print buffer, with formatting
**print-color**(2) Create a new printer color
**print-region**(2) Print region, with formatting
**print-scheme**(2) Create a new printer color and font scheme
**print-setup**(3) Configure MicroEmacs's printer interface
**printall**(3f) Formatted print job
**profile**(9) Shell user profile
**ps1**(9) PowerShell File
**py**(9) Python Language File
**py-doc**(3) Display Python manual page
**pydoc**(3f) Display Python nodule documentation in MicroEmacs.
**python**(9) Python Language File
**qd**(9) Quarkdown Markup File
**quarkdown**(9) Quarkdown Markup File
**query-replace-all-string**(3) Query replace string in a list of files
**query-replace-string**(2) (**esc C-r**) Search and replace a string - with query
**query-replace-string-in-files**(3) Conditionally search and replace strings across files
**quick-exit**(2) (**esc z**) Exit the editor writing changes
**quote-char**(2) (**C-q**) Insert literal character
**RegisterVariables**(4) Register variables
**RegularExpressions**(2) Regular Expressions
**r**(9) R Language File
**r-doc**(3) R documentation manual page viewer
**rb**(9) Ruby programming language
**rbin**(2m) Reduced binary editor mode
**rc**(9) Microsoft Developer resource file
**rcs-file**(2) (**C-x C-q**) Handle Revision Control System (RCS) files

**rdiff**(3) Recursively compare files or directories
**rdoc**(3f) Display R help pages in MicroEmacs.
**read-file**(2) (**C-x C-r**) Find and load file replacing current buffer
**read-registry**(2) Read in a registry definition file
**read-session**(3) Read in session information
**recenter**(2) (**C-l**) Recenter the window (refresh the screen)
**recompile**(3) Rerun compilation process
**reg**(9) Registry file
**replace-all-pairs**(3) Replace string pairs in a list of files
**replace-all-string**(3) Replace string with new string in a list of files
**replace-string**(2) (**esc r**) Replace string with new string
**replace-string-in-files**(3) Search and replace strings across files
**reread-all**(3) Reload all files
**reread-file**(3) Reload the current buffer's file
**resize-all-windows**(2) Resize all windows (automatic change)
**resize-window-horizontally**(2) Resize current window horizontally (absolute)
**resize-window-vertically**(2) Resize the current window (absolute change)
**restore-dictionary**(3) Save dictionary user changes
**restyle-buffer**(3) Automatically reformat a buffer's indentation
**restyle-region**(3) Automatically reformat a regions indentation
**reyank**(2) (**esc y**) Restore next yank buffer
**rgrep**(3) Execute recursive grep command
**rgy**(9) Registry file
**rmarkdown**(9) R-Markdown Markup File
**rmd**(9) R-Markdown Markup File
**rs**(9) Rust programming language
**ruby**(9) Ruby programming language
**rul**(9) Install Shield Rules
**rust**(9) Rust programming language
**SearchGroups**(4) Last search group values
**s**(9) Assembler File
**samba**(9) Samba smb.conf configuration file
**save-all**(3) Save all modified files (with query)
**save-buffer**(2) (**C-x C-s**) Save contents of changed buffer to file
**save-buffers-exit-emacs**(2) (**esc z**) Exit the editor prompt user to write changes
**save-dictionary**(2) Save changed spelling dictionaries
**save-registry**(2) Write a registry definition file
**save-session**(3) Write session information to session file
**save-some-buffers**(2) Save contents of all changed buffers to file (with query)
**scala**(9) Scala programming language
**sch**(9) Scheme File
**scheme**(9) Scheme File
**scheme-editor**(3) Color Scheme Editor
**scm**(9) Scheme File
**screen-poke**(2) Immediate write string to the screen
**screen-update**(2) (**redraw**) Force screen update
**scroll-down**(2) (**C-n**) Move the window down (scrolling)
**scroll-left**(2) (**C-x <**) Move the window left (scrolling)
**scroll-next-window-down**(2) (**esc C-v**) Scroll next window down

**scroll-next-window-up**(2) (**esc C-z**) Scroll next window up
**scroll-right**(2) (**C-x >**) Move the window right (scrolling)
**scroll-up**(2) (**C-p**) Move the window up (scrolling)
**search-backward**(2) (**C-x r**) Search for a string in the backward direction
**search-buffer**(2) Search the current buffer for a string
**search-forward**(2) (**C-x s**) Search for a string in the forward direction
**session-setup**(3) Configure the data written to a session file
**set-alpha-mark**(2) (**C-x C-a**) Place an alphabetic marker in the buffer
**set-char-mask**(2) Set character word mask
**set-cursor-to-mouse**(2) Move the cursor to the current mouse position
**set-encryption-key**(2) (**esc e**) Define the encryption key
**set-mark**(2) (**esc space**) Set starting point of region
**set-position**(2) Store the current position
**set-registry**(2) Modify a node value in the registry
**set-scroll-with-mouse**(2) Scroll the window with the mouse
**set-title**(3) Set frame title prefix
**set-variable**(2) (**C-x v**) Assign a new value to a variable
**set-window**(2) Save the current window for restore (historic)
**sh**(9) Bourne shell file
**shell**(2) [**C-x c**] Create a new command processor or shell
**shell-command**(2) Perform an operating system command
**shell-tool-setup**(3) Define shell tools
**show-region**(2) Show the current copy region
**shrink-window-horizontally**(2) Shrink current window horizontally (relative)
**shrink-window-vertically**(2) Shrink the current window (relative change)
**shut-down**(3) Editor exit callback command
**smb.conf**(9) Samba configuration file
**so**(9) UNIX t/nroff include file
**sort-lines**(2) Alphabetically sort lines
**sort-lines-ignore-case**(3) Alphabetically sort lines ignoring case
**space-rectangle**(3) Adds space(s) vertically
**spell**(2) Spell checker service provider
**spell-add-word**(3) Add a word to the main dictionary
**spell-buffer**(3) Spell check the current buffer
**spell-edit-word**(3) Edits a spell word entry
**spell-word**(3) (**esc $**) Spell check a single word
**split-window-horizontally**(2) (**C-x 3**) Split current window into two (horizontally)
**split-window-vertically**(2) (**C-x 2**) Split the current window into two
**sql**(9) SQL File
**sql-format**(3) SQL language, format SQL query
**start-kbd-macro**(2) (**C-x (**) Start recording keyboard macro
**start-up**(3) Editor startup callback command
**stop-mail-check**(3) Disable the check for new email
**str-to-regex**(3) Generate regex safe variant of a string
**string-rectangle**(3) Add a string vertically
**sty**(9) TeX Style File
**suspend-emacs**(2) Suspend editor and place in background
**svn**(3) MicroEmacs Subversion interface
**svn-add**(3) MicroEmacs SVN interface - add items

**svn-blame**(3) MicroEmacs SVN interface - annotate a file
**svn-cat-file**(3) MicroEmacs SVN interface - output specific version of a file
**svn-checkout**(3) MicroEmacs SVN interface - checkout a working copy
**svn-commit**(3) MicroEmacs SVN interface - commit changes back
**svn-copy**(3) MicroEmacs SVN interface - copy items
**svn-delete**(3) MicroEmacs SVN interface - delete items
**svn-diff**(3) MicroEmacs SVN interface - diff changes
**svn-gdiff**(3) MicroEmacs SVN interface - graphical diff changes
**svn-ignore**(3) MicroEmacs SVN interface - edit directory item ignore list
**svn-info**(3) MicroEmacs SVN interface - display info about an item
**svn-log-commit**(3) MicroEmacs SVN interface - get log for a commit
**svn-log-file**(3) MicroEmacs SVN interface - list log of changes
**svn-move**(3) MicroEmacs SVN interface - move an item
**svn-resolve**(3) MicroEmacs SVN interface - resolve conflicts
**svn-revert**(3) MicroEmacs SVN interface - revert local changes
**svn-server-copy**(3) MicroEmacs SVN interface - server based copy items
**svn-server-move**(3) MicroEmacs SVN interface - server based move items
**svn-setup**(3) MicroEmacs SVN interface setup
**svn-status**(3) MicroEmacs SVN interface - status of items
**svn-update**(3) MicroEmacs SVN interface - update directory files
**svn-xdiff**(3) MicroEmacs SVN interface - External diff changes
**swift**(9) Swift programming language
**Triangle**(3) MicroEmacs version of Triangle patience game
**tab**(2) (**tab**) Handle the tab key
**tab**(2m) Tabulation mode
**tabs-to-spaces**(3) Converts all tabs to spaces
**tcl**(9) TCL programming language
**tcltags**(3f) Generate a Tcl/Tk tags file
**tcshrc**(9) T-Shell start up file
**tex**(9) TeX Documentation
**tex2nr**(3) Convert a Latex file into nroff
**texi**(9) GNU Texinfo documentation file
**texinfo**(9) GNU Texinfo documentation file
**textags**(3f) Generate a LaTeX/BibTeX tags file
**time**(2m) File time stamping
**time**(3) Command time evaluator
**tk**(9) TK programming language
**tman**(9) Tcl Doctools Markup File
**tni**(9) UNIX t/nroff include file
**toml**(9) Tom's obvious minimal language file
**toolbar**(3) Toggle tool bar visibility
**toolbar-close**(3) Close the tool bar
**toolbar-open**(3) Open the tool bar
**toolbar-refresh**(3) Update the tool bar
**translate-buffer**(2) Translate buffer from one format to another
**translate-key**(2) Translate key
**transpose-chars**(2) (**C-t**) Exchange (swap) adjacent characters
**transpose-lines**(2) (**C-x C-t**) Exchange (swap) adjacent lines
**transpose-region-line**(3) Exchange (swap) region with adjacent line

**troff**(9) UNIX troff file
**ts**(9) Typescript programming language
**txt**(9) ASCII plain text file
**typ**(9) Typst Markup File
**typescript**(9) Typescript programming language
**typst**(9) Typst Markup File
**UserProfiles**(2) Defining a user profile
**uncomment-line**(3) Uncomment current line
**undo**(2) (**C-x u**) Undo the last edit
**undo**(2m) Retain edit modifications
**uniq**(3) Make lines in a sorted list unique
**universal-argument**(2) (**C-u**) Set the command argument count
**unix2dos**(3f) Convert UNIX format files to DOS format files.
**unix2win**(3f) Convert UNIX format files to Windows format files.
**unmark-buffer**(3) Remove buffer edited flag
**unset-variable**(2) Delete a variable
**upper-case-region**(2) (**C-x C-u**) Uppercase a region (upcase)
**upper-case-word**(2) (**esc u**) Uppercase word (upcase)
**user-setup**(3) Configure MicroEmacs for a specific user
**Variables**(4) User defined macro variables
**v**(9) Verilog hardware description language
**vala**(9) Vala programming language
**vb**(9) Visual Basic
**vbtags**(3f) Generate a Visual Basic tags file
**vcs**(3) MicroEmacs VCS interface - VCS-agnostic front-end
**vcs-add**(3) MicroEmacs VCS interface - add items
**vcs-annotate**(3) MicroEmacs VCS interface - annotate a file
**vcs-blame**(3) MicroEmacs VCS interface - annotate a file
**vcs-cat-file**(3) MicroEmacs VCS interface - output a specific revision/commit of a file
**vcs-checkout**(3) MicroEmacs VCS interface - checkout (create) a working copy
**vcs-coci**(3) MicroEmacs VCS interface - check-out/check-in a file
**vcs-commit**(3) MicroEmacs VCS interface - commit changes to repository
**vcs-delete**(3) MicroEmacs VCS interface - delete/remove items
**vcs-diff**(3) MicroEmacs VCS interface - diff changes
**vcs-fetch**(3) MicroEmacs VCS interface - download from another repository
**vcs-gdiff**(3) MicroEmacs VCS interface - graphical diff changes
**vcs-id**(3) MicroEmacs VCS interface - identify the VCS controlling a file/directory
**vcs-log-commit**(3) MicroEmacs VCS interface - show log for a commit
**vcs-log-file**(3) MicroEmacs VCS interface - list log of changes
**vcs-move**(3) MicroEmacs VCS interface - move an item
**vcs-pull**(3) MicroEmacs VCS interface - download and integrate another repository
**vcs-push**(3) MicroEmacs VCS interface - update remote repository
**vcs-remove**(3) MicroEmacs VCS interface - remove items
**vcs-resolve**(3) MicroEmacs VCS interface - resolve conflicts
**vcs-revert**(3) MicroEmacs VCS interface - revert local changes
**vcs-setup**(3) MicroEmacs VCS interface - setup the underlying VCS
**vcs-status**(3) MicroEmacs VCS interface - status of items
**vcs-switch**(3) MicroEmacs VCS interface - switch or create branches
**vcs-unstage**(3) MicroEmacs VCS interface - unstage items

**vcs-update**(3) MicroEmacs VCS interface - update directory files
**verilog**(9) Verilog hardware description language
**vhdl**(9) VHDL hardware simulation File
**view**(2m) Read only
**view-file**(2) (**C-x C-v**) Load a file read only
**vlang**(9) V programming language
**vm**(3) Email viewer
**void**(2) Null command
**vrml**(9) VRML File
**watch**(3) Monitor a changing file
**watch-reset**(3) Reset the watch command counters
**watch-time**(3) Change the watch command poll time
**which**(3) Program finder
**wiki**(9) GNU Wiki File
**wish**(9) TCL shell file
**wrap**(2m) Line wrap entered text
**wrap-word**(2) Wrap word onto next line
**write-buffer**(2) (**C-x C-w**) Write contents of buffer to named (new) file
**write-region**(3) Write contents of region to named (new) file
**x86**(9) Intel .x86 Assembler File
**xdiff**(3) External graphical file difference
**xfind**(3) Execute find command
**xgrep**(3) Execute grep command
**xlate**(2m) Translated buffer editor mode
**xml**(9) Extensible Markup Language
**xml-backward-tag**(3) Find the previous XML tag
**xml-forward-tag**(3) Find the next XML tag
**xml-goto-matching-tag**(3) Find matching opening/closing XML tag
**xml-verify**(3) Verify syntax of XML file
**xrdb-scheme**(3) Load a XRDB scheme file from the user's config folder
**xsl**(9) Extensible Stylesheet Language
**xul**(9) XML User Interface Language
**y**(9) YACC programming language
**yank**(2) (**C-y**) Paste (copy) kill buffer contents into buffer
**yank-rectangle**(2) (**esc C-y**) Insert a column of text
**yank-rectangle-overwrite**(3) Insert a column of text with overwrite
**zfile-reset**(3) Compressed file support reset
**zfile-setup**(3) Compressed file support setup
**zig**(9) Zig programming language
**zsh**(9) Z-Shell file

## ACKNOWLEDGMENTS

The following persons contributed to this release of **MicroEmacs** over the last decade, roughly in the order of participation. This list represents the main developers:

Dave Conroy (Original Author)
Daniel M. Lawrence (Original Maintainer)
Martin House
**Jon Green** - *Current Maintainer*
Callen McNally
**Steven Phillips** - *Current Maintainer*
**Detlef Groth** - *Current Maintainer*

Additional contributions have been made as follows, in chrononlogical order:

Sebastian Bojers
: Finnish support.

Jeremy Cowgar
: Lua language template.
: X-Windows modifications.

Pedro Gomes
: Portuguese Dictionary.
: Cobol and Intel x86 language templates.
: Metapost/Meta Font templates.

Detlef Groth
: Setting up and validating the German environment.
: Latex features and excellent feedback.
: Document support.
: Fixed PERL abbreviations.

Mike Hopkirk
: Macro execution speed up suggestions.

Thomas Hundt
: Fixed $frame-width/depth macro operation on the set.
: Fix for XML macros.
: Fix for VM mail searching.

Matthew Robinson
: Developed the WinConsole version for Windows NT.

Salman Problem reporting and feed-back.

Peter Smith
: Code optimizations and extensions.

Gabriel Tabares-Barreiro
: ADA language template.

Thanks to everybody else that has used and abused it locally feeding back comments and preferences, wishes and desires.

### And Finally ....

Answers to those burning questions that people ask from time to time ...

### Where does the name JASSPA come from ?

The current maintainers initials mixed up, padded out to make words, short listed and searched on the WWW to find a unique name. This process took approximately two months.

### How was the JASSPA logo derived ?

The logo was derived from the program initials Me. Given the limited number of pixels available in an icon then the 'e' was superimposed in the center of the 'M'. With some further refinement the legs of the 'M' were later lost leaving a triangular shape - this was then adopted as the iconic outline. The colored lines at the top and bottom portray the visual appearance of a JASSPA's MicroEmacs from afar when viewing an X-term running me within an X-Windows environment. The blue line representing the window header color (that I use), the red line representing the MicroEmacs mode bar; the screen was typically black with white or yellow characters. The icon was incrementally refined over a period of approximately six months.

Ref: MicroEmacs - Acknowledgments File: me.0 Date: 2026/09/23

## COPYRIGHT

Copyright (c) 1988-2026 JASSPA (www.jasspa.com)

JASSPA's MicroEmacs is free software; you can redistribute it and/or modify it under the terms of the GNU GENERAL PUBLIC LICENSE (GPL) as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version. A copy of the GPL version 2 license may be found in this file.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

This license is valid from January 1st 2002 and applies to the JASSPA Distribution of MicroEmacs only. These license terms may not be applied to other versions of MicroEmacs or retrospectively to older versions of JASSPA MicroEmacs which fall under their own licensing terms.

Special thanks go to Danial M. Lawrence, the original author and license holder of MicroEMACS, who granted JASSPA permission to move the code base to GPL on 19th December 2001. Permission was granted on the grounds that code base is nearly 14 years diverged from the original and has been substantially modified.

Jon Green & Steve Philips (JASSPA - www.jasspa.com) 30th December 2001

### GNU General Public License (GPL)

```
                    GNU GENERAL PUBLIC LICENSE
                       Version 2, June 1991

 Copyright (C) 1989, 1991 Free Software Foundation, Inc.
    59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

                            Preamble

  The licenses for most software are designed to take away your
freedom to share and change it.  By contrast, the GNU General Public
License is intended to guarantee your freedom to share and change free
software--to make sure the software is free for all its users.  This
General Public License applies to most of the Free Software
Foundation's software and to any other program whose authors commit to
using it.  (Some other Free Software Foundation software is covered by
the GNU Library General Public License instead.)  You can apply it to
your programs, too.

  When we speak of free software, we are referring to freedom, not
price.  Our General Public Licenses are designed to make sure that you
have the freedom to distribute copies of free software (and charge for
this service if you wish), that you receive source code or can get it
if you want it, that you can change the software or use pieces of it
in new free programs; and that you know you can do these things.

  To protect your rights, we need to make restrictions that forbid
anyone to deny you these rights or to ask you to surrender the rights.
These restrictions translate to certain responsibilities for you if you
distribute copies of the software, or if you modify it.

  For example, if you distribute copies of such a program, whether
gratis or for a fee, you must give the recipients all the rights that
you have.  You must make sure that they, too, receive or can get the
source code.  And you must show them these terms so they know their
rights.
```

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

GNU GENERAL PUBLIC LICENSE
TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

0. This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under

these conditions, and telling the user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

4. You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

5. You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

6. Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

7. If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

8. If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

9. The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

10. If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

NO WARRANTY

11. BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

12. IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

```
    <one line to give the program's name and a brief idea of what it does.>
    Copyright (C) <year>  <name of author>

    This program is free software; you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation; either version 2 of the License, or
    (at your option) any later version.

    This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.

    You should have received a copy of the GNU General Public License
    along with this program; if not, write to the Free Software
    Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
```

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

```
    Gnomovision version 69, Copyright (C) year name of author
    Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type 'show w'.
    This is free software, and you are welcome to redistribute it
    under certain conditions; type 'show c' for details.
```

```
The hypothetical commands `show w' and `show c' should show the appropriate
parts of the General Public License.  Of course, the commands you use may
be called something other than `show w' and `show c'; they could even be
mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your
school, if any, to sign a "copyright disclaimer" for the program, if
necessary.  Here is a sample; alter the names:

  Yoyodyne, Inc., hereby disclaims all copyright interest in the program
  `Gnomovision' (which makes passes at compilers) written by James Hacker.

  <signature of Ty Coon>, 1 April 1989
  Ty Coon, President of Vice

This General Public License does not permit incorporating your program into
proprietary programs.  If your program is a subroutine library, you may
consider it more useful to permit linking proprietary applications with the
library.  If this is what you want to do, use the GNU Library General
Public License instead of this License.
```

## GNU Free Documentation License (GFDL)

All documentation is covered by the GFDL.

```
                GNU Free Documentation License
                   Version 1.1, March 2000

 Copyright (C) 2000  Free Software Foundation, Inc.
     59 Temple Place, Suite 330, Boston, MA  02111-1307  USA
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other
written document "free" in the sense of freedom: to assure everyone
the effective freedom to copy and redistribute it, with or without
modifying it, either commercially or noncommercially.  Secondarily,
this License preserves for the author and publisher a way to get
credit for their work, while not being considered responsible for
modifications made by others.

This License is a kind of "copyleft", which means that derivative
works of the document must themselves be free in the same sense.  It
complements the GNU General Public License, which is a copyleft
license designed for free software.

We have designed this License in order to use it for manuals for free
software, because free software needs free documentation: a free
program should come with manuals providing the same freedoms that the
software does.  But this License is not limited to software manuals;
it can be used for any textual work, regardless of subject matter or
whether it is published as a printed book.  We recommend this License
principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work that contains a
notice placed by the copyright holder saying it can be distributed
under the terms of this License.  The "Document", below, refers to any
such manual or work.  Any member of the public is a licensee, and is
addressed as "you".

A "Modified Version" of the Document means any work containing the
Document or a portion of it, either copied verbatim, or with
modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of
```

the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (For example, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, whose contents can be viewed and edited directly and straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup has been designed to thwart or discourage subsequent modification by readers is not Transparent. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML designed for human modification. Opaque formats include PostScript, PDF, proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies of the Document numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present

the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a publicly-accessible computer-network location containing a complete Transparent copy of the Document, free of added material, which the general network-using public has access to download anonymously at no charge using public-standard network protocols. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has less than five).
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document’s license notice.
H. Include an unaltered copy of this License.
I. Preserve the section entitled "History", and its title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at

least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. In any section entitled "Acknowledgements" or "Dedications", preserve the section's title, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section as "Endorsements" or to conflict in title with any Invariant Section.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties--for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections entitled "History" in the various original documents, forming one section entitled "History"; likewise combine any sections entitled "Acknowledgements", and any sections entitled "Dedications". You must delete all sections entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in

the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, does not as a whole count as a Modified Version of the Document, provided no compilation copyright is claimed for the compilation. Such a compilation is called an "aggregate", and this License does not apply to the other self-contained works thus compiled with the Document, on account of their being thus compiled, if they are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one quarter of the entire aggregate, the Document's Cover Texts may be placed on covers that surround only the Document within the aggregate. Otherwise they must appear on covers around the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License provided that you also include the original English version of this License. In case of a disagreement between the translation and the original English version of this License, the original English version will prevail.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
      Copyright (c)  YEAR  YOUR NAME.
      Permission is granted to copy, distribute and/or modify this document
      under the terms of the GNU Free Documentation License, Version 1.1
      or any later version published by the Free Software Foundation;
      with the Invariant Sections being LIST THEIR TITLES, with the
      Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
      A copy of the license is included in the section entitled "GNU
      Free Documentation License".

If you have no Invariant Sections, write "with no Invariant Sections"
instead of saying which ones are invariant.  If you have no
Front-Cover Texts, write "no Front-Cover Texts" instead of
"Front-Cover Texts being LIST"; likewise for Back-Cover Texts.

If your document contains nontrivial examples of program code, we
recommend releasing these examples in parallel under your choice of
free software license, such as the GNU General Public License,
to permit their use in free software.
```

### License History

JASSPA MicroEmacs is derived from the MicroEmacs 3.8 source base of 1998. As such, all of the software has been under a commercially restrictive license. JASSPA has upheld the original license terms laid down my the original author and copyright holder Daniel M Lawrence.

JASSPA is the collective name given to the maintainers of JASSPA MicroEmacs. The current maintainers at the 1st January 2002 are Steven Phillips and Jon Green.

On Wednesday 19th January 2001 JASSPA applied for, and was granted, permission by Daniel M Lawrence to move to the less restrictive licensing terms of GPL. As of 1st January 2002 the JASSPA distribution of MicroEmacs shall be supplied under this licensing arrangement. The licence is not transferable to earlier versions of the distribution or to the original program from which is was derived known as MicroEmacs.

### License Terms to 1988

The following copyrights apply from the original source code of version 3.8. No explicit copyrights were found with the original distribution apart from the following found in the main source code,

> **(C)opyright 1987 by Daniel M. Lawrence**
> MicroEMACS can be copied and distributed freely for any non-commercial purposes. Commercial users may use MicroEMACS inhouse. Shareware distributors may redistribute MicroEMACS for media costs only. MicroEMACS can only be incorporated into commercial software or resold with the permission of the current author.

### License Terms 1998-2001

The following notices apply after 1988 to 31st December 2001

> **Copyright (C) 1988 - 2001, JASSPA**
> JASSPA MicroEmacs can be copied and distributed freely for any non-commercial purposes. Commercial users may use JASSPA MicroEmacs inhouse. Shareware distributors may redistribute JASSPA MicroEmacs for media costs only. JASSPA MicroEmacs can only be incorporated into commercial software or resold with the permission of the current author.

### License Terms 2002 and subsequent years

GNU Public License (GPL) for all source material. GNU Free Documentation License (GFDL) for all documentation material.

Ref: MicroEmacs - Copyright File: me.0 Date: 2026/09/23

### Spelling Dictionary Copyrights

The spelling dictionaries are converted from *ispell* dictionaries, each spelling dictionary has it's own copyright which is reproduced within the appropriate language spelling macro file.

### NO WARRANTY

THIS PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

**THIS NOTICE MUST BE CARRIED IN ALL COPIES OF THE DISTRIBUTION**

### CONTACT INFORMATION

Use GitHub to report any issues or to make suggestions.

```
https://github.com/bjasspa/jasspa/
```

Ref: MicroEmacs - Contact Information File: me.0 Date: 2026/09/23

## HISTORY

This version of MicroEmacs is based on an early *MicroEmacs* release of 3.8 in 1988, the origins of which are unknown, except to say it was delivered on a unmarked 5 1/2" floppy disk.

The program was originally ported to a Motorola MVME147 UNIX box as an alternative to **vi**. Reliability of the program proved to be a problem as it constantly crashed. In an attempt to rectify the problems the development of MicroEmacs commenced.

Development has continued from 1988 through to today, on the whole oblivious to further developments of the existing **MicroEmacs** program. This was due to no Internet access. It was not until 1996 the next version of **MicroEmacs** and **mewin** (Microsoft Windows (TM) port of the same program) was downloaded from the Internet and compared. By this time MicroEmacs was radically different and we were not about to mesh the two together - that would be a step backwards.

Development of MicroEmacs has been biased towards the UNIX platform, as most of the early development was performed in the UNIX domain. The first of the window servers was X-Windows, which in turn has shaped the implementation of the Microsoft Windows port. Latterly, we have seen the resurgence of the IBM-PC platform which is now commonplace. For the return port to the DOS environment, and subsequent development of the Microsoft Windows port, a UNIX like interface was required. Most existing users could not abide the primitive editors found on these machines; Microsoft Windows was an alien and hostile environment when compared with UNIX. Hence, the MicroEmacs interface utilizes UNIX style cut and paste across all platforms.

For portability, MicroEmacs utilizes character rendering on all platforms regardless of the window manager. Under X-Windows and Microsoft Windows, the display is still treated as a character based display, the subtle difference is that the display pane is re-sizable. This means that the scroll bars, fonts etc. are not as slick as they could be, certainly under Microsoft Windows MicroEmacs looks positively primitive!! Regardless of the look, the goal of a common editor across all working platforms has been achieved!

### Development History

**Release - 1st September 2026**
Release date specified as 2026-09-01.

- New **md-view** command and **@MDView** command-line interface.
- Improvements to UNIX terminal mouse support.
- Fixed issues with moving files across different drives.
- Fixed ipipe'd command dropping important control characters.
- Several other minor fixes and improvements.

**Release - 1st August 2026**
Release date specified as 2026-08-01.

- New Arkanoid, Balls and Hearts games.
- Added support for RGB colours in ipiped terminals.
- Many improvements and fixes to ipipe terminal support on all platforms.

- Unified several VCS commands and improved vcs-* wrapper.
- Added bit flag 2 to **save-some-buffers** to stop prompting of buffers with no file.
- Fixed issues with Cygwin and MSYS installer script.
- Changed method of hiding cursor to a buffer mode.
- Renamed **keep-next-buffer** to **keep-buffer** and made more general.
- Added support for **-1** copy-region in Unix terminal via OSC52.

**Release - 1st July 2026**
Release date specified as 2026-07-01.

- Added **vcs** wrapper for all supported VCS1's.
- Auto-convert ipipe shell command UFT-8 output to MicroEmacs charset.
- Added support for Windows ConPTY pseudo terminal.
- Added base support for ipipe colours via meth format.
- Fixed numerous ipipe PTY issues.

**Release - 1st June 2026**
Release date specified as 2026-06-01.

- New native MacOS GUI (no dependency on XQuartz).
- Added Windows Cygwin platform to release.
- Added simple update available check to main Help menu.
- Fixed and standardised multi-window message-line handling.
- Improve Unix install/update script to handle non-standard directories.

**Release - 1st May 2026**
Release date specified as 2026-05-01.

- Added case sensitive order option to directory major mode and made default case-insensitive on all platforms.
- Improved describe-word including use of OSD dialog interface for spelling.
- Fixed crash caused by handling of absolute pathed symbolic links.
- Fixed Unix and Cygwin group file permission issues.
- Fixed X window sizing issue.
- Several other minor fixes and optimisations.

**Release - 1st April 2026**
Release date specified as 2026-04-01.

**Note**: Support for homebrew based installation has been dropped, please uninstall the homebrew version and use one of the other install options, preferably the `microemacs-install.sh` script.

- Improved installer scripts method of identifying current release.
- Changed **file-exec**, **file-format** and **file-lint** key bindings to avoid conflict.
- Fixed missing Windows MSYS setup macro file and tags for Lua.
- Fixed conflicts with *set-char-mask* 1,2,3 and 4 word classes.

**Release - 1st March 2026**
Release date specified as 2026-03-01.

- New language support for Erlang and Haxe.
- Added support for SVG to XML major-mode.
- Improvements to Fusion major-mode.
- Improvements to describe-word, including support for `dict:// URLs`.
- Improved embedding of **diff-lines** in diff **major-mode** buffers.
- Improved **file-exec** and added support for executing SQL via MySql/MariaDB and SQL Server.
- Fixed build conflicts between *termcap* and *Xft* colors and other minor build issues.

**Release - 1st February 2026**
Release date specified as 2026-02-01.

- Updated support for platform Solaris x86
- Changed installer scripts to fix issue #26 which failed to obtain the version number of latest release.
- Fixed issues building NanoEmacs (ne).
- Improvements to regex replace string format.
- Minor fixes to MD, python, fusion and kotlin file support.
- Clean up format issues in documentation.

**Release - 1st January 2026**
Release date specified as 2026-01-01.
**Note**: For this release, any calls to **add-file-hook** must be migrated. See **major-mode**(2) documentation. It is recommended to run **zfile-reset** if you have used **find-zfile**.

- New major-mode command.
- New languages PlantUML, Groovy, Fusion, Typescript and Scala.
- Improvement and fixes in Markdown, Python (py-doc!), TCL and C#.
- Improved Info including `me[cw] @info ...` command-line support.
- Improved **find-zfile** to support single-compressed files.

- Changed Windows rendering to use *WChar* functions.
- Added base support for `dict://...` URLs.

**Release - 1st December 2025**
Release date specified as 2025-12-01.

- Added build option **-dasan** (ASAN support) for Microsoft VC.
- Fixed several memory issues particularly with re-sizing dialogs.
- Added automatic mapping of `\x7f` to backspace for UNIX consoles.
- Added support for http keep-alive timeout.
- Implemented git-setup and documented GIT interface.
- Improved generation of tags.
- Other minor fixes and improvements.

**Release - 1st November 2025**
Release date specified as 2025-11-01.

- Added support for accented Letter input on Unix X platforms with 'dead key' keyboards.
- Change Windows console to use Unicode output with extended char set support.
- Support for Haskell language.
- Support for Tcl manual pages markup
- Improved MacOS application launcher, setting user environment correctly.
- Support for ISO-8859-16 encoding.
- Fixed install issues with MacOS apple chips (M1 thru M4).
- Fixed user-setup font selection dialog issues on Unix X.

**Patch Release - 11th October 2009**
Release date specified as 2009-10-11.

- [091011] Patch release of fixes; release date specified as 2009/10/11.
- [091011] Fixed kill-rectangle scribble caused by hilight hidden characters overrunning the kill buffer.
- [091011] Fixed debug build error caused by narrow udata change. Changed debug fprintf of pointers to use %p to support 64-bit version.
- [091010] Fix for &rep and &irep truncating the resultant string prematurely.
- [091010] Fixes for 64-bit system compilation. Issues with spelling pointer arithmetic and undo structure alignment.

**Patch Release - 3rd October 2009**
Release date specified as 2009-10-03.

- [091003] Fixed documentation for crlf(2m) which is replaced by cr(2m) and lf(2m).
- [091003] Fixed documentation for quiet(2m) which has been removed and replaced with quiet(5).
- [091003] Bound cross note searching to 'esc s' and esc C-s' and removed the 's' & 'r' tree buffer binding. Implemented execution of cross note search within cross note search to jump to the next note. Changed note rename binding to 'r'
- [090930] Fixed crash caused by encrypt key being set to '' (i.e. disabled) on a reload, resetkey did not check this so crashed trying to use NULL.
- [090930] notes; fixed prompt of encryption key when the notes file is already loaded.

**Patch Release - September 2009**
Release date specified as 2009-09-26.

- [090926] Fixed Linux X-Window size issue.
- [090923] Fixed bad characters in Clearcase output window
- [090917] Fixed Linux makefiles for GCC 4.3 for the detection of ncurses.

**10th Release - September 2009**
Release date specified as 2009-09-09.

- [090903] **file-browser**(3) improved on Microsoft Windows to perform a case change i.e. "test.txt" to "Test.txt".
- [090903] Added **encase**(3) to encase a region with a starting and ending string.
- [090903] Added **number-items**(3) to perform renumbering on a vertical aligned list of numeric digits.
- [090901] Added template for Inno Setup Script files with extension .iss
- [090831] Changed buffer major setup so all major modes may enable abbreviations and templates. The user may create the appropriate file if there is no standard file supplied and it will be enacted.
- [090831] HTML template; indent  constructs.
- [090831] notes; maintain the exact and magic mode settings during a notes tree update.
- [090805] Added support for \i#, \u# & \c# in &xre macro function
- [090530] Fixed TCL abbreviation file expansion reported by Ferenc Deak.
- [090523] Fix fill-paragraph so standard 'o' and 'a' are supported.
- [090523] Fix jst paragraph fill so 'o' and 'a' are supported
- [090523] Bound "esc-'" to enter a very large arg n (99999999)

- [090213] Improved indent rules so two 'current-line and onward' tokens together are both considered on the current line, i.e. '</tr></table>' is a double indent.
- [090201] Fixed the 'No to all' 'o' option to fill paragraph prompt, was still 'v' for Never.
- [090201] Avoided crash caused by moving reg node to itself or one of its children.
- [090121] Better support for encrypted notes files, checks the first line and if not '-!-enf...' it assumes its encrypted and will prompt for a password.
- [081204] Fixed get-next-line buffer selection (logic inverted).
- [081204] Fixed file-tool-menu so it will work in a *find* buffer.
- [081116] Upped the macro minimum version to 2008-11-10
- [081111] Added 'Disable' to file-tool-setup, changed 'Default Open' to 'Open File' and added 'View File'
- [081111] Changed the default behaviour of open and view to open a menu if more than one Open/View tool matches.
- [081111] Added alias paths to a ' ' auto-completion list.
- [081111] Fixed compile buffer processor often missing error lines.
- [081106] Changed file tool mask so if it starts with '^...' the whole file name is matched not just the base name (required for 'http://...' type matching).
- [081106] Remove file tool setup from the context menu as it is now in the file tool menu itself.
- [081106] Document highlighting; improved the hilighting of http & https urls and added file: hilighting.
- [081106] Added a new 'Find Link' mouse command (use user-setup to bind) which will open the file/url clicked on by the mouse (typically bind to Ctrl Left mouse)
- [081106] Added 'File Tool Setup' to the file tool context menu.
- [081026] Added confirmation prompt if about to save a file which had binary chars or inconsistent line endings.
- [081024] Fixed forward def's for file-tool macros.
- [081024] Added 'File Tools' item to Advanced menu.
- [081024] Enhanced file-tool to have the features found in shell-tools (namely ability to capture output and save buffers with(out) prompt)
- [081024] Improved the get-next-line file name template to be more strict with matching drive letters at the start.
- [081019] Improved behaviour when ftp reg file is missing.
- [081019] Ensured large files are handled properly (fails for **replace-string-in-files**(3)) and improved error reporting.

- [081018] Fixed **find-file**(2) so it can load files and directories with square brackets (i.e. a[b])
- [081010] Removed compile highlighting as this is now done differently.
- [081010] Changed favorite file opening to check for a default open file-tool and use that instead.
- [081010] Clearcase; changed ipipe launch of cleartool to make use of new no-wrap bit 0x200.
- [081005] Implemented a more macro orientated interface to get-next-line (when n == 0), which only looks in the current buffer, if a 'next line' is found it simply changes the cursor location, sets next-file & next-line variables and does not pop-open the file. Got get-next-line to set :next-line as well as a next-file variable.
- [081005] find and grep; used new '&bstat o' to check that any loaded buffer used (-l) is up-to-date and if not prompt the user.
- [081005] find and grep; introduced -a option to automatically reread all ood buffers (Yes to All) and -o option to never reread the ood buffers (NO to all)
- [081005] Changed shell-tool interface so a tool can run concurrent without going to a buffer (uses relatively recent improvements to shell-command)
- [081005] Changed reread-all command so default action only rereads out-of-date buffers (uses new &bstat) if 0 arg is given then it rereads all as before.
- [081005] Protected ME from potential crash when homedir is NULL.
- [081002] File types, fixed abort caused by empty files.
- [081002] page-file; implemented a new -u/-U option to limit the size of a file to be grep'd to 100Mb unless -u is given (unlimited, includes -w for find)
- [081002] When a large file is encountered find & grep now use page-file.
- [081002] Fixed pressing C-g at read-only prompt of replace-string-in-files to abort the command.
- [081002] page-file; Implemented an absolute line counter which works as long as the user started from the top (v useful for grep)
- [081002] page-file; Implemented a macro orientated page-file-search-buffer macro.
- [081002] page-file; Added support for specifying the buffer name to be used by page-file (0 numeric arg)
- [081002] isearch macros now have an 'Yes to All' option when asked whether to change page.
- [081002] Increased the default page size to 1Mb
- [081002] Restricted the session kill history to only buffers which are 1020 chars long or less.
- [081002] Added @y0 ... @y15 support to directly extract the kill-ring buffers.

- [081002] Added a new &bstat command variable which is similar to &stat but for the current buffer, currently only has 'o' for 'out-of-date' test.
- [081002] Made double click on a window mode-line delete other windows (i.e. maximise the window)
- [0801002] Fixed opening toolbar when notes was active in the session.
- [081002] tools; added new save-kbd-macro command which makes it easy to save a keyboard macro for future use.
- [081002] xfind; use new 0x200 ipipe flag to not wrap lines, avoids get-next-line failures on long file names.
- [080929] Changed the 0 & +ve args for reyank, now acts as a yank yanking out the nth item in the kill ring (i.e. 0 reyank == yank (except no external clip check))
- [080929] Fixed bug in isearch - C-w was not storing status so a following backspace could lose the Okay status.
- [080916] find/grep -& (background) option working with new -f ME option.
- [080916] Added Windows MSVC 8 makefile
- [080916] Fixed pipe failure problem caused by GetConsoleScreenBufferInfo failing (no console) this is a fatal problem whe not in piping mode - not checking for failure and handling led to use of uninitialized variables.
- [080724] Fixed spin in ctags caused by a double '; ;'
- [080724] Added new 'Tag File' option to the generate-tags dialog
- [080724] Added command line -f option to set .about.arg# variables
- [080720] Added the session name to the main window title
- [080720] New schemes contributed by BS.
- [080720] Implemented $frame-title to partially set the main window title text.
- [080720] Improved delete-some-buffers so bit 0x01 only disables the 'is modified' test, 0x02 disables the main prompt, added new 'delete inactive buffers' prompt with bit 0x04 to disable.
- [080509] Improved the hilighting of grep output.
- [080426] Fixed PS scheme (now called Lumina). Update of Charcoal scheme.
- [080330] Enhanced the javadoc highlighting.
- [080330] PHP; Corrected string case and enhanced the doxygen commenting.
- [080311] Fixed cvs-remove when used in a dir listing (only worked in a cvs-state buffer)
- [080222] Extended 'C' Doxygen highlighting.
- [080207] Added new 'touch' feature to file-op (bit 0x400)

- [071125] Fixed file name input crash caused by using completion in a dir with files being removed.
- [071125] C, directory list and latex; Fixed 'o' (no to all) response to copy which did not work.
- [071125] Fixed clearcase -mkbranch files being given '?' status.
  Fixed cc-merge refresh bug.
- [071103] Build; added -D option to specify a define, e.g. -D _USETPARM to use the systems tparm function rather than ME's
- [070311] 2dos, 2mac, 2unix, 2win; use arg 0 as these are not able to prompt from the user if the file is read-only - just do it.
- [071103] cc-delete; removed bogus 'checking out parent' error message.
- [071021] find; added -& support to replace-sting-in-files and added new *replace* output buffer for the command so hte user can see what has been done.
- [071018] Template buffer binding; avoid abort if binding can't be made
- [071013] OSD; fixed line hilight when selection string has regex chars
- [071013] Fixed deletion of symbolic links when they point to directories.
- [070923] Correct a file name when adding it to the history if it has a path character (i.e. not just a base name).
- [070923] Fixed ftp file rename (i.e. 'r' command in ftp dir listing)
- [070923] Only rename dir-list buffer to *files* if the current buffer is called *files* (i.e. browser is being used). Handle a dirlist (not called *files*) being displayed in multiple windows. Added file being loaded to file history.
- [070923] jst; Added support for tables being terminated by a ]? on the start of the last row, this makes tables far easier to use.
- [270923] jst; Added support for item-list and collapse. Added 4th level titles.
- [070920] Persist wrap mode in ipipe-rerun and disable compile hilighting when wrap is not used as very long compile lines kill performance (ME takes a lot of CPU)
- [070920] Reset hilight and indent buffer variables on a bclear, otherwise the hilighting persists in *compile* buffer across multiple compiles etc
- [070920] Added a bit 0x02 to the compile command to run the ipipe command without wrap (required if '<filename>:<line>: Error' lines are longer than the window width)
- [070920] Fixed html comment definition (should be <!– –>)
- [070920] jst; improved title.
- [070920] Added new 0x0200 bit to ipip-shell-command to disable wrap, similar to raw but still adds the header and exit info. Need this for compile when file names are longer than the width of the window (breaks goto-next).

- [070920] Fixed build.sh to build mec when X11 is not available - issue raised by TH.
- [070920] Added capability to 'set-variable @y ...' to set the current kill buffer.
- [070713] Fixed dir-list toolbar tool to handle dirlist with seconds.
- [070713] Perl template; added '-' to list of allowed hash characters.
- [070713] Build; handle VS8 fatal error compile messages.
- [070713] Fixed main edit menu's buffer-mode numeric args.
- [070524] Added a new 0x20 flag to find-buffer to stop the history of the outgoing buffer from being incremented - used by new find and grep to avoid trashing the buffer history.
- [070524] Fixed the handling of find-buffer's -ve arg, introducing a new 0x10 bit flag to control the forced hiding.
- [070524] Fixed file name input so trailing '/.' & '/..' have a further '/' added, this fixes several file loading problems.
- [070524] Changed file saving prompts to be like delete-buffer, they now given the file name first and then the 'save file' prompt so the user can always see the action they are being prompted for.
- [070524] dded an f3 buffer binding to be the same as v
- [070524] Improved filetype.emf to report if the last line is missing its line termination.
- [070507] Documentation correction from Bryan for "notes".
- [070426] Implemented cycling through loaded note files using the tree mode-line arrows (like paged tools in toolbar)
- [070426] Avoid opening the toolbar twice which can cause tool window sizing issues.
- [070425] Fixed cc-state end process cursor positioning prompt, now moves the cursor to after the [***....] exit line so the user knows its finished.
- [070424] Fixed a few start-up session restore and notes interaction issues.
- [070424] Fixed startup performance issues caused by testing the existence of every file
- [070424] Clearcase removed the 'Checkout parent' option for move and delete.
- [070424] Tool setup fixed issue with adding a new window tool after a paged tool.
- [070322] Handle file loading errors in find.
- [070322] Fixed the spelling error scheme.
- [070212] Fixed large file support on UNIX/Linux.
- [070211] find and grep -x option also skip ignorable directories.
- [061206] Fixed issue with the scheme editor.

- [061206] Fixed the OSD entry character handling.
- [061205] Changed creferences to replace-all-string commands to new (query-)replace-string-in-files commands.
- [061205] Implemented new (query-)replace-string-in-files commands based on new find. Changed default for grep to not use -l (use find-setup to do this). Changed -x option so it always ignores auto-save files.
- [061205] Fixed console/window session variable muddle, two run-mode variables are now kept separate.
- [061205] Added new find-setup comand
- [061205] Added latest Charcoal scheme (thanks to Bruno)
- [061203] Up-date of Mirkwood scheme (thanks to Bruno)
- [061119] Corrected minor warning on regular expression evaluation.
- [061110] Fixed spin on a cyclic symbolic link and added loop limit of 10 for chasing a symbolic link.

**9th Release / Patch 1 / Build 2 - November 2006**
Release date specified as 2006-09-09.

- [061105] JASSPA MicroEmacs 20060909 patch - build 2.
- [061105] Linux i386_x64 port corrections - Thanks to Vincenzo.
- [061105] Correct Java highlighting of package and import declarations - Thanks to Bryan Schofield.
- [061105] SQL template updated, removed bad token - Thanks to Rick Owen.
- [061105] Fixed word wrap horizontal scroll when using a window that is a few characters wider than the fill column. Caused next line to scroll which is very annoying - this is a very old bug!
- [061105] Fixed the Win32s Windows build which was using a later day Windows construct.
- [061105] Fixed NanoEmacs Window version under MS-Windows, crashing on start up due to accessing curFrame which had not been initialized before a screen draw took place from the Window message queue. Inhibit screen access during the construction of the frame.
- [061105] Documentation translate-key examples added - Thanks to Thomas Hurdt.
- [061105] Macros bailing out when the main menu is disabled on start-up. osd -1 0 returns FALSE which causes the macro execution to terminate. Thanks to Bryan Schofield for identifying this problem.

**9th Release - September 2006**
Release date specified as 2006-09-09.

User perspective summary of changes

- **MainMenu**(3) - A more consistent main menu and user dialog interaction.
- **file-browser**(3) - New key bindings to create, delete, copy files and directories. **file-tool-setup**(3) allows open actions to be associated with file types allowing automatic launching of external tools.
- Introduction of major modes for management of different file types. **buffer-major-mode**(3) assigns a new operating mode to a buffer affecting the highlighting, indentation, key binding etc. **major-mode-setup**(3) configures the behavior of the major mode.
- Introduction of sessions which store the context of editor including window layout and loaded files. Multiple sessions may be used, "**me -c**<*sessionName*>" restores a specific session. **save-session**(3) and **read-session**(3) handle the restore/saving of sessions. The existing history commands have been removed.
- **ClearCase**(3) support for IBM Rational ClearCase.
- **cvs**(3) improved support within the **file-browser**(3).
- **notes**(3) utility extended, new features allow quick notes to be generated with **notes-instant**, notes associated with a file may be added with **notes-context**.
- **rdiff**(3) recursive difference and **xdiff**(3) for access to an external graphical difference tool. **forward-kill-word**(2) inconsistencies removed.
- Highlighting improved, especially for long script files (html / php / javascript) where the highlighting used to be lost.
- **jst**(9) - Structured text documents.
- **insert-password**(3) password generation utility for system administrators who tire of thinking of new passwords.
- Fill columns are now defined on a per buffer basis **$buffer-fill-col**(5).
- **indent-increase**(3) and **indent-decrease**(3) commands introduced to change the indentation of a region.
- The default date format may be specified by the user with **%ascii-time**(5).
- Quick directory aliases such as " **proj**" or " **work**" etc. may be defined to a longer pathname using **alias-path-setup**(3).
- Language template added for Adobe Macromedia ColdFusion.
- Bug fixes and enhancements to existing commands and functions.

**Development history**

- [061005] java - Added missing highlighting tokens reported by Tom Hundt. Added new indentation definition and highlighting defined by Bryan Schofield.
- [061005] Fixed the cygwin f3 mode toggling to be the same as ishell.
- [061004] sessions - Avoided any potential timestamp issues by changing the divider to letters.

- [061004] Changed $timestamp default to be regex compliant (i.e. \\.)
- [061004] Changed copyrights to 2006
- [061002] jst - Up-dated jst hook and utils to latest standard.
- [061002] Changed arg to popup-window from 3 to 4 to resize as well, restoring a session on a different machine could easily break things otherwise.
- [061002] browse mode - Added wheel mouse support.
- [061002] notes - fixed notes-file bug caused by creating a new notes file while in notes. Close notes before importing skwurel notes.
- [061002] Fixed find bug.
- [061002] Improved the $file-names path/regex division detection and checked for '\\<' which can always be used as a divider if required.
- [060928] Fixed crash caused by ftp login being given a NULL password.
- [060927] Set macro version to *2006-09-09* as the release version. Set the executable to *2006-09-09* as the release version.
- [060925] Added a '`g`' (goto) to **save-some-buffers** prompt.
- [060925] Cosmetic changes to **buffer-info**, 2nd Col to 'ACol' and printed current character in decimal as well.
- [060921] Improved the notes display to better preserve the current notes file.
- [060921] Stopped **read-session**(3) from trying to restore lock and save buffer modes.
- [060921] Improved the **query-replace** prompt to list letter options when the width allows.
- [060921] Browser. Added seconds to the info given for a file in a directory listing. Fixed dir list hooks to handle mod times with seconds.
- [060921] Changed char prompts to have '`/`' dividers.
- [060921] Added new bit 0x20 to **read-file**(2) to discard changes to buffer.
- [060919] Build. Added $MAKEWINDEFS and LIBS to the ME_WINDOW defines, removed references to xpm from the makefiles as this is not a standard package and moved references out of *emain.h* into *unixterm.c*. Vamped up build shell script to check for X11 & Xpm availability. Changed horrid dos style empty string tests in UNIX scripts with **-z** & **-n** test options
- [060918] Standardized the form of character input prompts and improved routine so '`?`' swaps between help and prompt. Added a '`N(o) to all`' option to **save-some-buffers**(2) and changed **fill-paragraph**(2) to '`Never`' to '`N(o) to all`'. For **query-replace-***, changed the help for '`!`' to '`Yes to (a)ll`' and added '`a`', changed '`i`' to '`e`' and removed '`.`' option.
- [060918] Changed debugger '`!`' to '`c`', in keeping with other debuggers for *continue*.

- [060918] Added bit 0x04 to **@mc**(4) flag so macro can supply a help string
- [060917] Corrected the X-Windows ordering to allow us to receive (Un)Map events on Linux - X-Server behavior is different from Sun Solaris.
- [060917] UNIX. Added XPM colour bitmap support. Just smartens the ME icon up in Gnome and makes ME look a bit more 21st century without the monochrome icon.
- [060917] Shell template, addition of missing common commands.
- [060917] JST, addition of other text insertions.
- [060917] UNIX. Corrected X-Windows positioning from popup-window.
- [060917] Added new 48x48 icon for gnome with transparency boarder - complaints that existing icons were not Gnome friendly!
- [060917] UNIX. Corrected X-Windows hints for font size change. Requires that the X-Window is unmapped before the size hint may be modified and the re-mapped.
- [060915] Notes. Fixed **next-frame**(2) prefered selection order.
- [060915] OSD. Sanitised the quick file open and file type buffer open menus and changed them to use a scrolled list if more files/buffers than lines rather than filling the screen.
- [060915] In notes find a file that is loaded but out of date, this leads to an reread prompt during which the notes ehook/idle callback kicks in and changes frame - this breaks the message line of both frames. This is a rather horrid fix to a rather grotty problem!
- [060915] Browser, fixed the sort order to be case sensitive on UNIX, insensitive on Windows
- [060914] User setup. Stopped user-setup's apply from losing the current session's override colour scheme
- [060914] ClearCase. Fixed Describe context menu hotkey overload. Used `-mkpath` to implement a '*checkout parent*' for **cc-add**.
- [060913] Notes. Fixed the loosing of edit and crypt mode from the main notes buffer due to the setting of the major mode. Fixed notes tree context menu hot keys.
- [060913] Fixed kill-list tool up-date trashing the real kill ring.
- [060912] Changed session-update to store view mode for a buffer only if it was forced on by the user (i.e. the file is not readonly). This stops files re-loading in view mode if the permissions have changed - this was a problem with ClearCase reported by Enver Haase.
- [060912] Simplified the new note-setup and fixed the notes buffer losing the hide mode (which causes problems for session restore).
- [060911] Removed the notes 'note' dummy hook in favour of using a real hook. Set the default hook to be jst but can be set in the -!- enf; major-mode: <hook> -!- header. Added new note-setup which users buffer-setup to allow the user to setup the buffer and stores the values for that note.

- [060908] File browser, bound a **file-attrib**(3) based 'Change File Attributes' option to 'g'
- [060908] Changed **file-attrib**(3) to handle directory buffers correctly and, with a numeric arg of 0, change the attibutes of a given file (i.e. not the current buffer) Implementation changed to now have a Cancel button.
- [060908] File tool, implemented new `%"...%"` tag to enable proper quoting of arguments.
- [060907] Fixed startup buffer hilighting issue caused by order of session restore.
- [060905] ColdFusion template, added string highlighting to the SQL statements.
- [060905] File browser, added binary edit, bound to 'b'.
- [060905] Added basic xml "/>" single tag support to html and fixed xml's insert-attrib to handle this
- [060905] Changed refences to *white-char** to whitespaces (i.e. display-white-chars is now display-whitespace) Fixed hide main menu to popup an info dialog informing the user how to get it back.
- [060905] Fixed ftp dialog to conform to new *<user>.erf* setup standard by adding a new 'Make Default' dialog, and improved the behaviour of Enter in the dialog.
- [060904] Corrected the ColdFusion closing bracket hgilight.
- [060904] Shortened the document template formating menu text strings.
- [060829] Fixed HTML template **html-insert-tag** to add the closing "/" on insertion of a "`<singleTag/>`". Fixed **html-insert-attrib** to deal with attribute insertion into a HTML tag ending in "/>".
- [060828] SQL Template. Corrected SQL string hilighting.
- [060827] New language template for Adobe/MacroMedia ColdFusion (file extensions .cfm, .cfml .cfc).
- [060827] Extended indentation with the 'u' and 'v' indentation types.
- [060822] 2006 Release Candidate 1 (RC1)
- [060822] Added password commands **password-to-phonic**(3) and **insert-password**(3).
  [060816] Added settings for fill-col and fill-mode to the **major-mode-setup**(3).
- [060816] Added support for a "`.fhook-???.fill-*`" variable and changed latex over to using this variable rather than burning the fill-mode in as 'l'
- [060816] Crossword - Changed Exit button to Close.
- [060816] Changed organizer Exit buttons to Close or Cancel.
- [060812] Additions to the PHP script - missing tags etc.
- [060812] Corrected sun keysym inclusion - inclusion limited to X-Windows build only

- [060811] Fixed session resize issue and incremented the version.
- [060811] Added $buffer-fill-* variables to the buffer-setup dialog
- [060811] Stopped read-session warning about the *server* buffer being a process.
- [060811] Updated to use the new $buffer-fill-* variables - simplifies the doc hooks as ehooks and bhooks are no longer required.
- [060811] Incremented the version. MicroEmacs 06 - Date 2006/08/10 - sunos
- [060811] Introduced new $buffer-fill-col and $buffer-fill-mode variables to allow every buffer to have its own setting, initialized to the global variable value when the buffer is created.
- [060810] Added 'a' for All option to the save-some-buffers prompt
- [060810] Added save -> prompt -> delete of all buffers when loading another session. If bit 1 in arg is not set (i.e. 0 read-session "name") then this is not done and the new session is effectively merged into the current.
- [060810] Added setting of main buffer modes to the buffer-setup osd.
- [060807] Updated emulation modes to handle the new main menu.
- [060807] Fixed major mode drop down list to not scroll horizontally
- [060807] A few tweaks to help with the other editor emulation.
- [060807] Fixed major-mode-setup so Cancel cancels all changes (was 50-50)
- [060807] Removed old buffer-setup file and fixed the makefile
- [060807] Removed 'correction' of main osd - not needed and wrong.
- [060807] Tweaks to osd to remove a few annoyances: up on main menu now goes to last item of current sub menu, fixed re-frame and focus issue on an opened child and made C-g always just close a child even if not in focus.
- [060803] Added new -2 and -3 values for 2nd arg to &lins to insert alphabetically and case insensitive alpha.
- [060803] Added setting of the major mode to the new buffer-setup
- [060803] Changed osd dialogs to use 'Cancel' or 'Close' rather than 'Exit'
- [060803] Closed notes if open when applying changes as this corrupts the notes frame.
- [060803] Made the buffer sub-menu be case insensitively ordered in windows and have first letter hotkeys.
- [060803] Changed frame-id comparison to use seq as should handle "ERROR" better.
- [060803] Fixed notes to handle a reread-file type operation.
- [060803] Changed osd dialogs to use 'Cancel' or 'Close' rather than 'Exit'

- [060802] Added saving the session to save-all and associated a file in $user-path with the File->New command.
- [060801] Set .gz extracted buffer name and file name so its location is kept.
- [060728] Fixed clear-case to handle multi-line comments with SHELL set to csh.
- [060728] Changed ”symbol” to ”insert-symbol” and ”indent-setup” to ”buffer-setup” as this is what it will become. Fixed a few hot-key issues with new main menu.
- [060728] Improved help page and looked for context-menu binding to support esc +.
- [060728] Moved the location of making CVS and clear-case always available into their setup dialog and added a C-x C-q replacement option.
- [060728] Changed backspace in a directory to move to the sub-directory in the parent listing.
- [060728] Fixed notes startup so it returns the user to the same place.
- [060728] Documentation, replaced -ve with negative and +ve with positive.
- [060726] gdiff. Added termcap friendly key bindings to move between difference regions.
- [060726] Added new indent-increase and indent-decrease macros. Added indent increase and decrease to format, changed notes binding to f8, move shell tools from tools to advanced and setup commands into tools.
- [060726] Fixed setting of scheme when session file does not exist (problem hit when old user uses session for first time)
- [060725] Changed goto commands to not stop the show-region highlight
- [060725] Major rearrangement of the main menu, replaced the Insert and Execute sub-menus with a View and Advanced, re-balanced and relocated many items, most notably the setup dialogs out of Help into Advanced.
- [060725] Renamed the existing tool-bar macro to toolbar-open (mirrors existing toolbar-close) and introduced a new tool-bar macro that toggles state, i.e. closes if opened. This command is more useful in key-bindings and menus etc.
- [060725] Improved error handling of osd based find tag, also adding support for tags file being in a parent directory.
- [060725] Removed command-line validation and relax file selection requirement so user can create a tool that does not require a file name (e.g. open . in explorer)
- [060725] Used set/goto-position to store current dot and added <no favorites> entry if menu is empty.
- [060725] Added hot-keys to the file attrib dialog
- [060725] Changed simply fence matching of quotes and double quotes, if match is not found by the end or start of the line it’s assumed to be a single and ignored.

- [060725] Changed &mid so the 3rd argument (length) can be -ve to mean from the right (as per the &rig etc) - not sure why this was not done first time round.
- [060624] Changed session support to handle both window and console environments. Changed startup to only initialize one scheme.
- [060623] Added file-type-setup to the list of misc setups.
- [060622] Changed buffer-setup to major-mode-setup, also changed buffer-help to major-mode-help etc and fixed comments. Left buffer-setup in buffstp as there will be a buffer-setup which really does setup up the current buffer. Change me.emf date to 20060606
- [060620] Stopped session trying to change the frame for a console run.
- [060619] Sessions. Stopped restore buffer's whose file does not exist and changed the order of frame size setting so on UNIX the depth of windows are maintained correctly
- [060619] Notes. Fixed current buffer position reset problem caused by buffer being shown on hidden notes frame
- [060619] Changed all ME setup macros to reload the /history registry before change the registry and writing it out as soon as the changes are accepted to reduce the likelihood of settings (like new alias paths) being lost. One except is when the setup is run from within user-setup in which case the registry is only read and written by user-setup, this could have some unwanted side effects. Added session-setup to the user-setup misc page.
- [060619] Change to allow macros access to inactive buffers broke file and ipipe hooks which get executed but the buffer is officially made active - fixed by maintaining the BIFNACT internal flag and ensuring the algorithm used for inactive buffers is only used wh ere appropriate. Found buffer context was not set correctly for session buffers whose name was set to 'name<1>' as the BIFNAME flag is not set, fixed by removing this flag and using code to assess if the <1> should be removed based on the filename.
- [060619] Fixed read-registry's replace mode which was doing a merge as the old node was not removed.
- [060619] Fixed a bad prompt and added a couple of brackets to protect compare order.
- [060616] Changed the start-up process to not use the history commands (which have been removed) but simply use the read-registry commands.
- [060615] Moved the storage of all history type info into the new session (esf) files.
- [060615] $user-name.erf only contains setup info and not often edited changed user setup to minimize the likelihood of losing edits (at last).
- [060615] Broken session-setup out into its own macro file and made quite a few fixes to the new session support.
- [060615] Removed the save- and read-history commands the me.emf start-up now simply use the read-registry command

- [060615] Changed find-buffer to allow limited access to inactive buffers (allows base getting and setting of variables)
- [060615] Changed the exit-emacs numeric argument (again) so it can be used by meDie (0x28 exit-emacs = emergency exit)
- [060615] Created new $window-mark-line & $window-mark-col to aid inactive buffer support (not sure why these were not already there as they are usefully in macros)
- [060615] Enabled availability of save-some-buffers by default (quick-exit relies on this!)
- [060615] Changed the -c command-line option to simply set the variable %session-name (relies on session macros)
- [060615] Changed clipboard support to not mess with it during start-up and shut-down
- [060615] Created new change-scheme command.
- [060615] ClearCase. Fixed *cc-state* buffer not being in view mode when command completes.
- [060615] Corrected types of exchange-point-and-mark.
- [060608] Improved OSD search dialog by making the buttons always enabled and reporting an error if search string is empty.
- [060608] Made the existing hidden scheme macro a visible command which can be used to change the current color scheme. Set .scheme.current to the file name of the current scheme Saved the current scheme in a session so a user can create a session with its own different scheme.
- [060526] Sessions. Changed to use new shut-down pseudo key.
- [060526] Changed shut-down process to call the ehooks for the current buffer of all frames and changed the shut-down macro to a pseudo key-binding.
- [060525] Added a new buffer-major-mode command to change the current buffer's hook, giving it an arg of -1 or major mode of "" removes the hook. No completion list yet. Fixed tab-width setting so all 3 windows get the same new value.
- [060525] Changed gdiff to insert the 2 files into temp buffers so if the files are already loaded by the user these buffers are not effected. Added the ability to set the tab width.
- [060525] Fixed crash caused by watch refreshing the buffer while user is using isearch - stopped watch from refreshing the buffer.
- [060525] Added a new bit 0x10 to find- read- and view-file to disable the context fhook evaluation.
- [060525] UNIX fix. Call screenUpdate directly after a window resize - fixes gdiff update problem.
- [060523] Sessions. First serious rev of session-load and -save.
- [060523] Added arg to tool-bar to allow the tool-bar to be initialized without re-reading the registry - required by new session support.

- [060523] Changed left click on mode line to change to the window if the mouse is not moved.
- [060523] Fixed C++ '::<function>() ;' indentation bug introduced with c++ initialize ': <base-init> ()' indentation support.
- [060523] Encrypted files. Added file/host to the password prompt so on a -c the user should know which password is required.
- [060523] Fixed UNIX compiler warnings
- [060516] me.emf. Fixed missing def of file-type-setup & shell-tool-setup.
- [060515] Added new bit 0x2 to arg for change-buffer-name which force the buffer to have the given name, changing the name of an existing buffer if required.
- [060512] ClearCase. Fixed cc-setup view deletion, pressing No does not remove it. Removed the temporary file created by cc-diff if me's diff of gdiff are used, file is still left if xdiff is used.
- [060512] Added a new $mode-chars variable which returns the letter ids of all modes as a string, this variable cannot currently be set.
- [060512] Changed the numeric arguments for quick-exit and save-buffers-exit-emacs to make them simple to use.
- [060510] Improved the ftp logging so errors get put into the console with error numbers rather than just mlwrite'd. Removed the KEEPALIVE as ME closes the sockets with 10mins anyway. Fix the ftp kb/s calc.
- [060510] Dirlist. Fixed context menu so the FTP additions are added for any ftp:// listing.
- [060504] Major addition to notes: a) Added forw/back note. b) added notes-instant and notes-context c) implemented notes-session-save (not done window layout yet) but not implemented the matching load yet. d) fixed a few annoying up-date and behavior problems.
- [060504] ClearCase. Fixed windows \\r\\n issue with diff - constructed predecessor could have the wrong format breaking the diff.
- [060504] Changed reyank numeric arg so if a -ve arg is given reyank will fail if at the end of the kill chain rather than loop - required for notes-session-save
- [060504] Found differences between copy and kill-region's handling of narrows, particularly if there's no markup line Found I could crash ME if only the end part of a narrow-to is killed - buffer line number all go wrong. To fix the issues I've created a standard function to handle the expansion of narrows over a region and made the rule that for a narrow without a markup line to be included in the region at least the \\n of the previous line and 1 char of the container line must be included, this means that narrows at the top and bottom cannot be included.
- [060504] ClearCase. Fixed cc-merge comment bug.

- [060504] Added support for a .ascii-time.format variable which defines the format for the output using '%' tags as per strftime function (supports most tags) - output is unchanged if variable is not set. Fixes request by Dick Pierce. Changed the OSD insert time function to use the same tokens and made the first format in the list be the output of ascii-time.
- [060503] Provided macro interface to get a list of alpha marks for the current buffer.
- [060427] Fixed context-menu element activation issue caused by selected file count being wrong.
- [060427] Fixed notes and gdiff window resize issues by locking the size.
- [060426] Improved (but not quite fully fixed and I don't think I can) the mouse word selection when the word is half hidden at the left or right edge of the window.
- [060425] Fixed bug with setting $window-line, if value is < 1 then the goto-line function prompts the user, this is incorrect behavior for a set variable command and the cause of a mouse driver spin. Fixed mouse left button spin caused by missing the pick event and a bug in set-variable for $window-line. Added new horizontal scroll feature when user clicks on a $window-char truncated line char.
- [060421] JSP. Fixed java <%...%> within JavaScript start token
- [060421] Fixed C++ comment hilighting
- [060413] Added an abort-command safty trap to reset the $buffer-input variable.
- [060413] JSP. Fixed most JSP hilight and indent issues caused by java and javascript confusions.
- [060412] Bug fix. Put checks in to ensure the hilight/indent to be used according to a look-back scheme has been defined - fixes crash when viewing JSP files.
- [060403] DIRECTORY LISTING. Fixed 'not-exist' fix in last commit - ftp dir lists can be only a couple of lines Made file/dir copy automatically try 3 times before prompting the user (large ftp copies will fail at points). Fixed browse mode close problem when browse deletes the buffer.
- [060402] Used an ehook and a callback to auto detect a change in the notes frame buffer/window layout, if there is a change the callback closes the notes frame. This approach resolves many issues.
- [060331] Fixed XML magic hook.
- [060331] Fix annoying find-zfile last buffer history change. Changed the buffer & file name of a loaded zfile file to something more useful.
- [060331] Created a '2 find-buffer <file-name>' arg which finds an existing buffer via its file name - used to fix annoying find-zfile buffer history change.
- [060330] Fixed CVS & CC context menu placement issues. Fixed cc-add to use -mkpath for directories

- [060330] Change fhook-dir to always detect if its a source-controlled dir and call the SC hook, the SC hook can then add its own bindings etc. Fixed fhook-dir so loading of ".../not-exist/" does not error.
- [060330] Made the esc = & esc + menu bindings global.
- [060329] Changed unix start-up mode to favor X over termcap if DISPLAY is set, even if term is a vt100.
- [060329] Fixed mode-line udate bug when all edits are undone - reported by Tom Hundt.
- [060322] Fixed the position of the context menu when opened by mouse or menu key. Fixed auto-spell not being available when context menu is opened using the menu key.
- [060322] Improved gdiff-next command to error at the end of the file so the use knows they have reached the end. Bound 'menu' to open the contex menu and bound 'esc up' & 'esc down' to move between unassigned sections.
- [060322] Bound f10 to close the notes frame and bound C-x k to a new notes-delete command
- [060317] ClearCase. Added new cc-merge command (which uses cleartool's find-merge command!). Added new cc-describe command (which uses cleartool's describe command), fixed the CC context menu.
- [060317] Directory listing. Increased 2Gb directory file size limit to 4Gb by making size variables meUInt.
- [060317] Fixed '\\' quoting in command-wait example.
- [060316] Up-dated docs for command-wait, added some meaty examples and added cross references to them.
- [060316] indent fix
- [060316] Changed describe-word osd dialog so that it has a 'Look Up' button which is the default. Also improved the insert description process so C-w kills the description inserted.
- [060316] Improved the behaviour of the enter key in the help osd so it does a search or nothing.
- [060315] Fixed a win32 console crash caused by the new frame reposition function. Stopped the messing about of the windows console when ME is in piped mode.
- [060309] Fixed regex \\a -> \\g for bell and added support for \\a & \\m to [...] character classes.
- [060309] Updated *shell-command docs.
- [060309] Documented &lef, &mid & &rig -ve 2nd arg usage.
- [060309] Documented insert-string's -ve arg.

- [060309] Up-dated goto-matching-fence docs
- [060309] Fixed docs for when no arg is given to forward-kill-word.
- [060309] Updated jst template to new version. Fixed a couple of bugs with tables (empty cells and bad indent) Implemented a table cell '!' flag to indicate a start of a row (helps debugging) Fixed bug when title variable not set.
- [060309] ClearCase. Minor improvement to setup layout
- [060222] Exit handling fixed, saving of modified buffers, history, spell dictionaries etc. "**Emergency Exit**" messages removed from exit handling.
- [060214] Corrected user setup mouse defaults - missing wheel mouse default bindings.
- [060214] Improved the php look-back scheme to identify php script quicker.
- [060213] Fixed crash caused by the new hilight look back scheme using the indent look back function which used the indent scheme rather than the hilight scheme
- [060210] Improved php hilighting by making use of the new look-back-scheme and look-back-only-token features.
- [060210] Improved the time printout
- [060210] Benchmark tests. Improved the disptest.
- [060210] ClearCase. Fixed cc-add when adding just a directory.
- [060210] Added basic mouse support to browser and honours exit bindings
- [060210] Added look-back-scheme support to hilighting (uses same code as indents quick-lookback scheme). Added new 'only use during look-back' hilight token (combined 0x800 & 0x200) which can solve make multi-line string hilighting issues.
- [060210] Fixed compiler warning with -m changes
- [060210] Removed unused variable in new reposition function
- [060210] Fixed unused variable on NDEBUG build
- [060209] Made changes to the -m server message option to make it usable. a) Process the message using the macro 'token' function so \\n is translated to a new-line char. b) If the message is not terminated with a \\n then one is added. c) The popup-window message it only given when -o option is used
- [060209] Added support for a server 'M' message command.
- [060209] Changed the hilight and indent meHilight structure mis-uses access the data via #define (added to estruct.h) so the code is clean and easy to read and the miss uses can be easily found and changed. Changed the storage of the main indent and hilight scheme flags to use the ignore field (only 5 bits currently used) so the trunc scheme can use the type and the hilight look back can use the close (i.e. unlimited).
- [060209] Fixed clear-case update list on checkin option.

- [060208] Fixed crash caused by mlGetStringFromUser being called recursively - only possible since the introduction of ftp completion which can prompt for the user password while inputting a file name.
- [060207] Added an implementation of meFrameRepositionWindow for XTerm, not a very good one as there is not a clear standard between WMs.
- [060207] Fixed obscure crash caused by horizonal scroll mode 2 and long lines.
- [060202] Build script. Fixed MS-Window's typo.
- [060201] ClearCase + CVS. Fixed hiding of *??-console* buffer and closing the window with loading a file from a dir list.
- [060127] Broke the user-setup 'Tools' 'File Types' and 'EMail' pages out into separate independent setup dialogs. Added new Miscellaneous page to user-setup with links to these and other setup dialogs (such as alias-path and file-tool) Changed the buttons to the more standard Okay, Apply Cancel. Slightly changed some of the Mail & VM registry settings to make each platform independent (except filtering).
- [060127] ClearCase. Added new cc-delete and alias-path-setup commands. Added f4 to diff binding. Moved the window resizing call to nearer the end (after the font resize as this could change to a smaller font allowing more lines etc).
- [060127] Fixed quite a few issues with the CVS and clear-case UI interfaces. Brought cvs-state into line with CC, implemented cc-delete and rationalised the hot-keys and context menus.
- [060127] Added new alias-path-setup OSD gui.
- [060127] Added new 3 and 4 arg to popup-window to manipulate the main frame window, implemented meFrameRepositionWindow function on windows.
- [060127] Fixed find-reg index in error message.
- [060127] Moved macro parser's \\a to \\g (bell) and added support for \\a - alpha and \\m - alpha-num to hilight and regex.
- [060125] Changed the default debug for 'deb' abbrev to 2.
- [060125] Improved the hilighting of html &???; chars - issue raised by Tom Hundt
- [060124] Fixed crash caused by moving right (cursor key) into a sub-menu with no active items - bug found by Tom Hundt.
- [060122] gdiff. Fixed the bugs introduced with the 3rd 'current line' window was introduced.
- [060122] Changed call from saveExitEmacs to exitEmacs to avoid compile errors for NE.
- [060122] Fixed the meDie function to not exit straight after the emergency exit message, now autosaves as it should. Change the handling of the CLOSE message on windows so it prompts okay/cancel if buffers need saving and then exits. Fixed really annoying modifier state mismatch problem on windows - reproduce by pressing control right-click on title bar to get the system menu and then release the control, ME misses the C-drop event so thinks its still pressed.

- [051220] JST extensions.
- [051219] JST. Added implementation of TOC to html export.
- [051219] ClearCase. Disable check out of parent dir for cc-add as its not implemented
- [051215] JST. Changed tag highlighting to handle split line tags correctly. Implemented support for f and b color export to HTML using style.
- [051215] First commit of new v4 JST format hook and export macros.
- [051215] Fixed the return value of transpose-lines command which was always returning false causing cc-add to fail.
- [051212] SQL Template. Missing hilight statement causes execution to fail - Thanks to Th.
- [051212] Fixes to old JST before it gets trashed
- [051125] Fixed regex bug where "[\\s]" incorrectly matches an 's'
- [051125] ClearCase. Added support for not loaded to cc-state
- [051123] Added support for alias-paths.
- [051122] Fixed highlight problems with "^ *$" & "^\\\\s*$" close bracket tags, they should match any line containing only spaces/white-spaces but they did not work reliably.
- [051122] ClearCase. Fixed the cc-diff, external diff settings had got muddled.
- [051122] Directory listing. Changed the size ordering to list largest first on initial sort.
- [051118] ClearCase. Implemented the auto get checkout comment and removed the temp command-line buffer.
- [051118] Directory listing window. Added binding of 'f' to toggle the filter and changed the 's' and 'S' bindings round (s is far more useful)
- [051118] Improved the horizontal scrolling of gdiff and browse
- [051110] Make debugger stop at executed !elif & !else will work for all macros, may not work when executing a buffer as the line may start with a white space.
- [051110] Stopped osd check-box macros from triggering the debugger when $debug is set to 1.
- [051110] Added some more key translations based on bug report by Francesco Abbate.
- [051104] 2nd and much improved version of cc-setup. Changed to using -1 insert-string to allow multi-line comments. Fixed horrid problem with cleartool's handling of checking out a file when its been hijacked, no command-line option to make it use the hijacked file me has to use ipipe-write a send 'yes' at the prompt, but as the prompt is not given me must be preempt and send them first, one for each file - clear-case is pants and cleartool sucks.

- [051104] Changed cmd-line construction to use -1 insert-string to insert \\n's into the line - not really very nice.
- [051104] Fixed meGetChar to be like GetString in that if running a macro the value has not been given it goes directly to the user rather than simply failing, this improves macro reliability and eases debugging of macros.
- [051104] Multi-line shell fix. The new 0x100 flag to shell-command has one drawback, you cannot run commands that include newline characters as the cmd-line must be on the first line. But CVS and clear-case commit comments are often multi-line so this is not acceptable. To fix the problem I have added a -ve arg to insert-string comment which then performs a literal insertion, i.e. every char including a \\n char is inserted are text. This fixes the immediate problem but inserting a \\n char in a line can break other thing in a very major way (like undo) so it must be used with extreme care.
- [051103] ClearCase. First rev of a CC tool defaults setup in cc-setup. Moved CC over to using 0x100 bit in shell-command to remove command-line length limit.
- [051102] Moved multiple-file file-tool executions over to using the new 0x100 shell-command flag (first line of buffer) to remove command-line length limit in almost all cases (cmd has a 4kb limit)
- [051102] Introduced a new 0x100 bit for shell commands which means the 'cmd-line' given is a buffer names whose first line holds the command-line to run.
- [051028] ClearCase. Fixed cc-diff abort when a file has no differences. Added a 'delete line' function to cc-state bound to C-k.
- [051019] Visual basic highlighting - added And Or and Xor operators.
- [051013] Unix shell window, corrected highlighting of backslash escape character outside of a string enclosure.
- [051011] ClearCase extensions. Added button to cc-checkin to get checkout comment. In cc-state - fixed file selection on last empty line and mouse selection and added support for mouse drag selection.
- [051011] Changed dirlist copy/move to preserve the original files time stamp.
- [051011] Fixed windows rename to check the to name is case insensitively different.
- [051011] Implemented file time-stamp preservation in the file-op copy and move.
- [051011] Improved the window-popup on windows, more likely to work but can still fail.
- [051011] Fixed UNIX drag-n-drop, broken by -l l:c addition.
- [051007] Integrated Ric Yeates command line **-l <line>:<col>** submission - thanks for the submission.
- [051007] ClearCase. Removed VOB option from **cc-state** as it did not work. Made **cc-console** read only and better integrated including supporting it as a toolbar tool (like cvs-console).

- [051007] Fixed bad variable assignment in directory list macros.
- [051001] Corrected Sun Solaris x86 makefile to use the Sun Native compiler.
- [050930] ClearCase. Fixed **cc-state** to check the state of selected directories, which requires a double cleartool call (horrid command-line) Added support for config-spec viewing and editing. Added directory refresh option for checkin/out add etc. Added many improvements to cc-state buffer support, hilighting, file listing style sorting and selection (i.e. following are bound `1`,`2`,`3`,`+`,`-`,`u`,`a`,`*`)
- [050930] **file-tool**, changed the buffer list toolbar tool to keep the current line to the current buffer when using **find-buffer** etc.
- [050930] Latest *Mirkwood* update from Bruno.
- [050930] Fixed ftp drag'n'drop copy bug, the source dir needed an file: prefix to remove the `ftp://` of the current buffer. While test I blew away my home directory, nasty way to discover that ftp can return `".."` not `"../"` and `".."` is not ignored when deleting! Fixed.
- [050929] File listing, stopped copy, move etc commands from accepting `'../'` when its the current line Fixed `'v'` browse file open and close.
- [050928] browse, removed the delete buffer flag from browse - wrong way to do this.
- [050928] ClearCase, make **cc-annotate** work if file is checked out (cleartool command-line is awful, why do you have to pay money to use this rubbish?)
- [050928] Added wrap feature to **browse** mode.
- [050928] Changed *Mirkwood* main OSD scheme to make current focus item more visible.
- [050920] Changed the $debug variable to be bit based as its old values just could implement the required functionality.
- [050920] Latest *Mirkwood* highlight scheme - added underlining for spelling errors
- [050920] OAD, added a comment tab to the scheme editor and up-graded the scheme selectors to use combo-boxes.
- [050920] Fixed compile error on macos caused by bad macro definition.
- [050919] Fixed 2 bugs: meRegHead's force could be left set to 1 which has the side effect of `C-g` not hiding the active region. Due to last change to **command-wait** the screen is not up-dated before waiting for a key breaking **gdiff**.
- [050919] Readability update to *Mirkwood* highlight scheme.
- [050919] Changed ClearCase **cc-state** to handle file names with '`[`'s in them.
- [050915] Added Up, Down, Hide window and default open options to the **file-tool** setup GUI. Added support for a default open to the directory listing, pressing return now checks for a **file-tool** default open (explorerish), '`o`' for open always loads the file into me.

- [050915] Added support for dbx on sunos. Changed the ipipe input mode to be toggled using `f3` rather than set via `f2`, `f3` or `f4`.
- [050915] **zfile**, fixed filename -> absolute file name when given as an macro arg - should be based off current buffer's path.
- [050908] Added **file-tool**.
- [050908] ClearCase, fixed `%diff-com` definition problem and **cc-uncheckout** buffer update problem. Fixed **cc-checkout** and **cc-checkin** buffer update problems as well.
- [050908] Improved the diff highlight, more readable and correct.
- [050908] Added new *Mirkwood* highlight scheme.
- [050806] Added better support for MinGW. Fixed the console color bug and most of the compiler warnings. Added support for the MSYS shell by creating a unix style `mingw.gmk` (uses rm instead of del etc) updating the old `mingw.gmk` and renamed to `mingw32.gmk` and added support to the unix build script.
- [050803] Fixed **ne** crash when an unbound key is pressed, bug introduced with last commits.
- [050721] Fixed annoying mouse refocus in osd dialog even though mouse is not moved - another windows nasty.
- [050720] File browser, added directory closing binding to 'c' and '+' '-' windows compatible bindings.
- [050720] Pop-up menus, shifted the context menu down one line when opened by a key.
- [050720] Directory listing. Changed the tag file list macro to support moving to the closest selected so the user can see which files they are deleting! Improved the file-tool so \\H will see a hot-key, '\\' chars and '"' must be quoted (i.e. '\\\\' & '\"') Bound 't' to open the file tool menu directly so tools can be easily executed without use of the mouse.
- [050720] Source code reorganisation. Standardized the # preprocess forms, removed 2 uses of #elif and cplusplus and changed odd 'defined(_BSD)' style bracketing which the **c-hash-eval** does not handle.
- [050720] Removed unused functionality from NE. Made the following available as extended functionality: **backward-delete-tab**, **change-buffer-name**, **change-file-name**, **define-macro-file**, **delete-blank-lines**, **delete-some-buffers**, **named-buffer-mode**, **save-some-buffers**, **save-buffers-exit-emacs**, user(%), buffer(:) and command(.) variables, **$buffer-input**, **$kept-versions**, **$mouse***. **$pause-time**, **$MEBACKUP-PATH**. Removed **-c** from ne's help page as this is not supported. Moved macro regex **&xseq**, **&xrep** based directives and most @ variables to extended functionality.
- [050719] Organiser address search broken due to recent changes to macro argument getting. Fixed by changing the numeric argument which was ambiguously defined.
- [050718] Source code reorganisation. Made the file hook variables and HideBuffer function statics. Changed ne to use fixed char groups for isUpper etc rather than the large look-up tables as it does not have **set-char-mask** to configure them anyway.

- [050718] C template. Fixed and greatly simplified the #if handling in the hash eval code. Added support for `#undef`.
- [050715] Source code organisation. Removed unused code when there's no ipipe support. Hashed out location functions which are only required by socket or ipipe support.
- [050714] Combined **cc-log** and **cc-annotate** into a single **cc-annotate** cammon with GUI. Added **cc-vtree** and **cc-explorer**.
- [050714] Fixed **user-setup** setting of number of backups and Ignore files settings.
- [050714] Fixed **file-op** copy so the source file's attributes are preserved (i.e. remains read only etc). Implemented the 4th parameter (attribs) in **&stat i** Implemented a **file-op** chmod using arg bit `0x200`.
- [050712] Corrected NE build under Windows, console build including windows calls.
- [050712] Directory listing, fixed delete copy and move prompts, flag `0x08` got changed to `0x04` in **file-op**.
- [050712] Implemented a proper ClearCase context diff by recreating the old file from the diff and then calling a real diff program.
- [050712] Changed the diff hilighting to be just one scheme as a CC diff could contain a gdiff style for files and diff style for directories.
- [050712] Corrected NE build problem with macro debugging.
- [050711] Corrected Latex template load error in `hklatex.emf` (detected with `$debug=1`).
- [050710] Fixed old `$debug` interface, found a long forgotton mode 3 (now 4) which simply prints the current line without waiting, this broken the new mode 3.
- [050709] Removed the **-d** command line option - pointless.
- [050709] Improved the macro command argument handling to get missing arguments from the user, i.e. having a macro line of: **fine-file** Works okay.
- [050709] Vastly improved the macro debugging, the **$debug** value of 1 is now 2, 2 -> 3, a value of 1 now tells the user of any unhandled failures.
- [050709] Fixed the Windows window resizing behaviour to be a lot more usable. The window size now increases in character size steps rather than bouncing around and the window is refresh as its changed. The Horizontal width limit of the window has effectively been removed (set to 400 chars) the depth is still restricted to the screen size, issue raised by Tom Hundt.
- [050709] Added new file tools interface, creating windows explorer type file name based tools (e.g. extracting a zip file).
- [050709] Added ClearCase **cc-log** and **cc-annotate** (similar to cvs's commands)
- [050709] Stopped the default file hook from creating a bogus tool menu and help page.

- [050709] Fixed the previous-line macro and the kill-word key bindings.
- [050709] Changed the menu items which open a sub dialog to having '`...`' in their name as per the gui standard, issue raised by Tom Hundt.
- [050708] Changed sub-menu display chars, auto open menus are now '`>`' and manual are '`+`', issue raised by Tom Hundt.
- [050708] Got thoroughly fed up with the emacs style **kill-word** - useless. Came up with a compromise, if no numeric arg is given then it works the old way, if an argument is given (i.e. 1, 2 ...) it works in like emacs (which makes sense as you are asking it to kill 1 word etc.)
- [050708] Changed **list-commands** and **command-apropros** to not list hidden macros by default. They can be listed if bit 1 of the numeric argument is clear (aka **list-buffers**)
- [050625] Updated mode line letter mnemonics which were out of date, issue raised by Tom Hundt.
- [050625] Stop me crashing when describe-variable is given something like "**&or**", now restricted to variables of type `#`, `$`, `%`, `:`, `.` only.
- [050618] Added support for -ve offests to **&lef**, **&mid**, **&rig** functions to go from the end of the string.
- [050618] Created a *no wait* option to **shell-command** so processes can be launched and forgotten about. Also added support for all the other process launch flags present in pipe and ipipe shell commands. Removed some very dead code.
- [050618] Avoided a '**&mid -1**' in display matching fence (`dmf.emf`) by restructuring the logic a bit.
- [050616] Revision 3 of ClearCase support.
- [050614] Second rev of clearcase support. restructured the execution of cleartool so ipipes are used (even though the user may not be able to edit) giving feedback on progress. Implemented a basic diff which can only diff one file at a time and only with its predecessor but at least it supports a context output of sorts!
- [050614] Fixed **command-wait** when no arg is given to return when a callback sets the `.wait` variable. Command now returns imediately if the `.wait` variable is not defined. Fixed windows sleep function which was managing to break out before it should.
- [050612] latex added macros for bold/italic/mono character formatting.
- [050612] nroff region support for character formating.
- [050612] **.fhook-xxx.command-n** syntax added to **buffer-init** to simplify comamnd definition.
- [050610] Removed "**C-c C-g**" key binding in document macros and replaced with "**C-c <**" because of **C-g** issue.
- [050610] CVS added support for a **cvsco.etf** checkin template.

- [050610] First revision of ClearCase support, very incomplete and primative but the hooks required in the rest of the system are in place.
- [050609] Changed the directory listing so '`.`' is removed and `..` is always the first *file* and cannot be selected. Fixed some directory display annoying refresh issues.
- [050609] First implementation of directory deletion, copying and moving, including support for ftp directories etc. Areas of concern are still copied file's permissions and to a lesser degree ownership. Also the copies behaviour with symbolic links, this must be tested further.
- [050609] Fixed file permissions and ftp minor issues a couple of major ftp bugs and **$file-names** working for ftp directories.
- [050609] Added new **&stat** "`i`" flag and argument bit 4 to **file-op** to abort on a problem rather than prompt, it also returns an error code in *$result*.
- [050609] Changed the kill word commands so have the same word boundary behaviour as **forward-word** and Emacs (was killing word only then whitespaces only etc). Corrected word movement and killing command's EOB and BOB error handling.
- [050608] Added new **rdiff** (recursive diff) command
- [050608] Updated **string-rectangle** and **space-rectangle** to improve its usability.
- [050608] Updated **describe-word** setup comment.
- [050601] Corrected `ne.emf`, broke Window mode in disabling console mode for **ne -R** option change.
- [050601] Corrected crash on hilight with alpha-numerics of form `[[:digit:]]` when spelt incorrectly.

**8th Release - May 2005**
Release date specified as 2005-05-05.

- [050508] JASSPA MicroEmacs 8th release - final. Core and macro versions 20050505.
- [050514] Fixed ftp copy lock-up (most noticable on windows) caused by the response read at the start of a command also reading (and disgarding) the finishing message so ME hangs at the end of the command waiting for a response.
- [050514] Enabled ipipes under cygwin by default.
- [050514] Fixed win32s build.
- [050514] Fixed **mecygwin** interactive MS-DOS and cygwin BASH shells.
- [050513] Found the UNIX **waitpid(-1,...)** in the sigchild handler (which removes any zoombie processes) removes the pipe commands process so the pclose call cannot get the process's exit code (returns -1).
- [050509] Changed the name of aix43 make files to aix4 and sunos 55 and 56 files to a single sunos5 makefile
- [050509] Added support for **aix v5** (tested on aix v5.1)

- [050507] Added Finnish support to ne.emf.
- [050506] Improved python collapse and item-list support and added eaf and etf for python - thanks to Detlef.
- [050505] Base version numbers reset for next release.
- [050429] Added support for keybinding to item-list column ordering, function is item-list-order and is bound to 1, 2 and 3 by default. Thanks to Bryan Schofield for his submission in this area.
- [050429] Spelling - protect '.'s in the word with \\.
- [050428] Corrected TCL hilighting w.r.t. comment at end of line - submitted by Bryan Schofield.
- [050427] Major reworking of main commands in a directory listing buffer. Implemented typical features found in an OFM standard browser and create standard bindings as well as termcap friendly ones. Changed delete binding to delete files rather than close browser which is now bound to f10.
- [050427] osd - Added numeric bit 8 on osd-entry arg to put the text above the entry.
- [050427] Added C-x k support to browser mode to kill the buffer, also added hook for the file-browser's view file.
- [050427] Fixed user-setup to use only one value of $file-ignore & $kept-versions per platform rather than one for window and one for console based versions.
- [050427] Shrunk the size of the file-attrib dialog to ensure it fits on a termcap screen.
- [050427] Fixed nasty bug with transpose-line at the end of the buffer swapping the last line with the first!
- [050427] main.c - Fixed minor compiler warnings.
- [050426] Added a binding for file completion (any key bound to insert-tab) so OSD dialogs can do completion (although awkwardly).
- [050423] Fixed wiki edit mode for Russian keyboards, the 'k' flag key mapping was breaking things.
- [050423] Windows - Fixed windows Ctrl numeric pad keys, now come through as C-0 ... C-9 C-* C-+ etc.
- [050420] Added new Dot/Neato support fhook (for .dot files) - thanks to Detlef for the submission.
- [050420] TCL - Added more hilight tokens from Detlef.
- [050420] Document macros - fixed cpp->tcl typo - thanks to Detlef for spotting this.
- [050410] Fixed the -R option for DOS box, cursor same as the foreground color which is not ideal!
- [050410] Fixed the SunOS x86 makefile to disabled #pragam warnings under GCC.

- [050409] Release Candidate 2 - MicroEmacs 2005 (RC2)
- [050408] Added flag to the user-setup Tools to force the rereading of the current buffer's file, thanks to Bryan Schofield.
- [050408] Added new .item-list.width variable to allow control of the item window list - submitted by Bryan Schofield.
- [050408] Moved the generate tags into the sub folder and added a compile option to c and cpp - submitted by Bryan Schofield.
- [050405] Added support for DOS code page 866 (Cyrillic Russion) - submitted by Nicholas Kudriavtsev
- [050331] Added support for the setting of the history in macros via @h#.
- [050325] registry.c - removed compiler warning indicating porting problem hilighted under Darwin. Post increment of variable on left of assignment and same variable appeared on the right, split out the increment to reduce ambiguity of the statement.
- [050320] UNIX only - add a rule for .dir files found in the X11 directory. We do not want this files identified as "dir" info root file
- [050320] Added termcap (white) scheme for reverse video operation.
- [050320] Newuser registry file - added default entry for Apple Mac "darwin". Confirmed operation under Mac X-Windows.
- [050320] Extended ME exit commands to support the discarding of modified buffers '`0x8 discard modified buffers`' and '`0x10 discard modified buffers and create recovery file for changes`'.
- [050320] Corrected XML syntax for strings "`...`" to fix indent and hilight anomalies.
- [050319] Added **-R** command line option to reverse the built in video schemes - specifically for **ne**(1) also works with **me**(1).
- [050319] Corrected default mode settings. Recent removal of **letter(2m)**, **line(2m)** etc. changed the mode byte assignment of some modes. The built in global mode default was not updated for this change in bytes and incorrect global modes appeared as a result - specifically **quiet(2m)** which has jumped bytes.
- [050315] LEX/YACC template - Corrected LEX and YACC highlighting. Aligned C, C++ and YACC/LEX templates to be consistent.
- [050315] C/C++ templates - buffer setup modified for GNU Indent Style setting.
- [050315] Added new **browse**(3) command and binding to f3, changed existing collapse-all **f3** binding to **C-f2**.
- [050315] Modified ME exit commands to support a bitwise arg n where main **exit-emacs**(2) has '`0x01 Prompt`','`0x02 Save each buffer/reg/dic (with prompt if 0x01 set)`' and '`0x04 Prompt for exit code and exit ME with that code.`'

- [050313] Latex template - Corrected the latex-fill-paragraph. Set the fill-mode to '**l**' rather than '**L**' because its a pain when it selects centre rather than left and messes things up.
- [050311] Directory list - changed column sorting so Modified is sorted most recent first.
- [050311] Fixed perldb pipe operation.
- [050310] OSD simplified the main context menu and moved paste a cvs items in the directory context menu
- [050310] Fixed diff's file name input, was not completing correctly
- [050310] CVS changed the default setting of the recursive add to off.
- [050310] JASSPA MicroEmacs quick start document released.
- [050309] Fixed bug in regex multi-line search backwards, searching for "`foo[^a]`" is a mult-line as `[^a]` matches \\n, if "`foob`" exists at the start of the line isearch-backward refuses to move on.
- [050309] Catman pages added for ne(1) and me(1) complaints with gentoo as missing. Added to the package builds.
- [050308] Fixed a nasty bug with ftp connect dialog when the host has a port. Changed the default path to be nothing which becomes `/ /` to list the login directory. Change the tab order to not include the password as tabbing through the entry will lose the password, must now access via mouse or hot key which is not ideal.
- [050308] Added support for relative ftp browsing via new `ftp://???/ /???` url form. Added file name completion support for ftp
- [050306] Corrected the docmacro `http://` regular expression.
- [050306] Changed the date format of the **-V** command line option to include the century, same format as the **about**(2) buffer.
- [050302] HTLP and PHP - fixed **indent**(2) problems.
- [050302] Perl template - fixed indent problems caused by not skipping the = type regular expressions.
- [050302] TCL template - fixed continuation and comment indent bug.
- [050302] Added support for R/S-plus (.r) files - thanks to Detlef.
- [050225] HTML template - fixed an indent which was fixed at "`2`" rather than "`t`" and added comment skipping indent rules
- [050225] **dos2unix**(3f), **unix2dos**(3f) and **unix2win**(3f) - Removed use of **&spr** in favor of **&cat** so `dos2unix` etc. work in **ne**(1). [050225] Tracing a problem with hilighting a token of the form "`AB*$`", works if there is a '`B`' but fails if there are no '`B`'s, i.e. only matches "`AB+$`". Found that removing the end of line special case code fixes the problem. I believe the rest of the code has been improved to handle any end-of-line quirks - I hope so!

- [050225] Removed the 'start-up' macro function compile option so all versions support this - "ne" can now do a "@dos2unix"
- [050225] Regular expressions - extended replacement string syntax. Added support for "\c#" "\l#" and "\u#" to change the case of groups, where "\c#" is capitalise, "\l#" is lower "\u#" is upper and "#" is the group number (0-9, &)
- [050225] Regular expressions - changed syntax of "\s?" and "\S?" in favor of Perl's simpler "\s" and "\S" for a whitespace char class. Added new character classes "\d" = digits, "\D" != digits "\h" = hexdigits "\H" != hexdigits "\l" = lowercase "\L" != lowercase "\u" = uppercase and "\U" != uppercase. These changes were made to both the main regex search engine and the **hilight**(2) token matching engine. Added support for char classes within the main regex [...] character class, i.e. now supports regexs like "[\l\u_]" for any lower, upper or '_' char. This is not supported in the hilight which currently has a more simplistic "[...]" char class support.
- [050225] Regular expressions - changed use of "\s ", "\s-", "\S " and "\S-" classes to simply "\s" and "\S" which is perl regex compliant (ditching Emacs "\s" syntax as it is cumbersome and not very useful). The only nasty side effect is hilight's miss-use of "\S1" - "\S9" to mean not "\1" - "\9" group char, changed to "\!1" - "\!9"
- [050224] **ne**(1) set to not auto-save or create backups by default.
- [050224] Added "-a" and "-B" command line options to enable and disable auto-save **auto**(2m) and **backup**(2m) modes.
- [050224] Tcl template - some more fixes to the item list from Detlef
- [050224] **watch**(3) improved the refresh rate of a type 2 or 3 watch by ignoring the :watch-time and just look at the :watch-size. Fixed the split line issue with type 2 of 3 watching caused by last line not being complete.
- [050224] **M-x ftp** - stripped 'ftp://' and any trailing '/' of the users entered host name as these break ftp. Improved the dialog's handling of passwords, the user can now tell if a password has been set.
- [050224] Added standard binding for mark-function
- [050224] Fixed the ME token hilighting in the C hook
- [050224] Removed recently added %no_backup variable to disable backup create as new **-B** option is a much better way of doing it.
- [050224] edef.h - Removed definition of non-existant variable
- [050224] Fixed failure when trying to load "http://www.jasspa.com", there is an implicit trailing '/' which MicroEmacs now adds.
- [050224] Tcl template - major improvement - courtesy of Detlef.
- [050224] Fixed **collapse**to support the file hooks .exact setting (set to -1 means case insensitive)
- [050224] Added new **beginning-of-function**, **end-of-function**, **goto-function-line** and **mark-function macros** - thanks to Detlef.

- [050223] Windows MS-dev 6 project - corrected project settings for `_URLSUPP` to `_SOCKET` in keeping with the socket dynamic library loading.
- [050223] Added exec of wish to the tcl magic hook
- [050223] Fixed **$window-chars**(5) - added OSD default button chars.
- [050222] Fixed **$user-path**(5) being more that 35 chars breaking the **user-setup**(3) dialog.
- [050222] Directory listing - call `fhook-html` when file is a `http://` file - this is never a directory listing.
- [050222] Stopped loading of files in dir listing when clicking way to the right, must click on the name (MicroEmacs kept loading files when just trying to raise the window to the front - annoying).
- [050222] If 0 given to **watch**(3) call watch-time, this preserves the old interface.
- [050222] Added backup file disabling for **dos2unix**(3f) macros via command line defineable variable.
- [050221] Changed the new Windows process tree killing to use `LoadLibrary()` to detect whether the OS supports this feature (and NT does not). Also moved the URL support over to `LoadLibrary()` so we can remove the **-u** builds.
- [050220] Added UNIX Shell and Makefile hilighting to **notes**(3)).
- [050219] Fixed GNU GCC warnings in Makefiles for #pragma in X11 files under Sun Solaris and uninitialized variables.
- [050219] **osd**(2). Removed dialog '`F`' flag as it was not documented, did not work and only used in one obscure place. Fixed **C-g** and **Del** so they close up to the previous dialog but does close the main menu down in one call - was annoying.
- [050219] Release Candidate 1 - MicroEmacs 2005 (RC1)
- [050218] Corrected UNIX client-server socket file check – bad test on existence of connection file, using wrong variable with wrong test sense.
- [050218] Added site delete button to **ftp-connect** dialog.
- [050218] Fixed abbreviation and exec-string "`\\p`" and "`\\P`" store and goto position problem when the `C-x C-a` or `C-x a` keys are rebound. Fixed by replacing with '**esc x set-alpha-mark P**' (where **esc-x** is a safe internal special key).
- [050218] Updated **watch**(3) interface and made public. Created new **watch-time**(3) command and prompt user for watch type when no argument is given.
- [050218] Removed the use of bit `0x1000` in Ada string hilight as this is unlikely to be wanted (probably copied from Pascal).
- [050218] Changed the registry parser to abort on error rather than trying to carry on (a very painful process for a badly decrypted registry file).

- [050218] Fixed highlighting minor issues. **hilight**(2) scheme now returns back to EOL scheme after a bracket. A branch tag on an EOL line removes the EOL flag so it hilights correctly. Branch tag now supports the `0x1000` single line hilight. Lookback now supports `0x800` on a branch tag.
- [050217] HTML highlighting. Fixed some really grizzly scripted html page highlighting issues, predominantly with php.
- [050217] Change ftp so it opens the **ftp-connect** dialog if the current buffer is `*ftp-files*`.
- [050217] Fixed ftp dir list column detection and highlighting issue with windows ftp servers (time given as `9:15` not `09:15`)
- [050217] Fixed bug in **set-variable**(2) and **unset-variable**(2) when **&sprintf**(4) is used to create a variable name.
- [050217] Watch command (watch.emf). Improved the watch command, made '`0 watch`' call **watch-reset**, changed **watch-reset** so with an argument of `0` it removes all watches otherwise it resets the main watch counter and reports to the user which buffers are watched. Added new '`2 watch`' and '`3 watch`' which compares the file size, doing the following: **a)** file on disk is smaller than last watch - reloads whole file. **b)** file size is the same - does nothing. **c)** file on disk is larger - appends to the end of the buffer the tail of the file (`file_size - watch_file_size` bytes). This assumes that the files is always being appended (reasonable assumption in most cases. The only difference between 2 and 3 is that 2 forces a complete reread of the file on the first file change (ensuring the buffer is correct), 3 does not which allows the user to trash the first gigabyte of the buffer before starting the watch.
- [050217] Shell template - added local and function keywords and fixed a few string highlighting problems.
- [050216] Made the dir-listing's (Filtered) label on line 2 pick-able, changing the filter status when picked Made CVS's CVS-Quick label pick-able doing a full reread when picked.
- [050216] Fixed a couple of silly register bugs introduced with last **find-registry**(2) concat root and sub-key change. Improved behavior when ftp erf file is missing, now create a new one in memory and flags as changed so should be saved on exit.
- [050216] Do not abort `me.emf` execution if ftp erf file is missing or bad.
- [050216] Fixed by removing the '`startup`' function, the main init now uses dofile directly and it checks for either a NULL or `""` as a filname and replaces with "**me**" or ("**ne**" and does not complain) if found to be empty. This means that give **execute-file**(2) an empty string executes either "**me**" or "**ne**", i.e. resets me - nice side effect. Changed '`dofile()`' to `execFile()` and changed `execFile()` to `executeFile()` (as is function for command **execute-file**(2)) and change the other exec command function names for consistency.
- [050216] Fixed an annoyance reported by Bruno, ne should not complain if it fails to find `ne.emf` - fair enough.

- [050215] Fixed crash caused by using `-c` option with all files hidden (comment when browsing ftp dirs as cache dir lists are hidden) The startup process knows the history created buffers but the replacement was none of these and due to the toolbar it was a buffer without a file name and bang! Fixed in 2 ways, first check the replacement has a file name, second do not put hidden files into the history - they're hidden.
- [050210] Improved movement between items using the cursor in the search dialog.
- [050210] Fixed cvs dialog default settings.
- [050210] Corrected search path expansion, incorrectly handling a `PATH_CHAR` at the start of the string.
- [050209] Up-dated **user-setup**(3) platform page to new style and added **yank-rectangle**(2) binding.
- [050209] Fixed no-color termcap **$window-chars**(5) variable setting
- [050209] Fixed hot keys (were 3 'U's) in `osdmisc.emf`.
- [050209] Added new *BraceStatement* **indent**(2) value to support GNU C indentation style.
- [050209] Fixed the `osdDisp()` function required a `TTflush()` otherwise the cursor is left in the wrong position on termcap systems.
- [050209] Fixed the default **$window-chars**(5) variable setting.
- [050209] Fixed the DOS home directory setting and compiler warnings.
- [050208] Changed the C indent definition to be the same format as the existing indent definitions i.e. uses flag & indent value, this makes it more flexible and extendible. Updated macro files.
- [050205] Password generation macros added to `contrib` directory.
- [050205] Added JavaDoc highlighting to Java template. Enablement via **buffer-setup**(3). Removed the existing "`my`" extensions from contrib.
- [050205] Added **Doxygen**(1), <*stdint.h*> and error highlighting to C/C++ templates. Enablement via **buffer-setup**(3). Removed the existing "`my`" extensions from `contrib`.
- [050204] osd - Fixed problems with the *cursor shows current item* feature of osd caused by the cursor position going outside of the window.
- [050202] Added an initial item number to the favorite and buffer menus.
- [050203] Standardized narrow anchor and line flag movement behaviors with respect to deleting and inserting text. Enforced to effectively exist at character column -1 and been strict about maintaining a protected line. Made undo cope with markup lines that are lost. These changes may be a little to severe - we'll see.
- [050202] Fixed **C-g** close osd dialog crash.
- [050202] Directory listing - fixed nasty bug cause by the directory changing on disk before the hook has managed to do anything leading to a failure - this has been annoying Jon for some time!

- [050127] `hkemf.emf` - Fixed `!until` indent rule.
- [050127] First revision of **notes**.
- [050127] Fixed positional problems with mouse rectangle regions and added a mouse rectangle yank to user-setup.
- [050127] cvs - Set the initial osd dialog item to be the first item rather than the Okay button (with is still the default button).
- [050127] Fixed `*scratch*` hook assignment issues, initial startup buffer and only buffer delete problem.
- [050127] Added process tree killing for windows, tested on win98, 2k & XP, would be good to test on win95 (which should work) and NT which should not work but hopefully will be no worse than current version.

  This is a process tree killing for ipipes, in win32 killing the process we create is simply not good enough, all child processes continue unaffected. For example, run compile, kill the buffer in mid compile and it carries on behind your back!

  For Unix we use a new group and kill the group, DOS does not support ipipe. We could have used the new Job concept in ME but this is only present in win2000 & XP and the binaries are unlikely to run on win98 etc. This should run on every win32 platform although it is unlikely to do anything positive on an NT box (works in win2k)
- [050126] Changed the location of the **fhook-default** macro so its only defined if using **buffinit** (i.e. if running main `me.emf`)
- [050126] Fixed osd button repeat issue when current focus is an entry.
- [050126] Changed **insert-file-name** to insert buffer name if a -ve arg is given, bound to **C-x y** on the mode line, this replaces ml-lines **esc y** feature.
- [050123] Fixed a crash caused by too many closed fences, the recursive nature of the `findfence()` function triggers a stack crash at around 22000 on windows. Set limit to 256 nested fences.
- [050122] Fixed **dir**(2m) mode loss on a directory re-read, this caused a number of side effects including loss of highlighting and user prompt to reload the directory when it had changed.
- [050119] Removed the "*sub-key*" argument from all registry related calls (see **&reg**(4) and **set-registry**(2)). The registry calls used to comprise two arguments "*root*" and "*sub-key*", the "*sub-key*" is really redundant and the same effect may be achieved by a simple string concatenation of the "*sub-key*" to the "*root*".
- [050119] Added "`my-*hook-always`" macro checks in the buffer hook switching of **buffer-init**(3) to permit global file hooks to be defined, this allows a file hook to operate on all buffers. "`my-*hook-always`" is executed after the language file hooks of "`*hook-XXX`" and "`my-*hook-XXX`".
- [050116] Added "`TODO`" highlighting rule for Makefile comments.
- [050114] Fixed **narrow-buffer**(2) type numeric argument not processed correctly.

- [050113] Added tool bar tool mouse select region command.
- [050113] Added bindings for the cvs and ftp console tools.
- [050113] Fixed **organizer**(3) duplicate index bug.
- [050111] Deleted the **get-registry**(2) command, this has been depreciated and **&reg**(4) should be used instead.
- [050111] Extended the **read-registry**(2) and **save-registry**(2) commands to permit reading and writing from a buffer based registry (rather than file).
- [050111] Extended **list-registry**(2) to permit listing of specific nodes, the output buffer of the registry listing may be specified. Added a invisible flag to a registry node which hides the node completely (but can be toggled).
- [050111] Fixed windows server command file polling of up to 60 times a second by making the adder of the command responsible for 'waking' MicroEmacs by sending a `WM_USER` message. This is not a problem for `UNIX` as it is possible to wait on file input and `EOF` is not reported.
- [050105] Fixed several narrow-to **narrow-buffer**(2) usability issues, with these fixes the top and bottom narrow cannot be removed without a markup line and the order cannot get muddled.
- [050105] Fixed a problem with the anchor position up-date when inserting a whole line.
- [050105] Changed the osd entry mouse handling to always refocus, this allows the user to change location in the current entry using the mouse.
- [050105] Fixed. **insert-macro**(2) undo location was incorrect causing a crash.
- [050103] Spelling dialog. Changed the Ignore button to Continue if word becomes correct without changing (word added or language changed).
- [050103] Fixed. CVS osd dialog with bad hot-key.
- [050102] Added setting of **$recent-keys**(5) variable.
- [041231] Variable **$fmatchdelay**(5) renamed to **$pause-time**(5). *$pause-time* is now used for other MicroEmacs delay times when the user is prompted for information.
- [041224] Fixed several usability issues in the search replace dialog.
- [041224] Up-graded printer-setup dialog to new style OSD.
- [041224] CVS dialog - added read-only option to checkout and update.
- [041224] Fixed **goto-line**(2) offset which as always going to the previous line (one out). **u** [041221] Added support for proper kill buffer use in message line, i.e. **set-mark**(2), **kill-region**(2), **copy-region**(2), **yank**(2) and **reyank**(2) all work in the same way as a normal buffer.
- [041221] Fixed registry node deletion when the node is modified.

- [041221] Removed **osd**(2) `osd 0` '*d*' flag and added '*I*' initial item and '*D*' default button flags, existing osd dialogs which use '*d*' will fail to load and need are migrated to use the *D* and *I* flags.
- [041218] **auto-spell**(3) key bindings are promoted to global so its easy to enable in any buffer.
- [041218] Corrected setting of the **$home**(5) environment variable to handle MS-Windows '\\'s, auto terminate with '/'.
- [041210] Added Enable/Disable Auto-Spell to the context menu.
- [041210] Rearranged the **organizer**(3) dialog to be a little easier to navigate through.
- [041210] Fixed nedit emulation insert binding.
- [041210] Removed **ml-clear**(2) command, replaced with **ml-write**(2) option "`0 ml-write`".
- [041210] Removed the **change-directory**(2) command, a change in directory is achieved by changing the **$buffer-fname**(5) value such that the path defines the location of the command. All macros operated relative to the buffer location and never explicitly changed the directory location.
- [041210] Fixed CUA emulation insert binding.
- [041210] Deleted **letter**(2m) mode. Use the default binding of **forward-delete-char**(2) for letter mode off, where single deleted characters are to be added to the kill buffer then the existing **letter**(2m) mode may be achieved by binding as "`1 forward-delete-char`". Similarly for backspace, **backward-delete-char**(2).
- [041210] Deleted **line**(2m) mode. Use a binding with **kill-line**(2) of "`1 kill-line`" to mimic the line mode enabled and "`kill-line`" for line mode disabled.
- [041210] Deleted **cmode**(2m). Electric-c is now configured with **indent**(2). C and C++ indent styles can be different, can also have a GNU C style etc, can change the indent width as new indent rule supports '*t*', consistent **$buffer-indent**(5) interface.
- [041210] Created new osd file "`osdmisc.emf`" for miscellaneous OSD macros, this removes the OSD dependency when the MicroEmacs is executed in special start up modes with startup files. Moved search and new **indent-setup**(3) commands out of "`search.emf`" and "`format.emf`" into a new "`osdmisc.emf`" as they cause problems with utils that do not run "`me.emf`" (e.g. "`ehftools.emf`")
- [041210] Introduced a GNU emacs emulation mode, emulation file is "`meemacs.emf`".
- [041209] Fixed **newuser** "`$user-name.emf`" conflict issue with an existing macro (e.g. `zaurus.emf`) by testing for it and prefixing a digit to make the file unique.
- [041209] Added `esc C-\` binding to **restyle-buffer**(3).
- [041209] Fixed some issues with the Tools user-setup dialog.
- [041128] Added MinGW makefile posted to JASSPA group by Patrick Anderson, forms part of the release makefiles.

- [041126] Added 'k' keyboard setup to **set-char-mask**(2) which is used to translates foreign keys to a latin char.
- [041126] Changed the internal LEADER byte to `0x1c` as `0xff` is a valid Russian char and it badly breaks *$vars* and *macros*.
- [041126] Fixed spell dump word which was not translating the words to the users char-set.
- [041126] Added support for Russian language.
- [041126] Improved the aff ispell conversion routines (almost works automatically).
- [041126] Stopped **auto-spell**(3) from triggering the no dictionaries error, leave it to **spell-buffer**(3) etc.
- [041125] Fixed OSD multi-line `C-k` **kill-line** behavior and implemented "`-1 kill-line`" support.
- [041125] Added basic support for Verilog programming language.
- [041123] Fixed ftp OSD (auto entering entries broke the password field) and modernized.
- [041028] Fixed. `*find*` buffer exhibits the same behavior as `*grep*` buffer. i.e. the `*find*` buffer remains on screen and a pop-up window is used to show the file.
- [041028] Fixed. `*grep*` buffer does not load file when selected with <RETURN>.
- [041027] Spruced up old "`apache.conf`" highlighting rules and added to release.
- [041022] Fixed very obscure cvs load-up bug.
- [041122] Updated schemes to set '*Show cursor in osd*' suitably and to make sure the cursor is not red as this conflicts with the match fence display.
- [041120] Fixed **transpose-lines**(2) crash when **undo**(2m) mode is disabled.
- [041120] Removed the OSD explicit tab order value as this was not constructively used and causes many item order problems.
- [041119] Many improvements to OSD dialog user interface following suggestions for news group. Updated most of the osd dialogs to use the new `[*]` check-box chars. Fixed tabbing orders and moved some items over to combo-boxes. Added osd-cursor visibility support to scheme-edit. Made the displaying of the cursor in an osd dialog a **$system**(5) bit set by the highlight scheme and fixed the position when in a child dialog.
- [041118] Added a macro recursion depth limit of 20 to avoid blowing up MicroEmacs when the user has created an infinitely recursive macro.
- [041112] Added Paul Boekholt's suggested GNU Emacs key bindings.
- [041107] **kill-paragraph**(2), **fill-paragraph**(2), **forward-paragraph**(2) and **backward-paragraph**(2) commands redefined. **kill-paragraph** now kills forward and backward rather than control the kill buffer. Paragraph movement now defined blank-line to blank-line rather than the first/last paragraph line.

- [041105] Fixed **kill-line**(2), **forward-kill-word**(2) and **backward-kill-word**(2) numerical argument implementation.
- [041104] Removed **regex-forward**(3) and **regex-backward**(3) macros and moved uses of them over to **search-buffer**(2).
- [041104] Added **search-buffer**(2) command which replaces the macro commands **regex-forward**(3) and **regex-backward**(3). Simplifies macro processing of searching by providing a search that explicitly sets the **exact**(2m) and **magic**(2m) modes. The default is to respect buffers exact and magic mode and flags must have `E` or `M` to ensure they are disabled.
- [041104] Changed any use of **kill-line**(2) to "`1 kill-line`" as this should maintain its required behavior. A **kill-line** with no numeric argument kills to the end of the line but excludes the end of line character (GNU Emacs compatibility). The default binding is the "`1 kill-line`" form, re-bind the key to "`kill-line`" if the end of line is not to be killed.
- [041104] Removed **line**(2m) and **letter**(2m) buffer modes as these can be done using numerical arguments to bindings. See the the **kill-line**(2) binding which replaces *line* mode. See **forward-delete-char**(2) and **backward-delete-char**(2) binding which replaces *letter* mode.
- [041104] Changed the **forward-kill-word**(2) and **backward-kill-word**(2) commands to be more consistent with GNU Emacs.
- [041102] Moved winterm over to using `addKeyToBufferOnce()` for mouse move and time, also done same for UNIX mouse time event. This minimizes the number of mouse moved events that are processed by the system.
- [041101] MS Windows. Fixed mouse cursor disappearing after using a *Shift* or *Control* mouse button event.
- [041101] Auto reload the directory even if it is changed (`hkdirlst.emf`).
- [041030] MS-Windows. Removed the "`WM_INPUTLANGCHANGE`" message handler as this did the wrong thing (use `Alt-Shift` to change between two different keyboards e.g. US and Russian)
- [041030] MS-Windows. Changed the method used to get the APPDATA path to use `SHGetSpecialFolderLocation()` as this is available in all versions of Windows.
- [041030] UNIX. Fixed the mouse movement which caused a rendering anomaly. Mouse moved commands are only added to the key buffer if the key is not already present in the key queue. This alleviates a large build up of redundant mouse move keys on the queue. (Added function `addKeyToBufferOnce()`),
- [041030] Added mouse rectangle region selection methods, **copy-rectangle**(3) and **kill-rectangle**(2).
- [041029] Hardened **cpp**(9) C++ commenting on `//` alignment when the previous line is a solid `//////////` when re-styling buffer.

- [041029] Fixed **find**(3) "`*find*`" buffer operation so that it operates like "`*grep*`" on file selection. Keyboard file section enabled within the "`*grep*`" and "`*find*`" buffer.
- [041026] Spruced up old "`smb.conf`" highlighting rules, brought them up to date with latest keywords and added to release.
- [041026] Fixed **indent-setup**(3) osd dialog.
- [041024] Macro/source compatibility macro and source versions updated to "`20041001`".
- [041024] Fixed the **cvs-diff**(3) options, local with HEAD was not correct.
- [041024] **latex**(9) template extended. Buffer bindings for underscore and quote keys. Search tools to determine current context.
- [041023] Extended the buffer setup for the **indent**(2) mechanism. May now specify the default buffer *indent-width*and *tabsize* in the **buffer-setup**(3) dialog.
- [041022] Revised indentation implementation. The templates now define indent width independent rules. The **indent**(2) command '*indent*' parameter may be defined as an absolute indentation or using a '`t`' notation. The syntax is "`[-][[indent][/indent]]t`" which defines a vulgar fraction of '`t`' which is the indent width (or tabsize) as specified by **$buffer-indent-width**(5) defined below.

  The global variables **$tabsize**(5) renamed **$indent-width**(5) and controls the global indentation size. **$tabwidth**(5) renamed **$tab-width**(5) and controls the display size of a literal TAB character.

  The indentation is now specified on a buffer basis, the variables **$buffer-indent-width**(5) and **$buffer-tab-width**(5) define the indentation and TAB width on a per buffer basis. These are seeded from the aforementioned global variables.
- [041022] Fixed spelling mistake in user interface of XML macros.
- [041022] Removed CVS "`Log:`" field from all of the template files - undesirable.
- [041022] Added HTTP highlighting to the text document template.
- [041021] Enhanced Wiki language template.
- [041021] Added Windows CUA emulation mode.
- [041019] Fixed insert binding for Nedit emulation mode
- [041019] Fixed **perl**(9) highlighting rules. Using incorrect syntax.
- [041016] Fixed UNIX make files for building NE in debug mode.
- [041016] Fixed termcap entry for ^@ key (`C-space`). (Thanks to Bob Paddock for noting ommission)
- [041016] Added **M-%** binding to **query-replace-string**(2) as built in binding. (Thanks to Bob Paddock)
- [041016] Added Ruby programming language support.

- [041015] Corrected problem with directory listing loosing highlighting (a **@mna**(4) problem).
- [041014] Within OSD, fixed spelling mistake. *Loose -> Lose*.
- [041014] Fixed **delete-hilight-lines**(3) failure when buffer file has changed.
- [041009] Fixed osd entry scheme as the cursor was not visible in MS Dev-Studio Colors.
- [041009] Added Wiki language template
- [041003] Wheel mouse binding added as default.
- [041003] Catch long lines from cut and paste, resolve crash.
- [041002] Added X-Windows drag and drop support using **xdnd**.
- [040930] Added HTML, XML and SGML close bracket auto expansion.
- [040930] Added XML and HTML verification macro. Added XML utilities macros to allow next and previous tag to be matched. The XML utilities are also used by SGML.
- [040930] Added `xxxx-pre-expand-abbrev` hook so that macros may add their own expansion methods.
- [040917] Added <`...`> as supported fences.
- [040917] Fixed the `-1` **add-next-line**(2) ignore line feature.
- [040915] **cvs-add**(3) looks for a Tags file in the CVS directory and sets the GUI state appropriately. Added server to the **cvs-status**(3) line mangling.
- [040909] **cvs**(3) GUI updated with new features. Added repository set up to the UI. Reduced wait time on long commands by using ipipe. Allowed `-z` option to be used with **%cvs-com**(5).
- [040828] About page includes the century in the date (suggested by -Th).
- [040824] Fix for man page hyper-link when includes `-/./_` characters.
- [040814] Linux manual page formatting fixes.

**Minor Patch - July 2004**

- [040728] Released patches.
- [040728] Corrected Linux kernel 2.6 pty allocation.
- [040630] Linux manual page formatting fixes.
- [040518] Comment definition corrections for C, C++ and java, notable Doxygen comment formation omitted.
- [040518] Corrected block comment formatting, adding an extra space at the end of the comment.

### 7th Release - March 2004

- [040326] Release wrapped - public release tag is **20040301** for executable and macros.
- [040326] Support for MAC O/S Darwin image. Console only (no build machine available for X-11 version).
- [040326] Introduction of UPX on Windows and DOS executables for **ne** only.
- [040326] Support for MS-DOS long name under DJGPP v2.0. For version 2.0 then the DJGPP functions must be invoked rather than the MS-DOS BIOS calls otherwise the modified short name is returned. DJGPP v1 is still supported original definitions maintained.
- [040326] Added **filetype**(3f) macro to determine the file type of MS-DOS, Windows or UNIX.
- [040322] *$mouse* variables configured under console modes to allow pseudo mouse operations to be performed.
- [040322] Fixed the **win32s** build and emain.h definitions. Mouse was not supported under *win32s* through incorrect configuration definition.
- [040322] Key bindings for the file browser. `ENTER` now handled on the reserved fields to sort. `BACKSPACE` now moves up a directory level.
- [040322] Changed the interface to **cvs**, **ftp** and **file-browser** command so that if a numeric argument of `0` is given then the directory tree window is not opened, just the `*files*` buffer.
- [040322] AutoIt hook. Thanks to Dr Detlef Groth.
- [040322] Added CSS template. Thanks to Gabriel Tabares-Barreiro.
- [040318] Fix to Microsoft Windows clipboard when operating with VNC.
- [040317] Macro and source release set to "`20040301`".
- [040317] Added **find**(3) command to macro set.
- [040317] Corrected **rgrep**(3) to use the **%xfind-com**(5) to specify the **find**(1) command.
- [040310] Fix to graphical diff - **gdiff**(3). Occasions where all changes had been accepted and the changes could not be saved. gdiff incorrectly prompted user to accept all changes first.
- [040310] Support for multiple comment styles in the same language template. Additional comment styles are defined in the "`hkXXX.emf`" file with the syntax "`.fhook-XXX.comment-`
where "`N`" is an integer starting for *1* to *n*, the number of different comment styles. Templates updated to use multiple comment styles where appropriate (C, C++ and SQL).
- [040310] Support the '*r*' flag in "`.fhook-XXX.comment`" flag field. Causes the key binding to "`esc-o`" and "`esc-q`" to restyle the comment. All templates updated to include the flag where appropriate.

- [040310] **cvs**(3) operates asynchronously using ipipes. Refresh problems resolved.
- [040310] **narrow-search-forward**(3) bound to "`C-c s`", **narrow-search-backward**(3) bound to "`C-c r`".
- [040308] **ipipe-shell-command**(2) includes additional flag to run in "*raw mode*" where the buffer is not annotated and wrap is disabled. Additional argument added to command to allow the **$buffer-ipipe**(5) macro to be defined when the pipe is defined. Corrected operation of **pipe-shell-command**(2) where *ipipe* automatically degrades to a *pipe* (Windows only), if degraded then the **$buffer-ipipe**(5) macro is still invoked on completion.
- [040226] Deleted "`cygnus.emf`" added "`cygwin.emf`" for **cygwin**(3) BASH shell support under Windows. Corrected the shell operation with macro layer for echo and completion.
- [040224] Add new *narrow out unhilighted lines* feature.
- [040223] Add new buffer tab list toolbar tool for rapid access to buffers via a top or bottom toolbar.
- [040223] Add <> buttons to tool-bar mode lines to scroll left and right.
- [040223] Fixed. Calender day is off by one when going backwards to before 2004.
- [040223] Fixed. Executing 'me' (**user-setup -> Current**) breaks **auto-spell**(3) until a main spell command , **spell-word**(3) or **spell-buffer**(3) is executed.
- [040223] **cvs**(3) directory listing file loss solved. Differences between CVS command line operations and directory list operations resolved, both now produce the same results.
- [040229] Fixed UNIX signal lock synchronization, solves problems with read failing following FTP or HTTP command.
- [040223] Add new **narrow-search-forward**(3) and **narrow-search-backward**(3) commands.
- [040220] Mouse wheel operation fixed. Added new **mouse-wup-scroll** commands that should be used instead.
- [040218] Added '*Invert Selection*' to context menu.
- [040218] Add '`f5`' refresh binding to a dir list buffer.
- [040218] Fixed. Need some way of opening the right mouse button menu using a key. Also need so termcap bindings to open main and context menus. Created new context-menu command, bound to `S-f1` and the windows 'menu' key (aka windows) and created binds to open main menu (**esc =**) and context menu (**esc +**) on termcap platforms.
- [040217] Fixed. **2** **popup-window**(2) does not work on an XTerm, no implementation in function `meFrameXTermMakeCur`.
- [040217] Fixed. CVS status of directories is not shown. Also add '`b`' and '`o`' file type flags.

- [040216] Fixed. ME **build.bat -l** options do not work in standard cmd/command as there use 4DOS '>>&' extension. Changed to using `2>&1` which works on cmd & 4nt but breaks 4Dos and command.
- [040216] Fixed bottom Toolbar window grows when CVS is opened, **change-window-depth**(2) does not respect bit 2 of **$window-flags**(5).
- [040214] Added **.spec** template for RedHat RPM package file specification.
- [040214] Added **.psf** template for HP-UX depot product file specification.
- [040214] Recognize `-*- makefile -*-` and `-*- sh -*-`.
- [040213] Fixed reframe *cvs-console* better so the exit status can be easily seen from a small window.
- [040213] PHP script can be started with '<?' tag in a php file.
- [040213] Corrected UNIX ishell operation when "**su**" to another user. Locking up or generally behaving badly (termio not initialized). Affects all UNIX platforms.
- [040213] Documented the create/delete/next-frame keys.
- [040210] Fixed toolbar Directory File List does not contain "`../`" - UNIX only.
- [040210] Fixed. On UNIX only the **$progname**(5) may not be fully expanded (has just "me") which may lead to a ctags generation error and an alternate fix to **$search-path**(5).
- [040210] Fixed. On UNIX, **find-file**(2) of a symbolic link which points to a directory results in a bad dir listing. The list of files is correct but the statistics are all wrong (found on SunOS 5.8).
- [040210] Fix subtle ftp bug with alias ftp addressed. Create an ftp site "`ftp://test/`" say with user name foo and password bar and then load in "`ftp://test/dir1/`" the buffer file name is correctly set as "`ftp://test/dir1/`" but the 'Directory listing of:' line is set to "`ftp://foo@test/dir1/`". Firstly the user may wish the '*foo*' to be hidden but secondly with the introduction of being able to click on the '*Directory listing of:*' line this has introduced a subtle inefficiency/bug, if I click on the '`foo@test`' ME will load "`ftp://foo@test/`" not "`ftp://test/`" which to ME is a completely different file. This can in turn lead to "`ftp://foo@test/dir1/`" being loaded so you can end up with 2 copies of every dir being loaded! The fix is to make the '*Directory listing of:*' line the same, i.e. "`ftp://test/dir1/`"
- [040208] Fixed. "**-n**" command line option to a console build causes a option failure and the executable does not run.
- [040208] Corrected UNIX jasspa tree search when executable is in CWD and there is no executable path.
- [040206] **RC4 - Release Candidate 4 - MicroEmacs '04**
- [040206] Fixes for tool-bar Buffer Item List update problem. Thanks to Dr. Detlef Groth.
- [040206] Added template support for XML. Thanks to Dr. Detlef Groth.

- [040204] Re-worked Microsoft Windows me32.ini file specification. Order revised and lookup simplified.
- [040203] "*user-setup*" changed name to "*Rebind Home Keys*" for re-binding of `home` and `end` keys to start/end of line. Removed "`C-v`" binding that existed with old "*MS keys" label*.
- [040203] Reworked all user defined defined directory save locations, ftp, spell, etc..
- [040203] Corrected osf1, openBSD and zaraus emf files.
- [040203] Added **%cvs-com**(5) variable to define the CVS command.
- [040203] XML template extended to include javascript.
- [040130] 2nd iteration of prototype and near final revision of new user setup.

  Following an internal review of RC3 then it has been decided to address the issues of user configuration prior to a full release. This specifically addresses the location of user MicroEmacs configuration directories and search paths. User configuration has traditionally required the setting of external environment variables such as "$MEPATH" and for the user to create a private directory to store user settings. This is proving difficult for new users who just want to run the editor. RC4 and subsequent releases will address these problems by determining the correct location for configuration files and create the appropriate files and directories under user control.

  The net result of these changes should significantly simplify installation and setup as follows:

- All platforms will be able to run without any manual user setup or the requirement for specific MicroEmacs environment variables.
- User defined directories and configuration files will be created under editor control on detecting a new user.
- Existing installations using `$MEPATH` will be backwards compatible with the new build.
- All platforms now support a searching jasspa sub-directory structure of **company / spelling / macros**. This was traditionally limited to the Windows Install Shield installation but has been extended across all platforms.
- Two new environment variables are provided for user or system administrator use. **$MEINSTALLPATH**(5) defines the location of the MicroEmacs standard macros and allows the built in search path to be overridden. **$MEUSERPATH**(5) defines the location of the user private configuration directory and allows the user to locate their private directory in different location. In a standard install neither of these variables will be required.
- The Microsoft Windows system defined application path will be used for user configuration files. This is applicable for roaming users where the application directory is stored on a server.
- The use of the me32.ini file on MicroSoft Windows environments is not required, but is still supported.

- The UNIX path searching is improved. `/usr/local/microemacs` will be depreciated and replaced with `/usr/local/jasspa` in addition other standard path locations are supported on a per-platform basis.
- On UNIX then the directory `/.config/jasspa` shall contain the user configuration files. Optionally the **company**, **spelling** and **macros** sub-directories may be included in this directory where access to the system directory locations by the user is not permitted.
- Concept of guest has been removed. `guest.emf` and `guest.erf` removed replaced with *newuser*. **user-setup**(3) no longer supports setting of name.
- Internal variables **$user-name**(5) and **$user-path**(5) are defined.

- [040128] Fixed re-style for C/C++ comments in vertical alignment.
- [040123] "*user-setup*" mouse button selection fixed.
- [040123] `C-x 3` mapped to **split-window-horizontally**(2), `C-x 5` mapped to **create-frame**(2).
- [040122] Added support for Zaurus, thanks to Pierre Gaston
- [040122] Fixes for non-US keyboards under X-Windows, thanks to Pierre Gaston and Dr. Detlef Groth
- [040122] Added support for OSF1, thanks to Dr. Detlef Groth
- [040122] Added support for OpenBSD
- [040122] **ml-bind-key**(2) supports new binding for next/previous file, history search uses `up`/`down` `C-p`/`C-n`, prompted by Gabriel.
- [040122] New key bindings for **create-frame**(2).
- [040119] **RC3 - Release Candidate 3 - MicroEmacs '04**
- [040119] Fast exit out of fold fixed.
- [040119] Added **copy-rectangle**(3) and **yank-rectangle-overwrite**(3) commands.
- [040119] Collapse line format now includes the end token so that indent operates correctly following a fold.
- [040119] **narrow-buffer**(2) numeric argument and command line options extended for mark-up lines.
- [040119] `spawn.c` compilation warnings removed.
- [040119] Updated templates for Python and AsmX86 - thanks to Pedro Gomes.
- [040119] Added "`*compile*`" hilighting and `ENTER` selects error file and line - prompted by Salman Khilji.
- [040119] Fixed the **c-eval-defines** region hilighting problem.
- [040119] Updated the Lau HTML processing tags - thanks to Jeremy Cowgar.

- [040116] Latex ”item-list” and ”template” updated - thanks to Dr. Detlef Groth for diagnosing problems
- [040116] Perl and TCL abbreviation files - thanks to Dr. Detlef Groth
- [040115] **open-line**(3) added for correct binding of ”`C-o`”.
- [040113] **restyle-buffer**(3) fix when mode defined - thanks to Pedro Gomes.
- [040113] **create-frame**(2) canvas size fix on creating internal frame
- [040112] Highlighting fix for tool-bar Dir Files list with symbolic links
- [040109] **RC2 - Release Candidate 2 - MicroEmacs ’04**
- [040109] PHP template added.
- [040108] **RC1 - Release candidate 1 - MicroEmacs ’04**
- [040108] Shrink line crash on SunOS and IRIX fixed
- [040108] Fixed rendering anomalies of menus and pop-ups on Windows XP ClearType when anti-aliasing is enabled.
- [040106] Fixed comment re-styling for single start of line comments (REM in MS-DOS)
- [040104] Macro execution speed-ups - Thanks Hops for analysis.
- [040104] Code optimizations and extensions - Thanks Peter Smith.
- [040104] Lua language template - Thanks Jeremy Cowgar.
- [040104] SGML language template.
- [040104] **string-rectangle**(3) command added - Suggested by Salman.
- [040104] C++ re-style fixes - Problems reported by Salman.
- [031221] X-Window fixes - Thanks Jeremy Cowgar.
- [031120] Added new highlight schemes ”*Lemon Jelly*” and ”*Brownie*”.
- [031120] CVS. Fixed several issues with the why the cvs dir list processed the missing and added files.
- [031120] HTML Template. Improved indentation.
- [031120] Toolbar. Added ”trunc-sheme” reset to force a redraw if required.
- [031120] Fixed the games to cope with the toolbar by closing it.
- [031120] `buffer.c`. Changed the ’*n*’ arg to list buffer so user can select hidden or verbose etc. Added check after exec of ”ehook” to catch nasties, not fool proof but catches attempts to close tool bar etc.
- [031120] Fixed a **goto-line**(2) bug shown when using grep on a single lined file (like CVS/Root).

- [031120] `hilight.c`. Changed indent line count to use the close so values of greater that 256 can be used.
- [031120] `input.c` Added support for EOB & BOB keys & fixed fwd/bck word movement across lines.
- [031026] Fixed -ve re-center bug found by Alexander Doktorovich.
- [031026] Last commit to search fixed a **&xsequal**(4) failure on start-up, was not setting the internal flags correctly. Fix go a little messy as it should not affect the hunt commands.
- [031026] Minor restyle.
- [031026] Fixed a nasty buffer reload bug.
- [031026] Fixed lookback highlight bug.
- [031026] Fix for Sunos Xterm function keys.
- [031026] Added support for a look-back indent scheme which can be used to get the correct indent scheme to use.
- [031026] Fixed history line numbers (were one line out for non-current buffers).
- [031026] Changed the `mldelete` to respect the "scrollFlag" setting, if smooth scroll enabled then delete text smoothly.
- [031026] Fixed narrow look back highlight bug.
- [031026] Added ADA file hook - thanks to Gabriel Tabares-Barreiro.
- [031026] Nroff template. Removed unused variable.
- [031026] HTML Template. Major shake-up for HTML base files including major enhancements to indentation and current script auto-detection.
- [031026] Toolbar. Added support for the new "**:chook**" current file type variable and got toolbar and buffer tool menu to call the update command (toolbar only on a "`C-l`").
- [031026] CVS. Fixed UNIX checkout path and implemented cvs remove.
- [031026] **page-file**(3). Fixed backward isearch blow out.
- [031026] Perl Template. Updated the perl.eaf to <FOLD>.
- [031026] Abbreviations. Added support for the new "**:chook**" current file type variable and added support for an INCLUDE tag.
- [031026] Abbreviations. Improved the look-back abbrev to expand to one before if exec'd again.
- [030917] HTML Template. Fixed a hilighting and indent problem caused by embedded script.
- [030917] OSD. Changed OSD hot-key '**&**' to new style '**\H**'. Changed user-setup dialog to use new Combo-box.

- [030917] CVS. Major revamp of CVS support including an OSD dialog for each tool.
- [030917] Changed the CHAR defines and add defines for the LEADER and TRAIL bytes used for special chars. Re-implemented OSD hot-keys to use "**H**" instead of "**&**" and added new "**{**" and "**}**" to stop and start current item hilighting. Added support for C++ private public and protected token indentation. Added new Check-box chars to the extended chars (1,2,3,4,19 & 20) and improved the drawing of some of the others Fixed windows "close while minimized" test so only prompted if something has changed.
- [030917] Fixed a `build.bat` not setting MAKE to nmake problem.
- [030917] Added support for new Check-box chars in extended chars.
- [030827] Added use default printer setting.
- [030827] **page-file**(3). Fixed header and added isearch support.
- [030821] Rationalized the buffer sizes, increased the keyboard macro and search string buffer sizes. Removed the GNU regex code - not used or tested in years. Part one of Jon's Xterm key lookup fix to get the key pad working.
- [030710] User setup. Updated osd calls due to the change in '**d**' flag behaviour and changed the items with drop down menus to use the new combo-box.
- [030710] ODS. Updated osd calls due to the change in '**d**' flag behavior.
- [030710] Added new define for the number of history types and added new **@h0**(4) history variable.
- [030710] Added support to **@mn**(4) and **@mna**(4) in the "`meGetKey`" routine
- [030710] Added new undo type for line reordering undo's, used by **sort-lines**(2) and **transpose-lines**(2).
- [030710] Changed some define names to make them compatible. Changed the calls to "meGetUserKey" to "meGetKey" so a macro can give a list of keys Added new 0 argument which quits at the EOB rather than going to the top of the buffer etc (used in **page-file**(3) "isearch").
- [030710] Changed the logic of the '**d**' flag so that it disables the displaying of the item type, changed button rendering so that it adheres to this so a single char button can be created Added new '**O**' combo box item type which is a mixture between an entry box and a sub-menu item.
- [030710] Improved the special char drawing routines so that they remain inside the char space and triangles look good regardless of size.
- [030710] Fixed a nasty bug to do with file hooks that load other files and the find-file command manipulating the buffHistNo incorrectly so the wrong buffer is shown - obscure!
- [030625] Added macro release makefile.
- [030625] MicroEmacs 3.8 compatibility. Fixed isearch emulation.

- [030625] First revision of **page-file**(3).
- [030625] CVS. Removed debug line.
- [030610] Changed `TAB` and **BELL** to safer defines. Optimized the fill-para function so that it keeps one of the existing spaces between the words, this also help preserve the position of anchors.
- [030610] Package making files for SunOS.
- [030608] Changed the mouse default behavior to do a move yank.
- [030608] Tools. Removed superfluous endif.
- [030608] Toolbar. Only update the tool-bar if its the top level command, i.e. the screen-update is not required due to spell-buffer (say).
- [030608] MS-Shift region. Check for **kill-rectangle**(2) and treat like we do the **kill-region**(2).
- [030608] Added new -ve argument to **insert-file**(2) which inserts only a section of the file. Optimized the **file-op**(2) move command to try just moving the file first.
- [030608] Changed the addLine function so it always flags the line as changed, this avoids some bizarre issues when macros delete an insert lines of text (best shown by the buffer-info tool-bar tool).
- [030608] Added new macro oriented buffer-info numeric argument. Fixed bug in the ignore white-space buffer compare.
- [030608] Fixed default printer selection issues - not complete.
- [030608] Fixed a minor debug build issue in undo.c.
- [030608] Changed the history so it always removes any duplicate entries.
- [030608] Added **$window-eline**(5) and moved the aline into extended version.
- [030608] Changing the addLine function to always set the changed flag meant all macros became hidden, use different bits.
- [030608] Fixed a minor compile warnings.
- [030608] Fixed buffer reload a file while displayed in another window problems.
- [030519] Fixed minor compiler warning in search.c by removing pointless static initialization.
- [030519] Fixed poke over-writing cursor on win32 console.
- [030516] C Template. Changed C #define evaluation to use new protected narrows.
- [030516] Stopped the Organizer Find exit quitting the organizer itself.
- [030516] VM Mail. Fixes from Tom Hundt fixes message separator problem.
- [030516] Made MS-DOS redraw the cursor on a poke as well.

- [030515] Organizer. Fixed bugs with creating new addresses and indexing etc.
- [030515] Printer. Store the windows device information in the print-history as well.
- [030514] JST Template. Fixed jst magic hook and added Imakefile one. Changed jst magic hook variable setting setup to use the new $result variable.
- [030514] HTML Template. Changed **html-fold -> collapse**, reported goto-matching-tag failure.
- [030514] Buffer initialization. Changed the fold-setup call so that it gets called if likely to be required - use the new buffer-fhook **$result**(5) setting to help in determining this.
- [030514] Folding. Changed the fold-setup so a setup string can be passed in rather than searching for one in the buffer, aimed at utilizing the new buffer-fhook **$result**(5) setting.
- [030514] Set **$result**(5) to the magic fhook string so the fhook macro does not have to search for it again in buffer.c.
- [030514] Fixed "**xb**" bad regex bug.
- [030513] Search. Added region line hilighting support.
- [030513] Changed auto-spell hilighting scheme to a new spell-error scheme to avoid printing issues.
- [030513] Removed old `hkfold.emf` fold implementation. Moved `fold.emf` to `collapse.emf` and added a real folding editor support macros to a new `fold.emf`, based loosely on "*fue*".
- [030513] Abbreviation, Changed copy-region call to not expand the narrows.
- [030513] Latex Template. Added latex chars etc to the `.eaf` file.
- [030513] Search. Changed to handle the fold-open and fold-close split.
- [030513] OSD. Improved the Edit narrow commands to take advantage of the new protected narrow support.
- [030513] Tools. Fixed dummy tool removal and changed position setting to use alpha mark.
- [030513] VM Mail. Regex protected the boundary id string.
- [030513] Utils. simplified the str-to-regex by using &xrep.
- [030513] User setup. Changed **fold -> collapse.**
- [030513] HTML Template. Fixed script indent problem, improved the fence matching and added close tag feature.
- [030513] Buffer setup. Changed **fold -> collapse** and added proper folding setup entries.
- [030513] Commenting. Fixed silly "`\x80`" alpha char label bug.

- [030513] File Dialogue. Added tab ordering to dialog.
- [030513] Folding all templates. Changed the function folding commands to collapse to avoid confusion with real folding.
- [030513] XML Template, Added indent rules for "<? ... ?>".
- [030513] Mouse. Changed the column setting to use alpha marks.
- [030513] Color Schemes. Split the fold scheme into fold-close and fold-open.
- [030512] Fixed display matching fence issue with poke screen hiding the cursor on Unix X-Windows. May need to be done on MS-DOS as well.
- [030512] Fixed horrid crash caused in search.c by buffer change in macro followed by query replace and toolbar up-date changing the regex variables.
- [030512] Fixed fill-paragraph when no arg is given but at the beginning of para still move forward.
- [030512] Fixed compiler warning in spawn.c.
- [030512] Latest versions of build.txt, cygwin.txt, faq.txt and readme.txt.
- [030512] Build fix for SunOS.
- [030512] Ipipe fix for SunOS.
- [030512] Changedthe line flag implementation to be more flexible and variable in size so UE has a smaller memory usage and ME has line flags available.
- [030512] Improvedthe AMark setting and deleting.
- [030512] Added a Protected Line flag and vastly improved the narrowing. Renamed most function in `line.c` to make them more readable.
- [030512] Big return value change to get functions consistent - found that the return values of low level line functions where different to high level functions making the code hard to read and dangerous. Also needed a consistent triple value return (true false and error) so change `meABORT` to -1 across the board.

**6th Release - December 2002**

- [021216] Removed `.cvsignore` from source release.
- [021216] Fixed "**ne**" Unix build for a window build
- [021216] Fixed Perl abbreviations - Thanks to Dr. Detlef Groth.
- [021216] Fixed "$frame-width"/"depth" macro operation on the set - Thanks to Th.
- [021216] Confirmed Sun Solaris 2.6 PTY operation OK.
- [021208] Filed build file for cygwin, pushed cygwin binaries and re-released source.
- [021208] Added UNIX icons for CDE environment to meicons.tgz (me.t.pm, me.m.pm, me.l.pm)

- [021204] Fix for UNIX terminal resize with frames.
- [021204] Fix for Sun Solaris PTY creation and setup.
- [021204] Fix for VM mail searching - Thanks to Th.
- [021204] Fix for XML macros - Thanks to Th.
- [021203] Fix for frame resize crash.
- [021013] Fix for printing of *printer* buffer crash.
- [021013] Added PL/SQL file support.
- [021013] Updated mode change to use just a buffer type change flag.
- [021013] Added **@mna**(4) to all find-files incase there are problems.
- [021013] Broke the tool-bar tools out of the abbrev eaf file into there own tool eaf file.
- [021013] Added item-list and tag generation support to VB, fixed a few hilight issues.
- [021013] Shrunk the combo button size now that osd works better.
- [021013] Added "**!force**"s to all "unbind"'s to avoid bells.
- [021013] Made manual sub-menus have the .. And added Occurrences to the right button mouse menu.
- [021013] Updated macros for osd sub-menu position changes.
- [021013] Added new find and refresh tool toolbar options.
- [021013] Added Hot key for Microsoft friendly keys option.
- [021013] Added Finnish support - thanks to Sebastian Bojers.
- [021012] Fixed new frame ml junk text - new problem with line changes.
- [021006] Saved 3 bytes per line by using the rounded bytes and changing the length to an unused size. Fixed some minor Unix compile warnings.
- [020919] Changed the way manual sub-menus are opened and closed to improve usability (particularly for the combo box). Added auto side placement of sub-menus remove the 'e' 'w' and 'n' flags.
- [020912] Added crypting of registry files and Microsoft friendly key bindings.
- [020912] Fixed several html -> vb indent and hilight issues, re-implemented the mode change to be toolbar aware.
- [020912] Changed the "msshift" key list to a macro list which checks the non-shifted key is bound first. Added "`C-home`" and "`C-end`".
- [020912] Added **$window-id**(5).
- [020912] Added crypting support to registry files - crypt key is auto-generated.

- [020912] Fixed an odd win98 shell-command start-up dir bug and handled bogus consol sizes being returned (running "me -n" in me).
- [020912] Fixed a reload fhook bug and added code to handle the fhook deleting the buffer!
- [020912] Fixed windows -p lock-up bug caused by reading from stdin fp when no file is piped.
- [020830] VM Added code to handle empty multi message end section.
- [020723] New revamped Html support with toolbar support.
- [020723] Added right and bottom toolbars, and a paged tool concept, fixed quite a few problems found with the toolbar. Added a new Buffer Tools tool which loads the `hk???.eaf` file so that the abrev `???.eaf` file can be left as solely an abbrev file.
- [020723] Changed the interface to regexStrcmp to give better control. Added support for the "`@s#`" variables after a **&xsequal**(4), re-implemented the buffer context initialization to not use iscanner so it no longer need to swap buffers also changed the location of the fhook exec to earlier, where it was the window count was incorrect for the old current buffer and this could lead to a crash if the fhook deletes this buffer (found in CVS mode).
- [020723] Fixed a couple of goto-position problems.
- [020723] Fixed item-list search strings.
- [020723] Directory list - check for and fail on urls!
- [020723] Added new "`\\L`" -> x-line, "`\\X`" -> x-command special keys to improve eaf file read and portability.
- [020711] Added a system clipboard interaction disable flag to **$system**(5) so the tool-bar can stop messing up the clipboard whenever it updates.
- [020711] Fixed an uninitialized variable bug breaking the FTP logoff operation of **file-op**(2).
- [020629] HTML Added table creation and external preview commands.
- [020629] Fixed system clipboard being trashed problem by disabling interaction with it during tool-bar updates.
- [020629] Changed the List Functions to List Items in buffer setup.
- [020629] Added new kill list tool to the tool-bar.
- [020629] Added drag and drop support to directory view. Can now use this to load and copy/move files.
- [020629] Changed the FOLDER tags to FOLD and improved the eaf files.
- [020629] Added support for item-lists to C++ and added support for classes to C as the header is commonly used.
- [020629] Fixed nasty pick -> drop local buffer override problem.

- [020605] Added support for the toolbar and the initial frame size being set in the history.
- [020605] Pascal. Added "(* .. *)" and "//" comment hilighting.
- [020605] Macros - broke the setup default and standard tools into separate files. Improved the update flags for better control.
- [020605] New osd-find-buffer popup menu using the user-setup file types, based on Dr. Detlef Groth's.
- [020605] Fixed minor bug in file handling found by BC compiler.
- [020605] Fixed isearch so the next command will be executed in the correct frame.
- [020605] Updated makefile to new style which supports NE.
- [020605] Fixed NT and win2000 mouse cursor not staying hidden, caused by dubious mouse move events.
- [020523] CVS - added 'U'pdate file flag support.
- [020523] Modified to support **$window-flags**(5) and window resize changes.
- [020523] Added new "<-" and "->" truncate chars to **$window-chars**(5).
- [020523] Fixed mutli-frame console crash, window depth resize jump bug. Fixed drag-n-drop so it won't use a locked window. Fixed a nasty no-update bug involving the ERASEBKGND message. Added the drawing of a "<-" char to "0x08" (Backspace).
- [020523] Fixed a nasty multi-line regex search failing at end of buffer problem.
- [020523] Removed the ANSI and APRAM macros - redundant.
- [020523] Combined the shrink and enlarge window commands into a single **change-window-width**(2) and **change-window-depth**(2) command.
- [020523] Changed interface to frame resizing functions, numerical arg is now a delta change.
- [020523] Added left and right truncate chars to **$window-chars**(5), fix ABORT string value.
- [020523] Added extra parameter to HideBuffer flagging whether to hide the buffer if the window buffer lock is set. Changed **find-buffer**(2) numerical arg so -1 hides all but lock windows, -2 or less hides all.
- [020430] Improved the 'closest window match' algorithm so it gets the closest window!
- [020430] Windows - fixed an new compile problem and a frame loop bug in the font setup (frameCur == NULL for first call).
- [020428] Name changes (screen -> frame) due to frame support, new toolbar support, new **abbrev-list**(3) and **expand-look-back**(3), new filter on normal browser, auto-spell ignore, perldb sub-func fix, tables added to jst, fixes to bullet list in **ifill-paragraph**(3).

- [020428] Changed the ipipe wrap line length to be the 'term-size', using the frame size causes problems, particularly with the toolbar.
- [020427] Major edit! Added frame support, standardized many of the structure and function names, introduced $window-flags (not the size lock yet). Also fixed abbrev word boundary bug.
- [020427] US Letter support added by Th.
- [020301] CVS - added support for adding binary files.
- [020301] Added new JST support.
- [020301] Minor fix to the html processing and link following.
- [020301] Added latex template file.
- [020301] Mouse - changes required for changes to scroll commands which now abort at the top and bottom.
- [020301] Minor search fix to handle get-next-line requiring user input (out-of-date buffer etc).
- [020301] Major changes to create NanoEmacs! Made macro getting of keys for binding very strict, no superfluous chars allowed Changed mlwrite so that when aborting and no !force is used it will write message even with a MWEXCEL. Made printing optionally quiet for JST insert printing. Minor changes to scroll commands to abort when at the top or bottom. Small change to numerical interface to suspend-emacs.

**Experimental Release - January 2002**

- [020124] Split source code with conditional compilation into functional blocks.
- [020124] Added LATEX abbreviation file and updated template to load abbreviation file.
- [020124] JASSPA Structured Text (still prototyping - jst). Macros to convert structured text into other formats.
- [020124] Reworked the build files to build the multitude of debug, console and window variants.
- [020124] Added binary add mode to the CVS menu.

**5th Release - January 2002**

- [011230] Applied GPL License to code base. Transition to GPL following permission from Daniel M Lawrence.
- [011220] Fix to comment-restyle in comment.emf - stops a spin.
- [011220] nroff makefile full target building the txt files. (internal).
- [011220] Added support for a $CURSOR$ tag to the etf files which sets the initial cursor position, and included h.etf as an example (thats been annoying me for some time).

- [011220] Moved the template etfinsrt calls into the buffer-initialize macro.
- [011220] Fixed the exec fhook-cpp not working .
- [011220] Fixes to ctags for C++ function call initializers.
- [011220] Fixed Windows Console <RETURN> key keyboard mapping, returning incorrect binding.
- [011219] Fixed Termcap <TAB> key translation.
- [011219] Fixed Termcap tparm() to use a default stack argument if not specified by default.
- [011219] Added cygwin support to the build file.
- [011219] Added extra argument to buffer-initialise.
- [011218] Added support for Euphoria language.
- [011217] Italian spelling dictionary.
- [011217] Added scheme support for fence matching colors.
- [011217] Removed global binding of TAB key from unixterm.emf and build in translation.
- [011215] Fixed cygwin makefile for latest version, added cygwin macros.
- [011215] Fixed remaining termcap crash under cygwin.
- [011214] Merged prototype cygwin code base into release, use poll() following select().
- [011123] Fence matching added to user sertup (user-setup -> Platform -> Fence Display).
- [011115] Reworked the buffer hooks such that mybhook/myehook can be defined.
- [011031] Fixed a corking crash! (load random.c and do 2222 replace-string ”\n” ””).
- [011029] hklatex.emf item-list support (hit ”f7”).
- [011028] hklatex.emf has a greatly improved pseudo highlighting.
- [011028] Tweaked the ”@fs” variable to improve its usability.
- [011028] set-position can now store the mark position.
- [011028] Fixes to windows file times.
- [011016] New caps-lock, num-lock, scroll-lock and pause key support for win32 & unix. The ”*-lock” implementation to be more like ”S-pick” and ”C-pick” as the keys are automatically supported (like the shift) so there are no default binds to void, the keys are only generated if there are bound. Could not implement this on MS-DOS.
- [011016] which command built in i.e. ”M-x which”. Used for command utilities.

- [011026] A new Hide value feature when inputting strings, this is now used be the crypt and ftp Password entry. The string is shown as '*'s.
- [011025] Utilized the latest screen-update improvements to make the gdiff screen update rock solid.
- [011025] mlGetChar fix. The key "return" was the 78th special char, i.e. "SKEY_SPEC"—'N' and as the mlGetChar did not check the "SKEY_SPEC" bits, pressing return was like pressing 'N' so on a "Y/N" prompt try pressing return, its like saying No! .
- [011016] Spelling dictionaries renamed with ISO naming scheme. Because of the language changes, you need to rename your personal dictionary to lsdpengb.edf for British and lsdpenus.edf for American.
- [011016] Highlighting brackets/braces as requested from user base. The highlight braces required some other subtle changes to screen-update and show-region and the addition of the "rbin" mode means that this version is incompatible with the last in quite a few areas.
- [011016] Fixed mouse handler (turns out that if the pick event was lost (use "@cg" or an error) then me would lock!).
- [011016] Extended control on indent tab and return behavior (can disable the tab behavior).
- [011016] C++ name space indentation support.
- [011016] New language support interface so spelling is not required for a language to be supported (may only want the basic alphabet setup etc).
- [011016] Changed file loading interface so all commands (like view-file) can load in "bin", "crypt" or "rbin" mode.
- [010123] Reduced binary mode added (rbin).

**4th Release - August 2001**

- [010813] OSD bug fix for fixed size dialog boxes.
- [010812] Inclusion of occur regexp function.
- [010812] List function for summarizing file contents ("F7" / "Esc-F7" to close).
- [010810] Rendered character for the scroll bar - improves clarity.
- [010810] Modified the printer registry file location.
- [010801] New Guest default color scheme.

**3rd Release - Minor Updates July 14th 2001 onwards**

- [010718] LaTeX template includes incorrect bindings - fixed.
- [010716] Corrected Windows Install Shield "Company Entry" + "Registry entry fields".
- [010716] Minor macros modification to font selection for schemes. Default guest setup font selection for cleaner 'first' page.

- [010716] CVS macro update. Changed ”cvs-diff” to ”cvs-gdiff” and introduced a new ”cvs-diff” which just gives the ”diff” output and fixed several bugs when running with a CVS server. Also added support for the CVS/Template file when committing.
- [010716] Minor update to the CVS help page for the aforementioned (Affected all help files).

**3rd Release - July 2001**

- [010713] New style WWW JASSPA Pages.
- [010712] Added comment and menu support for MS-DOS and M4 (hkdos.emf/hkm4.emf).
- [010711] Added hpux11 support.
- [010711] Fixed windows erase when file doesn’t exist problems.
- [010710] Removed cbox - replaced by generic comment implementation (cbox.emf/me.ehf).
- [010710] Added a ’Plain Text’ char set which only uses plain letters (charset.emf).
- [010710] Fixed many little comment formatting bugs and annoyances (comment.emf).
- [010710] Changes required to build on MS-DOS using DJGPP v1.
- [010705] Bug fix for windows rendering crash.
- [010630] Minor reorder to X-Windows initialization for cygwin X-Server.
- [010507] Added generic commands for the creation, deletion and reformatting of comments now used by most of the file hooks.
- [010412] Added new generic buffer setup, menu and help macros, creating the new **buffer-setup**(3) command. Ported all existing file hooks to this new interface greatly reducing the buffer hook size while increasing its functionality and consistency.
- [010402] Added support for new **$buffer-backup**(5) variable for setting the back-up file location.
- [010312] Support for new key bindings allowing CU emulations. Nedit bindings provided as example. Major work to enable the complete re-binding of all keys so the user interface to MicroEmacs can be completely changed, this allows for example a Windows feel to be created. This required many internal ghosts to be exorcized and add new macro functions.
- [010308] User buffer configuration menu added.
- [010308] Improved Python support (Contribution PG).
- [010304] Bug fixes to hilighting logic.
- [010228] Improved dialog box for search/replace.
- [010210] Revised programming language templates; simplified the common elements used in configuring the templates.
- [010202] Added new **command-wait**(2) command to enable macros to wait for user dependent completion, used in ”gdiff”.

- [010202] Changed buffer variables to except the form ":<buffer-name>:<variable>", like command variables. This allows macros to access variable not defined in the current buffer.
- [010126] File "autosave" location changes. Control save name and location.
- [001005] OSD speedups.
- [001005] Improved and increased the file system operations via the new **file-op**(2) command, created better menus in the **file-browser**(3).
- [001010] Revamped the main ftp GUI to make it work much more rationally.
- [001010] Improved **ipipe-shell-command**(2) efficiency on Windows by introducing a new thread approach to listen for activity. Implemented a shell command-line in an ipiped environment so **ishell**(3) command is now usable on Windows platforms.
- [001010] Added new **set-position**(2) which can be used to store all information about the current window and **goto-position**(2) which will then restore them. Allowed macros to use non-letter characters for alpha marks and positions so they no longer need to clobber user ones.
- [001010] Added new "-k" and "-u" command-line options and improved "-s" option.
- [000922] Added GUI for tag dialogue for tags generation.
- [000922] Recursive tags support (i.e. nested directory tag traversal).
- [000810] emain.h "#define"'s properly supported again so MicroEmacs can be successfully compiled without options like SPELL and OSD etc. This can reduce the binary size by up to 37%.
- [000718] Source trees now maintained under CVS.
- [000719] Improved TCL/TK and Perl template support.
- [000704] Improved FTP and HTTP support.
- [000620] Java tags support.
- [000503] Revised printer support.
- [000503] Color printer support (Lexmark).

**2nd Release - Alpha patch - March 2000**

- [000322] FIRST PUBLIC RELEASE OF RENDERED FONT SUPPORT .
- [000322] Alpha build revision contains version information for 2000/03/22
- [000322] Internal reorganization of platform option selection.
- [000322] Minor fixes X-Windows color selection.
- [000322] Minor fixes in the OSD.
- [000322] Reviewed the documentation mode. A text description of the using the documentation modes is included in the distribution in the text directory.

- [000322] Minor modifications to support Borland 5.5 compiler (win32b55.mak).
- [000322] Support for Openstep 4.2 / NeXT Motorola 040 (Termcap only).
- [000322] Support for BSD 4.2/4.3.
- [000322] Revised UNIX Termcap support.
- [000322] New font rendering, supporting bold, italic, underline etc. All platforms modified to cope with font scheme definitions. Platforms optimized for speed.
- [000126] Beta build revision contains version information for 2000/01/26.
- [000126] Directory file copy. Within this release files in the directory listings may be selected for copy and pasting from the menu. Use the right mouse button to pop-up a menu in the directory buffers. Copy and paste to/from a ftp directory buffer to a local drive buffer may be used to initiate an ftp transfer. (This is how these pages are uploaded).
- [000126] New command ftp opens a ftp connection using a dialog. Use may save the host connection information in a registry file. Note that the password information is exposed so be careful where this is used - if in doubt do not enter the password and you will be prompted.
- [000126] New command **uniq**(3) operates as the UNIX of the same name. used with sort, and reduces to unique lines.
- [000126] Character translation; addition of character translation macros when moving between character sets. Accessible from Menu -> Format -> Change Buffer Char Set.
- [000126] Support for additional character sets, Windows GREEK, Windows TURKISH.
- [000126] Line highlighting selection with the use of the mouse and auto search. When using the mouse selection to set up the line highlighting when invoked from a `right-mouse -> Hilight Line -> color/remove` selection; the macro determines the mouse selected region and makes an appropriate decision as to how the line highlighting scheme is to be applied. The following rules apply to the mouse selection:
  - Point in line selected (i.e. mark == point) Highlight the line only.
  - Region in the line selected. Both point and mark are on the same line. Extract the selected text and hilight all lines that match the search pattern. The color ”remove” is also pattern matched to selectively remove lines.
  - Region selected mark & point on different lines Block up the region with appropriate highlight.
  - Note that line highlighting may be safely used in a folded buffer.
- [000126] Addition of special rendered space character. You may toggle this from the keyboard using a macro in your user.emf as follows:

```
; Macro to enable the display of TAB/CR. Note that we
; force the screen to be updated.
define-macro tab-display
set-variable $system &bxor $system 0x80000
screen-update
!emacro
global-bind-key "tab-display" "f11"
```

- [000126] Documentation update on new features (contained in me.ehf).
- [000114] Fixed font selection bug in 000112 beta.
- [000112] Revised color hilighting schemes, reduced the number of colors in a number of schemes. Introduction of the Sherwood Forest color scheme (green theme obviously !!).
- [000112] Small revision to the user interface with respect to font selection. In order to make character set selection easier then the multiple choices for the box characters have been removed from Menu -> Help -> User Setup. These have been replaced with a new check box Extended Char Set. This box should be checked to enable the special rendering characters and enable the box characters - this is typically enabled on X-Windows and MS-Windows.
- [000112] Provision for directory list sorting on name, modification time and size. Selecting the directory column title performs a re-sort based on the selection, a further selection reverses the order of the list.
- [000112] Addition of ftp support (meu build on Windows, enabled by default on UNIX). Enables files to be edited via ftp, (read and write supported). Transfer to another system using ftp://[<user>[:<password]@]><host>[:<port>]/<pathname>.
- [000112] Improved source code provision for foreign languages, various internal problems. Support for Russian (Windows and Cyrillic koi8-r).
- [000112] Y2K problems with time stamping and mode line date.
- [000112] New spelling rules for number sequences and specialized wild card sequences. Within the rule file (i.e. americar.emf, britishr.emf, etc.) then the following spelling rules may be added to cope with numeric sequences. The addition of these rules will spell check hexadecimal sequences and validate numeric sequences such as 1st, 2nd, 3rd etc. A spelling of "1nd" would generate a spelling error. This has not been performed for other languages so submissions for other language equivalents would be appreciated.

  ```
  ; Add some special word rules to make life easier.
  ; These are for 1st, 2nd 3rd etc.
  add-spell-rule "*" "\\([0-9]*[02-9]\\)?1\\(st\\|ST\\)"
  add-spell-rule "*" "\\([0-9]*[02-9]\\)?2\\(nd\\|ND\\)"
  add-spell-rule "*" "\\([0-9]*[02-9]\\)?3\\(rd\\|RD\\)"
  add-spell-rule "*" "[0-9]*1[0-9]\\(th\\|TH\\)"
  add-spell-rule "*" "[0-9]*[04-9]\\(th\\|TH\\)"
  add-spell-rule "*" "0[xX][[:xdigit:]]+"
  ; Spell rules for unit definitions.
  ; These are for 50Mb, 10Kg, -45.56cm 12ft 10.00am 16.00pm etc.
  add-spell-rule "*" "-?\\([0-9]*.\\)?[0-9]+\\(Kg\\|Kb\\|Gb\\|Tb ...
  \\|Mb\\|mm\\|cm\\|m\\|Km\\|ft\\|inch\\|mph\\|lb\\|am\\|pm\\)"
  ```

- [000112] Box selection copy and paste. Bindings are defined as `Esc C-w` **kill-rectangle**(2) `Esc C-y` **yank-rectangle**(2).

- [000112] Enhanced info file hilighting (Thanks to Dr. Detlef Groth for this).
- [000112] New copy-file command, typically used by the macro language for moving files around.
- [000112] Error reporting of bad command line arguments in Microsoft Windows environment (Thanks to Pedro Gomes for this).

**2nd Release - November 1999**

- [991101] Enhanced the latex support macros following various contributions from users.
- [991101] Added Favorites to the File pull down menu. This is a simple mechanism to allow the user to add a file to a favorites list.
- [991101] Ground up implementation of the regular expression pattern matcher, following licensing problems with the GNU regex. The resultant pattern matcher is now a little faster than GNU regex and is capable of all of the standard regular expression pattern matches. The pattern matcher has diverged from GNU in that double backslashes are required in the character classes [..] to allow for escape sequence short cuts such as '\n' (newline), '\t' (tab) etc. It was felt that this compromise was better than having to quote the more obscure search characters.
- [991101] Changed the hilighting and indenting syntax to be GNU regex compliant.
- [991101] Modified compare-windows to ignore white space by default, compare-windows-exact performs an exact character for character comparison.
- [991101] Enhanced the tags support to handle multiple tags and recursive directory tree searching.
- [991101] Implemented osd-help, a gui front end to the on-line help. Required changes to the help system which gives macros access to the on-line help buffer, needed for the index and search.
- [991101] Over-hauled the highlighting scheme files and editor to support the disabling of buffer highlighting.
- [991101] Many bugs fixed on all platforms, in particular unix cutting and pasting (crashes exceed) and focus problems and NT exit delay.
- [991101] Added tab and newline character printing in buffers.
- [991101] Introduced message line variables "@mx" and "@mxa".
- [991101] New Games - Crossword cracker. The ideal way to cheat at crosswords. Uses the spelling dictionary to guess at the word, given a number of known letters.
- [991101] Hook file protection against globally bound keys - prevents re-binding of user defined keys within the macro.
- [991101] Search/Replace fix for exact mode. Environment variable search path fix.
- [991101] hkm4.emf added back into the release (seemed to have gone missing).

- [991101] Cut and paste fix when running the UNIX version through an X-Server such as Exceed in the Windows environment.
- [991101] Plain text printer device added.
- [991101] Obscure OSD fix.
- [991101] New OSD help search facility Menu->Help->General Help.
- [991101] Fix for NT hang-up on shut down (typically encountered when running a source debugger).
- [991101] Fix for NT process spawning with spaces in the names.
- [991101] Metris macro fix, OSD bug fix, NT display fix.
- [991101] New - color scheme, GVim look.
- [991101] New - Unix/Linux support for Wheel Mouse (mouse.emf).
- [991101] New - Color schemes, Plain; for those that do not care for an abundance of colors.
- [991101] Modified the numeric key-pad behavior to be consistent on all platforms. Problems were noted with the UNIX number lock orientation which was reversed.
- [991101] New - Color scheme variables which allow the hkXXX files to determine the level of hilighting to apply to their templates.
- [991101] Line length fix; more graceful behavior (i.e. does not crash) when the line length exceeds 64K.
- [991101] Intercept the windows 105 keyboard application key (typically right hand side of keyboard next to the control key). The key is called ”menu”. By default this is bound in win32.emf, however users may bind it as a prefix for their own key combinations in much the same way as ”Alt” or ”C-x” if required.
- [991101] New; OMG’s IDL language template added.
- [991101] Fix for windows short name in next-buffer.

**2nd Release - Beta #1 - May 1999.**

- [990501] Bug fixes with multi-language support and spelling dictionaries.
- [990501] Fixed fill-paragraph such that it retains the cursor position in the paragraph when invoked without arguments.
- [990501] Moved to GNU regex for search/replace engine.
- [990501] Enhanced isearch such that it operates in *shell[990501 buffers (again).
- [990501] gdiff macro implementation of a graphical diff to allow color annotation of differences and difference selection. Uses the output of a standard **diff**(1) utility.
- [990501] 2nd port to IBM AIX.

**1999**

- [990101] Introduction of the address and date organizer (replacing the existing cal interface).
- [990101] Generic buffer folding using **narrow-buffer**(2).
- [990101] Rendered cursor support on all platforms. Smooth scrolling mode.
- [990101] Win32s port, for Microsoft 3.1/3.11 O/S.
- [990101] Re-worked ”translate-tcap-key” to generic ”translate-key” to solve many of the foreign language problems.
- [990101] Re-worked the ALT key mapping to allow conventional Emacs meta key bindings.
- [990101] Implemented Auto-Spell utility.
- [990101] Introduction of private macro variables of the form ”.name” and ”.macro.name”.
- [990101] Port to Sun Solaris Intel platform (2.6).
- [990101] Merged the init-hilight and hilight-token into the single highlight command. Similarly for indent.
- [990101] Enhanced the regexp support in hilight tokens, vastly improving it capability and usability. Similarly for indent tokens.
- [990101] Reorganized the highlighting files. Introduction of the scheme-editor and Highlight Search OSD’s.
- [990101] Implemented box character override support ($system bit 0x10000) on Win32 and Xterm platforms.
- [990101] Microsoft Windows native console support.

**Minor Patch - October 1998**

- [981001] Enhanced the operation of the Window pipes.
- [981001] Undo past the last save operation.
- [981001] Added Tabbed entries to OSD.
- [981001] Enhanced user setup using OSD.

**1st Release - September 1998**

- [980901] Support True-type fonts under windows, font selection dialog.

**1998**

- [981231] Registry features for configuration.
- [981231] Re-implementation of OSD.

- [981231] Re-implementation of termcap extended key support.
- [981231] Narrow support.
- [981231] Re-implementation of the session history.
- [981231] Re-implemented the speller based on ispell dictionaries.
- [981231] Re-worked the printer interface for Windows.
- [981231] Rationalized mouse key bindings.
- [981231] Added $system variable to configure MicroEmacs.
- [981231] Added random hilight-token addition and removal support.

**1997**

- [971231] Re-implementation of ipipes to enable terminal support.
- [971231] Initial Directory-Tree support.
- [971231] First implementation of the On Screen Display (OSD) menus and dialogues.
- [971231] Horizontal split window support.
- [971231] Scroll bar support.
- [971231] Added menu bar.
- [971231] Cursor position correction for hilights with invisible character.
- [971231] Indentation scheme support.
- [971231] Ported to Microsoft NT.
- [971231] Ipipes supported on NT.

**1996**

- [961231] Ported to Slackware Linux.
- [961231] First menu system (implemented in macros).
- [961231] Undo support.
- [961231] First ported to Microsoft Windows 95.
- [961231] Re-implementation of get-next-line.
- [961231] Auto mode support.
- [961231] Magic file hooks added.
- [961231] Proper key name support.
- [961231] Key bindings support numeric arguments.

**1995**

- [951231] Integral speller.
- [951231] Ported to HP-UX.
- [951231] Multiple ipipes supported on Unix.
- [951231] Poke-screen support.
- [951231] Call-back macro support.
- [951231] Metris created.
- [951231] Abbreviation and completion.
- [951231] First implementation of mailing and View Mail.
- [951231] Isearch expanded to support magic mode.
- [951231] Session history support.
- [951231] First ported to X-Terminal.
- [951231] Mouse support.
- [951231] Initial printer support.

**1994**

- [941231] DOS built with djgpp, allowed large files to be edited.
- [941231] Color hi-lighting.
- [941231] Re-implemented the macro language. Allowed separately named macros.
- [941231] Get-next-line support.
- [941231] File hooks added.
- [941231] Implemented Electric C.
- [941231] RCS support.
- [941231] Re-implemented backups and auto-saves.
- [941231] Re-implemented isearch.
- [941231] Re-implemented of keyboard macros.
- [941231] Binary file reading support.

**1992-1993**

- [921231] Ported to IBM AIX.
- [921231] Ported to Silicon Graphics

**1988-1999**

- [881231] Ported to MVME147, UNIX using curses.

- [881231] Fire fighting to get a stable version.
- [881231] Expanded regular expression syntax.

**Daniel M. Lawrence - Microemacs V3.8 distribution.**

- Daniel Lawrence turned MicroEmacs into an editor for the masses, it is his work that ultimately made MicroEmacs so popular. Daniel did a tremendous amount of work, both adding features and supporting other platforms resulting in the MicroEmacs v3.8 release on which JASSPA's MicroEmacs is based.

**Late 1985/early 1986**

- Dave Conroy, the original author, first developed the original code base for private use. Various people asked Dave for a copy of the program and it was distributed without concern for copyrights or ownership as was the custom of that era.

**Work In Progress or Planned**

Development of MicroEmacs is an on-going process, follows is a list of work items which is currently being undertaken or planned:

- Horizontal scroll bars.
- On-line tutorial to help beginners get up and running.
- Native printer support for the generation of postscript.
- Inclusion of **zlib**(1) to allow compressed file reading.

**Release History**

**September 2009**
: 10th Major Release - MicroEmacs 2009.

**August 2006**
: 9th Major Release - MicroEmacs '06.

**May 2005**
: 8th Major Release - MicroEmacs '05.

**January 2004**
: 7th Major Release - MicroEmacs '04.

**December 2002**
: 6th Major Release - MicroEmacs '02.

**January 2002**
: 5th Major Release - MicroEmacs '02.

**August 2001**
: Patch Release for MicroEmacs '01.

**July 2001**
: 3rd Major release - MicroEmacs '01.

**March 2000**
: 2nd Release alpha patch.

**May 1999**
2nd Major Release - MicroEmacs '99.

**October 1998**
Minor patch to MicroEmacs '98 to correct a font problem on Windows platforms.

**September 1998**
1st Major Release - MicroEmacs '98.

**1988** Daniel M. Lawrence - Microemacs V3.8 distribution.

Daniel Lawrence turned MicroEmacs into an editor for the masses, it is his work that ultimately made MicroEmacs so popular. Daniel did a tremendous amount of work, both adding features and supporting other platforms resulting in the MicroEmacs v3.8 release on which JASSPA's MicroEmacs is based.

**Late 1985/early 1986**
Dave Conroy, the original author, first developed the original code base for private use. Various people asked Dave for a copy of the program and it was distributed without concern for copyrights or ownership as was the custom of that era.

### Documentation

The documentation is all written in UNIX **nroff**(1) and converted to HTML, Microsoft Windows Help File format and MicroEmacs Help file format.

Ref: MicroEmacs - History File: me.0 Date: 2026/09/23

**FAQ**

This page contains frequently asked questions submitted to JASSPA.

FAQs - FAQ Contact information
faq00 - New functionality; what is useful to me as an old MicroEmacs user ??
faq01 - Languages; Are any foreign languages supported other than English ??
faq02 - C++ is not default, C is - how do I change this ??
faq03 - GNU Emacs; are there any GNU Emacs bindings. ?
faq04 - Icons are not displayed correctly in Microsoft Windows environments !!
faq05 - ipipes not working on Microsoft Windows network drives ?
faq06 - Language not supported - will it be ??
faq07 - Language file is incomplete
faq08 - Input locked up and not accepting keys; how do I unlock ?
faq09 - MicroEmacs Bindings; How do I get the original MicroEmacs bindings ?
faq10 - Microsoft Windows Locks up after killing an ipipe.
faq11 - Mouse support under Microsoft windows is strange !!
faq12 - Scroll bars too narrow !!
faq13 - Tab key; Why does the tab key not operate in some windows ??
faq14 - Termcap; On a color terminal why is there no color ??
faq15 - Termcap; Some of the keys do not work - how can I bind them ?
faq16 - Timestamp; Format incorrect, how can I change to MMDDYY.hhmm ?
faq17 - Windows; Component characters rendered incorrectly, how do I fix ?
faq18 - Windows Autosave and Backup files; are these potentially a problem ?
faq19 - Printing; Why in Windows does the output come out in a buffer ??
faq20 - Printing; On Windows which font should I use ??
faq21 - Printing; My printer is not supported ?
faq22 - Alt key maps to the Menu, how do I change ?
faq24 - Windows - Where is app850.fon ?
faq25 - Time; mode line is showing the date in DD/MM/YY format how do I change ?
faq26 - C or C++ indentation and effects; how can I turn off ?
faq27 - fill-paragraph function does not fill ??
faq28 - Key modifier which acts as the ESC key; what is it ?
faq29 - find-file start location; where is it ?
faq30 - Re-using a MicroEmacs session; how to ??
faq31 - Drag and Drop; is it supported ??
faq32 - Cut and Paste to/from other applications; is it supported ??
faq33 - Fonts; how can I change the font ??
faq34 - Colors; how can I change screen colors ??
faq35 - File Types; how do I interchange between UNIX, Windows and DOS files ??
faq36 - Non-English Languages; What font should I select ??
faq37 - MicroEmacs '99; How do I up-grade from MicroEmacs'98 ??
faq38 - Some keys on my foreign keyboard do not work properly, how do I get them working ??
faq39 - Tabs; How to change the tab width ??
faq40 - Windows/DOS; Where do I get grep/diff etc. ??
faq41 - Home/End Keys; How do I change the default bindings ??
faq42 - tags; How do I generate a MicroEmacs compatible tags file ??
faq43 - ne; how can I enable line wrapping in NE ??

Ref: Frequently Asked Questions File: me.0 Date: 2026/09/23

## NAME

me - MicroEmacs 2026 text editor

## SYNOPSIS

**me** [*options*] [*files ...*]

**me** [ **@** *startupFile* ] [ **-** ] [ **-a** ] [ **-A** ] [ **-b** *file* ] [ **-B** ] [ **-c** [ *sessionName* ]] [ **-f** ] [ **-h** ] [ **-i** ] [ **-k** [ *key* ] *file* ] [ **-l** *lineNo* [ **:** *column* ]] [ **-m** *command* ] [ **-n** ] [ **-o** ] [ **-0** *file* ] [ **-p** [ **-P** ] ] [ **-R** ] [ **-r** ] [ **-s** *string* ] [ **-u** *username* ] [ **-v** *variable*=*string* ] [ **-V** ] [ **-x** ] [ **-y** *file* ] *files...*

## DESCRIPTION

**MicroEmacs 2026** is a cut down version of the EMACS text editor, based on Danial Lawrences MicroEmacs. **MicroEmacs** is a tool for creating and changing documents, programs, and other text files. It is both relatively easy for the novice to use, but also very powerful in the hands of an expert. MicroEmacs can be extensively customized for the needs of the individual user.

**MicroEmacs** allows multiple files to be edited at the same time. The screen may be split into different windows and screens, and text may be moved freely from one window on any screen to the next. Depending on the type of file being edited, **MicroEmacs** can change how it behaves to make editing simple. Editing standard text files, program files and word processing documents are all possible at the same time.

There are extensive capabilities to make word processing and editing easier. These include commands for string searching and replacing, paragraph reformatting and deleting, automatic word wrapping, word move and deletes, easy case controlling, and automatic word counts.

For complex and repetitive editing tasks editing macros can be written. These macros allow the user a great degree of flexibility in determining how **MicroEmacs** behaves. Also, any and all the commands can be used by any key stroke by changing, or re-binding, what commands various keys invoke.

Special features are also available to perform a diverse set of operations such as file encryption, automatic backup file generation, en-tabbing and de-tabbing lines, executing operating system commands and filtering of text through other programs.

The command line options to **MicroEmacs** are defined as follows:

*@ startFile*

Initialize MicroEmacs using *startFile* [**.emf**]. The default when omitted is **me.emf**. See **start-up**(3) and Command Line Filters for more information.

**-**

Pipe *stdin* into buffer ***stdin***, when the buffer is saved the output is written *stdout*. Following is a simple example which changes 'a's to 'b's:

```
define-macro start-up
    find-buffer "*stdin*"
    beginning-of-buffer
    replace-string "a" "b"
    save-buffer
    quick-exit
!emacro
```

This can be used in the following manner:

```
me "@testpipe.emf" < foo.a > foo.b
```

See **-p** for disabling user interaction and the help page for command **ml-write**(2) for a further example.

**-a** Toggle the auto-save mode, this toggles the **auto**(2m) mode.

**Note:** that NanoEmacs disables auto save mode by default and MicroEmacs enables auto save mode by default. The behavior of **-a** is therefore different between the two editors.

**-A** When running in a UNIX console this option disabled the use of the alternative screen buffer. By default MicroEmacs will restore the console text when suspended or exited if the terminal supports the alternative screen buffer feature.

**-b** Load next file as a binary file (binary editor mode, uses **binary**(2m) buffer mode).

**-B** Toggle the backup mode, this toggles the **backup**(2m) mode.

**Note:** that NanoEmacs disables backup mode by default and MicroEmacs enables backup mode by default. The behavior of **-B** is therefore different between the two editors.

**-c**[*sessionName*]
Continuation mode. Load the **MicroEmacs** last edit session, restoring the buffers to their previous loaded state when history mode is enabled. The **-c** option is generally used with windowing interfaces (X-Windows/Microsoft Windows) as the shortcut icon invocation.

If the optional session name *sessionName* is specified then the session file *sessionName***.esf** is loaded restoring the previous editing session including the buffer window size, position and layout, the buffer mark and cursor location, main window size. When the *sessionName* includes space characters then the command line option should be bracketed e.g. `"-cFoo Bar"`. On a session restore then no additional files to load should be included on the command line as this will not enable the original window configuration to be restored as the window configuration will change. See **read-session**(3) for further information.

If the *sessionName* is omitted then the default session file *username***.esf** is loaded (see **esf**(8)) restoring the previously loaded buffers however the window layout and position is not restored. The default *sessionName* is the login name **$MENAME**(5).

**-h** Show the help page (does not start the editor).

**-f** Stop processing command-line arguments, instead set a .about.arg# variable for all following arguments.

**-i** MS-DOS versions of **MicroEmacs** only. Insert the contents of the current screen into the ***scratch**** buffer

**-k**[*key*] Load next file as an encrypted file (uses **crypt**(2m) buffer mode). The optional adjoining argument can be used to specify the decrypting key, if this argument is not specify the user will be prompted for it on start-up.

**-l***lineNo*[**:***column*]
Go to line *lineNo* in the next given file and optionally to the specified column number. The column may be omitted if not required. Typically used with utilities such a **more**(1) where an external editor may be invoked from other viewer.

**-m***command*

Sends a client-server command to an existing MicroEmacs session. The command takes the form "**C:**<*client*>**:**<*command*>" i.e. to write "`Hello World`" on the message line then a client may issue the command:

```
; launch server
me &
; send message
me -m "C:ME:ml-write \"Hello world\"
```

Note that the <*command*> is a MicroEmacs macro command, the escape sequences must be adhered to. The *client-server* interface is typically used to load a file, this may be performed as follows:

```
me -m "C:myutility:find-file \"/path/foo.bar\""
```

The absolute path is specified in this type of transaction as the current working directory of the active MicroEmacs session is unknown. The **-m** option de-iconize's the existing editor session and bring it to the foreground.

**-n** UNIX X-Windows environments only and MicroSoft Windows NT console versions. Execute **MicroEmacs** using termcap rather than X-Windows for UNIX; typically used within an **xterm** shell to fire up **MicroEmacs** for a quick edit. For Microsoft Windows, a console window is started as opposed to a GUI window.

**-o**<*file*>

Use already running version of MicroEmacs to load the <*file*>, if it exists, otherwise start a new editor session. This uses the *client-server* interface to push the new file into the existing editor session. Refer to the Client-Server Interface for details.

**-p** Used with the **-** which inputs from *stdin* and disables user interaction within MicroEmacs. To debug then the **-P** option may be used to enable debug on *stderr*. See the help pages for commands **ml-write**(2) and **command-wait**(2) for examples of this option's use.

**-P** Used with the **-p** option and is used for debugging the pipe mode, **ml-write**(2) messages are displayed on *stderr*.

**-R** Reverse the default video color scheme. The default foreground color becomes the background color and vice versa. This only affects the built in scheme and not any schemes defined in any start-up file.

**-r** Read-only, all buffers will be in view mode

**-s***string* Search for string "*string*" in the current buffer. e.g. `me -sfoo bar` starts **MicroEmacs**, loads file `bar` and initiates a search for *foo*. The cursor is left at the end of the string if located, otherwise at the top of the buffer.

**-u***username*

Set the current user name **$user-name**(5) to *username* before MicroEmacs is initialized.

**-v***variable*=*string*

Assign the MicroEmacs *variable* with *string*. The assignment is performed before the buffers are loaded. Typically used to change the start-up characteristics of the startup file(s).

**-V** Print the name and version of MicroEmacs. i.e.

```
orac% me -V
MicroEmacs 06 - Date 2006/08/10 - sunos
```

**-x** UNIX environments. Disable the capture of signals. **MicroEmacs** by default captures an handles all illicit signal interrupts. The option is enabled when debugging the source code allowing exception conditions to be trapped within the debugger.

**-y** Load next file as a reduced binary file (uses **rbin**(2m) buffer mode).

## ENVIRONMENT

The following environment variables are used by **MicroEmacs**.

**DISPLAY**

UNIX environments running X-Windows only. The identity of the X-Windows server. Typically set to **unix:0.0**, refer to the X-Windows documentation for details of this environment variable.

**MENAME**

**$MENAME**(5) is used to override the identity of the user **$user-name**(5). The variable may be used to determine which start-up configuration to use in the initialization of **MicroEmacs** (*$user-name*.erf). This is ignored if the **-u** option is specified on the command line.

Non-UNIX platforms usually need to explicitly set the **$MENAME** environment variable to identify the aforementioned files. for MS-DOS and early versions of Microsoft Windows this is typically performed in the `AUTOEXEC.BAT` file.

**PATH** The **$PATH** environment variable is used on most operating systems as a search path for executable files. This **$PATH** environment variable must be defined with **MicroEmacs** on the search path. Under UNIX this is set in the `.login`, `.cshrc` or `.profile` file i.e.

```
export PATH $PATH:/usr/name/me
```

Within MS-DOS or Microsoft Windows environments it is defined in the `AUTOEXEC.BAT` file. e.g.

```
set PATH=%PATH%;c:\me
```

**MicroEmacs** utilizes information in the **$PATH** environment variable to locate the start-up files, dictionaries etc.

**TERM** The terminal identification sting. In UNIX environments the environment variable **$TERM** is set to "`vt...`", in this case it is assumed that the machine is a server, and the host cannot support X (see command line option **-n**).

In MS-DOS the environment variable is usually set to define the graphics adapter mode. **%TERM** is assigned a string, understood by the `me.emf` start-up file, to set the graphics mode. Predefined strings include:

**E80x50** Initiates an 80 column by 50 line screen.

**E80x25** Initiates an 80 column by 25 line screen.

*userDefined*

A user defined string to set an explicit graphics card mode. The operation is dependent upon the support offered by the graphics adapter.

**MEINSTALLPATH**

**$MEINSTALLPATH**(5) is an environment variable that overrides any internally defined search path for MicroEmacs system macros and defines the root directory(s) of the MicroEmacs system macros. The presence of the environment variable affects **$search-path**(5).

**MEUSERPATH**

**$MEUSERPATH**(5) is an environment variable that defines the users private directory where user configuration files such as $user-name**.emf**, $user-name**.erf** and personal spelling dictionaries are stored. This is an absolute directory path. The environment variable sets the MicroEmacs variable **$user-path**(5).

**MEPATH**

**$MEPATH**(5) is an environment variable that may be used to define the absolute search path for MicroEmacs macro files and affects the setting of **$search-path**. The path should include the system macros location(s) and should include the user directory location **$user-path**(5) as the first path when the environment variable **$MEUSERPATH**(5) is not defined. If **$MEUSERPATH** is defined then this path name appears before the **$MEPATH** variable to define the user directory location. **$MEPATH** is not searching, so ALL directories to be searched must be included in the path specification.

**$MEUSERPATH**(5) is the preferred method used to defined the user directory.

**INFOPATH**

MicroEmacs uses the environment variable `$INFOPATH` as the directory(s) used to search for GNU **Info** files. Within the UNIX `$INFOPATH` is a semi-colon separated list of directories which are used to search for the MicroEmacs macro files. The path is searched from left to right. The environment variable is typically defined in the in the `.login`, `.cshrc` or `.profile` file i.e.

```
export INFOPATH /usr/local/info:$HOME/info
```

The default when omitted is `/usr/local/info`.

Within MS-DOS or Microsoft Windows environments it is defined in the `AUTOEXEC.BAT` file. e.g.

```
set MEPATH=c:\usr\local\info
```

There is no default location in these environments.

## FILES

All of the macro files and dictionaries are located in the **MicroEmacs** home directory. The standard file extensions that are utilized are:

**.eaf** **MicroEmacs** abbreviation file, defines completion definitions for buffer dependent text expansion.

**.edf** A **MicroEmacs** spelling dictionary. <*language*>**.edf** provide language specific dictionaries; *$LOGNAME*.**edf** is personal spelling dictionary.

**.ehf** **MicroEmacs** help file information. On-line help information for emacs, the main file is `me.ehf`.

**.emf** A **MicroEmacs** macro file. The following classes of macro file exist:

**me.emf** The default startup file.

<*platform*>*.emf*
A platform specify startup file, these include UNIX generic (`unixterm.emf`), UNIX specific (`irix.emf`, `hpux.emf`, `unixwr1.emf`, `linux.emf`, `sunos.emf` etc), Microsoft Windows (`windows.emf`), MS-DOS (`dos.emf`).

**hk***xxxxxx***.emf**
Buffer context specific hook files to initialize a buffer with macros and highlighting appropriate to the contents of the file type. e.g. 'C' language editing (`hkc.emf`), N/Troff typesetting (`hknroff.emf`), UNIX Manual page display (`hkman.emf`), Makefiles (`hkmake.emf`), etc.

**.erf** Registry files, used to retain personal information, users history in the file etc.

**.etf** Template files used to seed new files. Typically contains standard header information, copyright notices etc. that are placed at the head of files. The 'C' programming language is called `c.etf`

## MICROSOFT WINDOWS

Drag and drop support is provided in the Windowing version of MicroEmacs. The editor window is brought into focus following the drop.

## UNIX

Xdnd drag and drop support is provided in the X-Windows version of MicroEmacs. The editor window is not raised or brought into focus following the drop as expected in the UNIX environment.

## SESSIONS

Reading a session is generally performed when a MicroEmacs session is started using the "**-c**" command line option rather than from the mode line or main menu, see **read-session**(3). Invoking

**me -c**<*mySessionName*>

where <*mySessionName*> is the name of the session (the **$MENAME**(5) or system login name) will restore all of of the buffers, windows layout, point and mark positions, main window size etc. at the point of the last shut down. e.g. given a system login name of *jasspa* then

**me -c***jasspa*

restores the session to the state when *jasspa* last exited MicroEmacs. When the session name is specified then no other files should be specified, so the command line

**me -c***jasspa* `morefiles.txt`

restores the buffers but does not restore the window layout because this has been destroyed by the introduction of the new file "`morefiles.txt`". This type of command with another file behaves as the old "`me -c`" command prior to the 2006 release that did not support sessions. If <*mySessionName*> is not specified then the buffers are restored but the window layout is not restored.

## HISTORY

The 2006 release removed the `-d` command line option for debugging (not used) and added session support by extending the `-c` option with an optional *sessionName* argument.

## SEE ALSO

**read-session**(3), **$MEINSTALLPATH**(5), **$MENAME**(5), **$MEPATH**(5), **$MEUSERPATH**(5), **emf**(8), **erf**(8), **esf**(8), **$home**(5), **emacs**(1) [GNU], **more**(1), **ne**(1), **vi**(1).
Client-Server Interface.
Command Line Filters.

## NAME

ne - NanoEmacs 2026 text editor

## SYNOPSIS

**ne** [*options*] [*files ...*]

**ne** [ **@***startupFile* ] [ **-a** ] [ **-b** *file* ] [ **-B** ] [ **-c** ] [ **-h** ] [ **-k**[*key*] *file* ] [ **-l***lineNo*[**:***column*]] [ **-R** ] [ **-r** ] [ **-s***string* ] [ **-u***username* ] [ **-v***variable*=*string* ] [ **-V** ] [ **-x** ] [ **-y** *file* ] *files...*

## DESCRIPTION

**NanoEmacs 2026** is a cut down version of **MicroEmacs 2026** EMACS text editor, based on Danial Lawrences MicroEmacs.

**NanoEmacs 2026** provides the basic EMACS editor functions, allowing multiple files to be edited at the same time. The screen may be split into different windows and screens, and text may be moved freely from one window on any screen to the next. The editor typically runs in a console mode and is set up for basic system administration work or system bring up, macro files are not expected to be present, the system is still capable of running a macro file (with limited capability), the default start-up file is `ne.emf` and is executed at start-up if present.

The command line options to **NanoEmacs 2026** are defined as follows:

*@ startFile*
: Initialize NanoEmacs 2026 using *startFile*[**.emf**]. The default when omitted is **ne.emf**. See **start-up**(3) and Command Line Filters for more information.

**-a**
: Toggle the auto-save mode, this toggles the **auto**(2m) mode.

  **Note:** that NanoEmacs disables auto save mode by default and MicroEmacs enables auto save mode by default. The behavior of **-a** is therefore different between the two editors.

**-b**
: Load next file as a binary file (binary editor mode, uses **binary**(2m) buffer mode).

**-B**
: Toggle the backup mode, this toggles the **backup**(2m) mode.

  **Note:** that NanoEmacs disables backup mode by default and MicroEmacs enables backup mode by default. The behavior of **-B** is therefore different between the two editors.

**-c**
: Continuation mode. Load **NanoEmacs 2026** last edit session, restoring the buffers to their previous loaded state and position. Note that history mode must be enabled. The **-c** option is generally used with windowing interfaces (X-Windows/Microsoft Windows) as the shortcut icon invocation.

**-h**
: Show the help page (does not start the editor).

**-k**[*key*]
: Load next file as an encrypted file (uses **crypt**(2m) buffer mode). The optional adjoining argument can be used to specify the decrypting key, if this argument is not specify the user will be prompted for it on start-up.

**-l***lineNo*
: Go to line *lineNo* in the next given file and optionally to the specified column number. The column may be omitted if not required. Typically used with utilities such a **more**(1) where an external editor may be invoked from other viewer.

**-n**
: UNIX X-Windows environments only and MicroSoft Windows NT console versions. Execute **MicroEmacs** using termcap rather than X-Windows for UNIX; typically used within an **xterm** shell to fire up **MicroEmacs** for a quick edit. For Microsoft Windows, a console window is started as opposed to a GUI window. This option is not present if a console version of the application has been built.

**-R**
: Reverse the default video color scheme. The default foreground color becomes the background color and vice versa. This only affects the built in scheme and not any schemes defined in any start-up file.

**-r**
: Read-only, all buffers will be in view mode

**-s***string*
: Search for string "*string*" in the current buffer. e.g. `ne -sfoo bar` starts **MicroEmacs**, loads file `bar` and initiates a search for *foo*. The cursor is left at the end of the string if located, otherwise at the top of the buffer.

**-u***username*
: Set the current user name **$user-name**(5) to *username* before NanoEmacs is initialized.

**-v***variable=string*
: Assign the MicroEmacs *variable* with *string*. The assignment is performed before the buffers are loaded. Typically used to change the start-up characteristics of the startup file(s).

**-V**
: Print the name and version of NanoEmacs. i.e.

```
orac% ne -V
NanoEmacs 05 - Date 2005/02/24 - sunos
```

**-x**
: UNIX environments. Disable the capture of signals. **MicroEmacs** by default captures an handles all illicit signal interrupts. The option is enabled when debugging the source code allowing exception conditions to be trapped within the debugger.

**-y**
: Load next file as a reduced binary file (uses **rbin**(2m) buffer mode).

## ENVIRONMENT

The following environment variables are used by **NanoEmacs 2026**.

**DISPLAY**
: UNIX environments running X-Windows only. The identity of the X-Windows server. Typically set to **unix:0.0**, refer to the X-Windows documentation for details of this environment variable.

**MENAME**
: **$MENAME**(5) is used to override the identity of the user **$user-name**(5). The variable may be used to determine which start-up configuration to use in the initialization of **MicroEmacs** (*$user-name*.erf). This is ignored if the **-u** option is specified on the command line.

Non-UNIX platforms usually need to explicitly set the **$MENAME** environment variable to identify the aforementioned files. for MS-DOS and early versions of Microsoft Windows this is typically performed in the `AUTOEXEC.BAT` file.

**PATH** The **$PATH** environment variable is used on most operating systems as a search path for executable files. This **$PATH** environment variable must be defined with **MicroEmacs** on the search path. Under UNIX this is set in the `.login`, `.cshrc` or `.profile` file i.e.

```
export PATH $PATH:/usr/name/ne
```

Within MS-DOS or Microsoft Windows environments it is defined in the `AUTOEXEC.BAT` file. e.g.

```
set PATH=%PATH%;c:\ne
```

**MicroEmacs** utilizes information in the **$PATH** environment variable to locate the start-up files, dictionaries etc.

**TERM** The terminal identification sting. In UNIX environments the environment variable **$TERM** is set to "`vt...`", in this case it is assumed that the machine is a server, and the host cannot support X (see command line option **-n**).

In MS-DOS the environment variable is usually set to define the graphics adapter mode. **%TERM** is assigned a string, understood by the `me.emf` start-up file, to set the graphics mode. Predefined strings include:

**E80x50** Initiates an 80 column by 50 line screen.

**E80x25** Initiates an 80 column by 25 line screen.

*userDefined*
A user defined string to set an explicit graphics card mode. The operation is dependent upon the support offered by the graphics adapter.

**MEINSTALLPATH**
**$MEINSTALLPATH**(5) is an environment variable that overrides any internally defined search path for NanoEmacs system macros and defines the root directory(s) of the NanoEmacs system macros. The presence of the environment variable affects **$search-path**(5).

**MEUSERPATH**
**$MEUSERPATH**(5) is an environment variable that defines the users private directory where user configuration files such as $user-name**.emf**, $user-name**.erf** and personal spelling dictionaries are stored. This is an absolute directory path. The environment variable sets the NanoEmacs variable **$user-path**(5).

**MEPATH**
**$MEPATH**(5) is an environment variable that may be used to define the absolute search path for NanoEmacs macro files and affects the setting of **$search-path**. The path should include the system macros location(s) and should include the user directory location **$user-path**(5) as the first path when the environment variable **$MEUSERPATH**(5) is not defined. If **$MEUSERPATH** is defined then this path name appears before the **$MEPATH** variable to define the user directory location. **$MEPATH** is not searching, so ALL directories to be searched must be included in the path specification.

**$MEUSERPATH**(5) is the preferred method used to defined the user directory.

## FILES

All of the macro files and dictionaries are located in the **NanoEmacs** home directory. The standard file extensions that are utilized are:

**.emf** A **MicroEmacs** macro file. The following classes of macro file exist:

**ne.emf** The default startup file.

## SEE ALSO

**$MEINSTALLPATH**(5), **$MENAME**(5), **$MEPATH**(5), **$MEUSERPATH**(5), **emf**(8), **emacs**(1) [GNU], **more**(1), **me**(1), **vi**(1).
Command Line Filters.

Ref: ne(1) File: m1cmd002.1 Date: 2026/03/05

## NAME

abort-command - Abort command

## SYNOPSIS

**abort-command** (**C-g**)

## DESCRIPTION

Aborts the current command, when in trouble, this command will usually limit the damage. If you find yourself in a position where you do not want to be then this command will usually take you back to a sane state. This command rings the bell and stops keyboard macros.

Avoid re-binding this key where possible as it is used in other places.

When **abort-command** is invoked a warning is automatically given alerting the user, this may be an audible or a visual warning depending on the global state of the **quiet**(5) global variable.

## SEE ALSO

**buffer-mode**(2), **quiet**(5).

Ref: abort-command(2) File: m2cmd000.2 Date: 2022/06/12

## NAME

about - Information About MicroEmacs 2026

## SYNOPSIS

**about**

## DESCRIPTION

**about** displays information about the current MicroEmacs 2026 editing session and includes the following information:

- Version number and date (YYYY/MM/DD) information for MicroEmacs.
- Global status information including the number of active buffers and global mode status information.
- Current buffer status information; buffer modes and buffer size information.
- Full list of available optional features, see **&opt**(4) for more details.

## EXAMPLE

The following is an example output from **about**.

```
MicroEmacs 25 - Date 2025-09-01 - linux

Global Status:
  # buffers : 15
  Modes on  : autosv backup lf magic tab undo
  Modes off : binary cr crypt ctrlz edit exact fence hide indent justify lock
              nact narrow over pipe rbin time view wrap xlate

Current Buffer Status:
  Buffer    : *grep*
  File name : /home/jon/jasspa/microemacs/src/
  Lines     : Total    100, Current     26
  Characters: Total   7255, Current   1996
  Modes on  : backup lf magic tab view
  Modes off : autosv binary cr crypt ctrlz edit exact fence hide indent
              justify lock nact narrow over pipe rbin time undo wrap xlate

Options:
  abb cal cli col cry deb dir ext fen fho fra has hil hsp ipi ise lbi mag mou
  mwf nar nex osd pri rcs reg scr soc spa spe ssl tag tim typ und wor

Copyright (C) 1988-2025 JASSPA (www.jasspa.com)
```

## SEE ALSO

**describe-bindings**(2), **list-buffers**(2), **&opt**(4).

Ref: about(2) File: m2cmd148.2 Date: 2025/10/03

## NAME

add-color - Create a new color
add-color-scheme - Create a new color scheme

## SYNOPSIS

**add-color** "*col-no*" "*red*" "*green*" "*blue*"
*n* **add-color-scheme** "*schemeNum*" "*fore*" "*back*" "*current-fore*" "*current-back*"
"*selected-fore*" "*selected-back*"
"*current-selected-fore*" "*current-selected-back*"
["*fm-fore*" "*fm-back*" "*fm-cur-fore*" "*fm-cur-back*"
"*fm-sel-fore*" "*fm-sel-back*"
"*fm-cur-sel-fore*" "*fm-cur-sel-back*"]

## DESCRIPTION

**add-color** creates a new color and inserts it into MicroEmacs colors table, where *red*, *green* and *blue* are the color components and *col-no* is the MicroEmacs color table number. The color table contains 256 entries indexed by *col-no* in the range 0-255.

On some platforms (DOS and UNIX termcap) the number of colors is physically limited by the hardware to less than 256 (typically 16), in this case all 256 colors can be defined and for each created color the closest system color is used.

By default, only color 0 (white) and 1 (black) are defined. Once created, the colors may be used to create color schemes, this is the sole use of colors.

**add-color** may be used to modify an existing *col-no* index by re-assignment, the existing color definition is over-written with the new color definition. **add-color-scheme** creates a color scheme entry used by **hilight**(2), **screen-poke**(2), **osd**(2) and variables such as **$global-scheme**(5), **$buffer-scheme**(5), **$ml-scheme**(5).

The command takes an index number "*schemeNum*" and eight color values (defined by **add-color**) alternating between foreground and background colors. The 8 colors represent the 4 color paired states of foreground and background that may appear in a text buffer. The paired states correspond to current and selected lines (or permutations thereof). If an argument *n* is given to the command then *schemeNum* is set to a duplicate of the *n*th scheme, no other arguments are required.

*schemeNum* is the identifying index that is used to recognize the scheme. By default only two color schemes are defined at initialization, they are a monochrome scheme and inverse scheme with indices 0 and 1 using white as foreground and black as background, selected text is inverted. When defining a color scheme, if an existing *schemeNum* index is used then that scheme is modified.

The next eight arguments must be given, they specify foreground and background color pairs for the four different situations, as follows:

Default Color combination used when none of the following three are applicable.

Current Color combination used when the text is on the same line as the cursor. It is also used by the **$mode-line-scheme**(5) for the current window's mode line and for the current selection on an **osd**(2) dialog.

Selected

Color combination used when the text is in the current selected region, but is not on the current line. Also used by **osd** for non-current item Hot keys.

Current-selected

Color combination used when the text is on the current line and in the current selected region. Also used by **osd** for the current item's Hot key.

The following 8 arguments set up fonts and are optional, any missing arguments are defaulted to 0. Each argument is a bitmask indicating which font should be enabled, where each bit is as follows:

`0x01` Enable bold font.
`0x02` Enable italic font.
`0x04` Enable light font.
`0x08` Enable reverse font.
`0x10` Enable underlining.

Normally only the foreground value is used, i.e. the first, third, fifth and seventh values. But **screen-poke**(2) can be used to draw reversed color scheme in which case the background values are used.

## EXAMPLE

The color palette is typically created at start-up via the configuration file **scheme*X*.emf**. These files are not easily read as they are automatically generated via the **scheme-editor**(3) dialog. A more readable form of "`schemed.emf`" would be as follows:

```
; Standard colors
add-color &set .white     0 200 200 200
add-color &set .black     1 0   0   0
add-color &set .red       2 200 0   0
add-color &set .green     3 0   200 0
add-color &set .yellow    4 200 200 0
add-color &set .blue      5 0   0   200
add-color &set .magenta   6 200 0   200
add-color &set .cyan      7 0   200 200
; Light colors
add-color &set .lwhite    8 255 255 255
add-color &set .lblack    9 75  75  75
add-color &set .lred     10 255 0   0
add-color &set .lgreen   11 0   255 0
add-color &set .lyellow  12 255 255 0
add-color &set .lblue    13 0   0   255
add-color &set .lmagenta 14 255 0   255
add-color &set .lcyan    15 0   255 255
; Selection color
add-color &set .sel-col  16 91  78  131
; Set the required cursor-color
set-variable $cursor-color .col12
; Set up the standard schemes for the text, mode line message line, scroll bar and osd.
add-color-scheme $global-scheme  .white .black .lwhite .black ...
    ... .white .sel-col .lwhite .sel-col 0 8 1 9 8 0 9 1
add-color-scheme $ml-scheme  .white .black .lwhite .black ...
    ... .white .sel-col .lwhite .sel-col 0 8 1 9 8 0 9 1
add-color-scheme $mode-line-scheme  .white .red .lwhite .lred ...
    ... .white .red .lwhite .red 8 0 9 1 0 8 1 9
add-color-scheme $scroll-bar-scheme .white .lblack .lwhite .lblack ...
    ...  .lblack .white .lblack .lwhite 8 0 9 1 0 8 1 9
    .
    .
```

## NOTES

Color schemes can be created and altered using the **scheme-editor**(3) dialog, the created color scheme can then the used from start-up by using the **user-setup**(3) dialog. Therefore direct use of these commands is largely redundant.

The existence of a color or scheme index is checked as each entry is submitted, therefore any color or scheme used must have been previously been created, otherwise a default value is substituted.

Changing any existing color definitions causes all references to the color from a scheme to adopt the new color.

Changing any existing color-scheme definitions changes the rendered color of any **hilight**(2) etc., that was using that color-scheme.

A negative color scheme value (i.e. *-n*) uses the previous '*n*'th entry that is defined in the color block. i.e. if *current-fore* was specified as -2 then it would inherit the *fore* field color.

Not all UNIX terminals support all the above fonts.

On some telnet packages color is not directly supported and some of the termcap display attributes such as bold and italic are represented by a color (e.g. italic text is shown in green). Using this translation it is possible to achieve reasonable color support on a VT100 terminal - it is a little awkward but is worth while if you have to use this type of connection frequently.

## SEE ALSO

**scheme-editor**(3), **user-setup**(3), **change-font**(2), **hilight**(2), **screen-poke**(2), **$buffer-hilight**(5), **$cursor-color**(5), **$global-scheme**(5), **$trunc-scheme**(5), **$ml-scheme**(5), **$osd-scheme**(5), **$mode-line-scheme**(5), **$scroll-bar-scheme**(5), **$system**(5).

Ref: add-color(2) File: m2cmd001.2 Date: 2026/03/05

## NAME

add-dictionary - Declare existence of a spelling dictionary

## SYNOPSIS

*n* **add-dictionary** "*file*"

## DESCRIPTION

**add-dictionary** adds the given dictionary (specified by the given *file*) to the dictionary list. Note that the *file* may omit the **.edf** extension, this is automatically added.

The command accepts a numeric argument '*n*' which determines the actions to be undertaken. When *n* is omitted then the dictionary is marked for loading (on demand) - this is the standard invocation used in the start up files.

If an argument of **0** is given the dictionary is created but it is not marked for loading, this can be used to create an empty dictionary.

If an argument of **-1** is given the contents of the dictionary are dumped into the current buffer, used for dictionary maintenance. The two main uses of this command are discussed below.

### Dictionary Loading

A call to **add-dictionary** with no numeric argument does not perform an immediate load of the dictionary, instead the dictionary is only loaded on demand, i.e. when a call to **spell**(2) (usually via **spell-word**(3) or **spell-buffer**(3)) is made, this ensures that the start up time for MicroEmacs does not become too long. When the dictionary is loaded it is checked for efficiency, if found to be inefficient it is automatically optimized and flagged as changed. On exiting MicroEmacs, the user is prompted to save any dictionary that has be altered or optimized.

The spelling search order is made from the last dictionary added to the first, as soon as a word is found in a dictionary the search is halted. This implies that if a word has been defined incorrectly in one dictionary, but correct in another, the order in which the dictionaries are added determines the result.

The number of dictionaries allowed is unlimited but note that any words added are always added to the LAST dictionary. The size of the dictionary is restricted to about 16.7MB, the size is NOT tested when words are added and if this size is exceeded the results are undefined. However, it is unlikely that this limit will be reached, the largest dictionary created to date is 13.5MB.

A new main dictionary may be created as follows:

**1)** Find a file containing an **ispell**(1) compatible list of words.

**2)** **execute-file**(2) spellutl.emf to define macro **spell-add-word**(3).

**3)** Start up MicroEmacs and execute the command **add-dictionary** giving an appropriate new dictionary name.

**4)** Load up the file containing the words and execute the command **spell-add-word**(3) with a very large argument so all the words are added.

**5)** Save the dictionary by either executing the command **save-dictionary**(2) or exiting.

### Dictionary Dump

A call to **add-dictionary** with a numeric argument *n* of -1 causes the contents of the given dictionary to be dumped into the current buffer (make sure you are in an empty buffer or ***scratch***) where:

`xxxx` - Good word `xxxx` with no spell rules allowed
`xxxx/abc` - Good word `xxxx` with spell rules `abc` allowed
`xxxx>yyyy` - Erroneous word with an auto-replace to `yyyy`

The dump of the dictionary may be edited, allowing erroneous entries to be removed. The macro file `spellutl.emf` contains macros **edit-dictionary**(3) and **restore-dictionary**(3) which enable the user to edit a dictionary.

### NOTES

MicroEmacs is supplied with dictionaries for American and British English, it is strongly suggested that these dictionaries are **NOT** modified in anyway. Ensure that the dictionary is protected by loading the base dictionaries first, followed by a personal dictionary. New words added during spelling will then be added to the personal dictionary rather than the main dictionary.

### EXAMPLE

The MicroEmacs start-up file **me.emf** executes **language.emf** which in turn executes command **spell-rules-init** (located in **langutl.emf**) which adds the main language rules and dictionary. The command then adds the user's dictionary, this process can be simplified to:

```
; reset the spell rules
0 add-spell-rule
; Now add the American spell rules
-2 add-spell-rule "A" "" "" "re" "" ; As in enter > reenter
-2 add-spell-rule "I" "" "" "in" "" ; As in disposed > indisposed
    .
    .
; Now add the main American dictionary
add-dictionary "lsmenus"
; Now add the user dictionary
add-dictionary $MENAME
```

### SEE ALSO

**add-spell-rule**(2), **save-dictionary**(2), **spell-add-word**(3), **edit-dictionary**(3), **spell-buffer**(3).

## NAME

add-file-hook - DEPRICATED Declare file name context dependent configuration

## SYNOPSIS

**DEPRECIATED** as of January 2026, see **major-mode**(2).
*n* **add-file-hook** "*extensions*" "*fhook-name*" ["*macro1-name*" "*macro2-name*" ...]

## DESCRIPTION

**add-file-hook** defines a macro binding between a file name or file type and a set of macros. This binding enables file type dependent screen highlighting and key bindings to be performed. For a higher level introduction refer to File Hooks.

**add-file-hook** operates in two different modes to establish the type of file:

- Content recognition, by examination of the contents of the file.
- File extension recognition.

Content recognition has the highest priority and is used in preference to the file extension.

**add-file-hook** is called multiple times to add new recognition rules. The rules are interrogated in last-in-first-out (LIFO) order, hence the extension added last has a greater precedence than those added first. This ordering allows default rules to be over-ridden.

### Initialization

**add-file-hook** must be initialized prior to the first call, using an invocation of the form:

```
0 add-file-hook
```

with a numeric argument *n* of 0, and no arguments. This invocation resets the file hooks by deleting all of the installed hooks.

### File Extension Recognition

**add-file-hook** with no numerical argument *n* allows the extension of a file (or the base file name if there is no extension) to be used to determine which user defined setup macro is to be executed. The *extensions* argument is a space separated list of *file endings* (as opposed to true extensions) and is usually specified with the extension separator. For example, the extension "`.doc`" may indicate that the file is a document and therefore the indent, wrap and justify buffer modes are required. This may be performed automatically by defining a macro which adds these modes and adding a file hook to automatically execute this macro whenever a file "`*.doc`" is loaded.

The command arguments are defined as follows:

*extensions*
: A space separated list of file extensions, which are to be checked, this list includes the extension separator, typically dot ('`.`'). It should be noted that the extension comparison is always case-insensitive and with the tail of the file name string. As a result, files such as *makefile*, which do not have an extension as such, can still be matched using "`imakefile`".

*fhook-name*
: The name of the file hook to execute. This is the name of the macro to execute that initializes the buffer.

*macro1-name*
: Optional list of additional supporting macros located in the hook definition file, see notes section below.

As an example:

```
define-macro fhook-doc
    1 buffer-mode "indent"
    1 buffer-mode "wrap"
    1 buffer-mode "justify"
!emacro

add-file-hook ".doc" "fhook-doc"
```

It is quite possible that the same macro should be executed for a text file, i.e. "`*.txt`" this is achieved by a single **add-file-hook** as the space (' ') character is used as an extension separator, e.g.

```
add-file-hook ".doc .txt" "fhook-doc"
```

There are three special file hooks, which are **fhook-binary**, **fhook-rbin** and **fhook-default**, these are not predefined, but if the user defines them then they are executed whenever a file is loaded in binary or reduced binary mode (see **buffer-mode**(2)) or the extension does not match any of those defined.

Considering the `fhook-XXX` prefix, the initial '**f**' character must be present as this is changed to a '**b**' and an '**e**' when looking for the enter (begin) buffer and exit buffer hooks. These hooks are executed whenever the user swaps to or from a buffer (including creating and deleting). So for the given example, if the fill ignore characters are to be changed then this operation this is performed by defining the `bhook-XXX` and `ehook-XXX` macros, e.g.:

```
; Begin hook - entering the buffer - save the current fill-ignore characters
define-macro bhook-java
    !if &exi .fhook-java.fill-ignore
        set-variable .fhook-java.ofill-ignore $fill-ignore
        set-variable $fill-ignore .fhook-java.fill-ignore
    !endif
!emacro

; End hook - leaving the buffer - restore the fill-ignore characters
define-macro ehook-java
    !if &exi .fhook-java.fill-ignore
        set-variable $fill-ignore .fhook-java.ofill-ignore
    !endif
!emacro

; Fix up the $fill-mode so that we wrap java doc parameters. Do this by
; removing the "@" from the $fill-ignore variable.
!if &sin "@" $fill-ignore
    set-variable .fhook-java.fill-ignore $fill-ignore
    set-variable .fhook-java.fill-ignore &rep .fhook-java.fill-ignore "@" ""
!endif
```

File hooks are often used to setup the desired *buffer modes*, *hilighting*, *local key bindings*, *abbreviation file*, etc.

Buffer hooks are usually used to set and restore conflicting global variables.

### File Content Recognition

**add-file-hook** with a non-zero numerical argument *n* defines a macro binding between the content in a file and a set of macros. This binding enables file type dependent screen hi-lighting

and key binding to be performed. For a full description of file hooks refer to File Hooks, for file extension dependent hooking refer to **add-file-hook**(2).

The content defined file hooks interrogate the contents of a file on loading and search for a *magic* string identifier embedded in the text which uniquely identifies the file type.

The recognition process performs a search of the first *n* (numerical argument) non-blank lines of the file, searching for the regular expression specified by the *extensions* argument. The sign of the numerical argument *n* is interpreted as follows:

- **-ve** - Case insensitive search
- **+ve** - Case sensitive search

The command arguments are defined as follows:

*extensions*
: A regular expression string defining the text to be searched for.

*fhook-name*
: The name of the file hook to execute. This is the name of the macro to execute that will initialize the buffer.

*macro1-name*
: Optional list of additional supporting macros located in the hook definition file, see notes section below.

The search commences from the first non-blank line in the file, if the regular expression, defined by *extensions* is located then the file hook *fhook-name* is invoked. This is typically used to identify files which do not have file extensions i.e. UNIX shell script files. To identify a shell script file which commences with:

```
#!/bin/sh
```

The following file hook is used:

**1 add-file-hook** "**#!/.*sh**" "**fhook-shell**"

Note that "`.*sh`" also matches `/bin/csh`, `/usr/local/bin/zsh` etc, so care should be taken to ensure that the regular expression string is sufficiently well specified to recognize the file type.

The second class of embedded text are explicit identifiers embedded into the text. The embedded strings take the form:

*-*- mode -**
*-*-* **Mode:** *mode*; ... *-*-*
*-!- mode -!-*

The *-*-* notation belongs to GNU Emacs, but MicroEmacs recognizes the construct and extracts the string correctly. The *-!-* notation is MicroEmacs specific and is provided so as not to cause conflict with GNU Emacs. MicroEmacs searches for either construct on the first non-blank line of the file.

The explicit strings are defined with a negative numerical argument *n*, which identifies them as **explicit** rather than **magic** text strings. The *string* should be defined in lower case and matches a case insensitive string take from the file. e.g. to define a file hook for a make file:

```
#__________________________________-!-Makefile-!-____________________________________
#
# Make file for MicroEmacs using the Microsoft MSCV 2.0/4.0 development kit.
#
# Author      : Jon Green
# Created     :  020197.1002
# Last Edited : <150297.1942>
# File        : makefile.w32
....
```

might be defined as:

-1 **add-file-hook** "-!-[ \t]*makefile.*-!-" fhook-make

## NOTES

Optionally **add-file-hook** can be given a list of macros, defined within the hook file **hk***name***.emf**, the command will define the macros in the same way as command **define-macro-file**(2). Only macros useful before a file of the given type is created or loaded should be defined as loading the hook will define all macros within it.

The numeric argument given when the **$buffer-fhook** macro is executed is a bit based flag where the bits have the following meaning:

**0x01** When this bit is set the buffer is for a file that already exists, when clear the buffer is for a new file so the **fhook** should handle the insertion of any required file templates etc.

**0x02** If this bit is set a file content hook was used to identify the file type and registry variable #l9 will be set to the matched magic string. This can be useful as the magic string often contains configuration values such as the tab and indent width.

### Automatic Macro File Loading

**add-file-hook** performs an automatic load of a macro file if the **fhook** macro is not present in memory. The file name of the command file containing the macro is automatically derived from the *name* component of the **fhook** macro name. The **fhook-** part of the name is stripped off and prepended with **hk** and suffixed with **.emf**. Hence, macro **fhook-doc** would be searched for in file `hkdoc.emf` within the MicroEmacs directory. The command file is automatically loaded and executed.

In cases where the fhook macro is not located in an equivalent hook file, the file location of the macro may be explicitly defined for auto loading via a **define-macro-file**(2) invocation.

As an example, consider the C-mode file hook, used to load `.c` files. The loading of a C header file (`.h`) utilizes the same highlighting modes, but it's startup sequence is slightly different when handling new files. In this case the **fhook-cmode** for `.c` and **fhook-hmode** for `.h` files are located in the same hook file namely `hkcmode.emf`.

```
define-macro-file hkcmode fhook-hmode

add-file-hook ".c .cc .cpp .def .l .y .i .ac"   "fhook-cmode"
add-file-hook ".h .hpp"                         "fhook-hmode"
```

In this case the define-macro-file has been used to inform MicroEmacs of the location of the **fhook-hmode** macro thereby overriding the automatic load of a file called **hkhmode.emf**. The **fhook-cmode** macro requires no such definition as it is located in a hook file that matches the mode name, `hkcmode.emf`.

### Extending a standard hook definition

The standard file hook files **hk***XXX***.emf** should not be modified. The standard file hooks may be extended with local definitions by defining a file **my***XXX***.emf**, which is an extension to the hook file **hk***XXX***.emf**. This is automatically executed after **hk***XXX***.emf**. Refer to sections Language Templates and File Hooks for details.

### File Extensions

The file extensions are specified as a space separated list of file name endings. Back-up file endings such as tilde ( ) are not classed as correct file endings and are skipped by the file hook search, hence a file ending "`.c ` " invokes the same hook function as a "`.c`" file. It is therefore not necessary to add the backup and auto-save endings to the file hook definition.

The extension separator, usually dot (`.`), is typically added to the *extensions* list, they may be omitted with effect where a file always ends in the same set of characters. A notable example is "`makefile`" which includes no extension, as such, MicroEmacs applies the same hook function to a file called `Imakefile` as the endings are the same.

### Binary Files

It is sometimes useful to associate file types as binary files, so that they are immediately loaded in binary. In this case, both file extension and content recognition methods (i.e. of a magic string) are applicable. In both cases the file is bound to the well known hook `fhook-binary` which automatically loads the file in a binary mode.

Note, that for the content recognition process for a binary hook, the load time is doubled as the file is initially loaded in the default text mode, the binary hook function forces a second load operation in binary.

### SUMMARY

**add-file-hook** is summarized as follows:

- Binds one or more extensions to a macro called fhook-*xxxx*.
- Extensions are typically specified with the dot (`.`) separator.
- Multiple extensions are specified as a space separated list.
- Binds a regular expression search string to a macro called fhook-*xxxx*.
- The absolute value of the numerical argument determines the number of lines in the file over which the regular expression search is made.
- The sign of the numerical argument determines if the regular expression search is case (in)sensitive.
- When one of the files with a known file extension, or recognized content, is loaded macro **fhook-xxxx** is executed.
- **fhook-xxxx**, if undefined, is automatically searched for in file **hk***xxxx***.emf**.
- When the buffer containing the known file is entered (i.e. gains focus), then entry macro **bhook-xxxx** is executed.
- When the buffer containing the known file is exited (i.e. looses focus), then the exit macro **ehook-xxxx** is executed.

## EXAMPLE

The standard set of supported file types by MicroEmacs, at the time of writing, is defined as:

```
; reset the file hook list
0 add-file-hook
; Add file extension hooks.
; Files loaded in binary mode do not need hook as fixed
add-file-hook "*help* *info* .ehf"                              fhook-ehf
add-file-hook "*bindings* *commands* *variables*"               fhook-lists
add-file-hook "*buffers*"                                       fhook-blist
add-file-hook "/ *directory* *files*"                           fhook-dir
add-file-hook "*registry*"                                      fhook-reg
add-file-hook "*icommand* *shell* *gdb* *dbx*"                  fhook-ipipe
add-file-hook ".emf"                                            fhook-emf
add-file-hook ".doc .txt"                                       fhook-doc
add-file-hook ".1 .2 .3 .4 .5 .6 .7 .8 .9 .so .tni .sm"         fhook-nroff
add-file-hook ".c .h .def .l .y .i"                             fhook-c
add-file-hook ".cc .cpp .hpp .rc"                               fhook-cpp
add-file-hook "Makefile makefile .mak"                          fhook-make
add-file-hook "Imakefile imakefile"                             fhook-imake
add-file-hook ".sh .ksh .csh .login .cshrc .profile .tcshrc"    fhook-shell
add-file-hook ".bat .btm"                                       fhook-dos
add-file-hook ".man"                                            fhook-man
add-file-hook ".dmn"                                            fhook-dman
add-file-hook ".ini .hpj .reg .rgy"                             fhook-ini
add-file-hook ".htm .html"                                      fhook-html
add-file-hook ".htp .hts"                                       fhook-hts
add-file-hook ".tcl"                                            fhook-tcl
add-file-hook ".rul"                                            fhook-rul
add-file-hook ".awk .nawk .gawk"                                fhook-awk
add-file-hook ".p .pas"                                         fhook-pascal
add-file-hook ".vhdl .vhd"                                      fhook-vhdl
add-file-hook ".fvwm .fvwm2rc"                                  fhook-fvwm
add-file-hook ".java .jav"                                      fhook-java
add-file-hook ".nsr"                                            fhook-nsr
add-file-hook ".erf"                                            fhook-erf
; Add magic hooks
 1 add-file-hook "^#!/.*sh"              fhook-shell ; UNIX shell files
 1 add-file-hook "^#!/.*wish"            fhook-tcl
 1 add-file-hook "^#!/.*awk"             fhook-awk
 1 add-file-hook "^#VRML"                fhook-vrml
-4 add-file-hook "<html>"                 fhook-html
-1 add-file-hook "-[*!]-[ \t]*c.*-[*!]-"         fhook-c     ; -*- C -*-
-1 add-file-hook "-[*!]-[ \t]*c\\+\\+.*-[*!]-"   fhook-cpp   ; -*- C++ -*-
-1 add-file-hook "-[*!]-[ \t]nroff.*-[*!]-"      fhook-nroff ; -*- nroff -*-
-1 add-file-hook "-!-[ \t]*shell.*-!-"           fhook-shell ; -!- shell -!-
-1 add-file-hook "-!-[ \t]*msdos.*-!-"           fhook-dos   ; -!- msdos -!-
-1 add-file-hook "-!-[ \t]*makefile.*-!-"        fhook-make  ; -!- makefile -!-
-1 add-file-hook "-!-[ \t]*document.*-!-"        fhook-doc   ; -!- document -!-
-1 add-file-hook "-!-[ \t]*fvwm.*-!-"            fhook-fvwm  ; -!- fvwm -!-
-1 add-file-hook "-!-[ \t]*erf.*-!-"             fhook-erf   ; -!- erf -!-
-1 add-file-hook "-!-[ \t]*fold:.*-!-"           fhook-fold  ; -!- fold:... -!-
```

## OBSCURE INFORMATION

This section includes some low-level information which is so obscure it is not relevant to the typical user.

### Resolving Loading Order Problems

There is a potential loading order problem involving auto-loading of file libraries and the setting up of **bhook** and **ehook**. E.g. if the main fhook function has been defined as a **define-macro-file**(2), but the bhook or ehooks have not then when a buffer is created as only the fhook is define, only the fhook is set, the rest remain disabled even though the execution of the macro file will define these extra hooks.

To solve this problem simply define the bhook/ehooks as well. Note that automatically loaded hooks do not suffer from this problem as the macro file is executed before the hooks are assigned, thereby ensuring that all the hooks are defined.

## SEE ALSO

File Hooks, Language Templates, **major-mode**(2), **$buffer-bhook**(5), **$buffer-ehook**(5), **$buffer-fhook**(5), **define-macro-file**(2).

Ref: add-file-hook(2) File: m2cmd003.2 Date: 2026/03/05

## NAME

add-next-line - Define the searching behavior of command output

## SYNOPSIS

*n* **add-next-line** "*buffer-name*" [ "*string*" ]

## DESCRIPTION

**add-next-line** is used to set up the *next-line* functionality which is used by the **get-next-line**(2) command. The *next-line* feature is aimed at giving the user easy access to file locations which are stored in another buffer. This buffer may typically be the output from the **grep**(1) command or a compiler (e.g. **cc**(1)) and needs to contain the file name and line number of the required location.

As long as the format of the buffer is consistent and there is a maximum of one location per line, the *next-line* feature can be successfully configured.
The first argument, "*buffer-name*", gives the name the aforementioned buffer, this is "***grep***" for the **grep**(3) command etc. There is no limit on the number of next-line formats, nor on the number of **add-next-line** strings which are given. While there is no real need to initialize each new type, it is advised that the first **add-next-line** is called with a numerical argument of zero, e.g.:

```
0 add-next-line "*grep*"
add-next-line "*grep*" "....."
```

This tells MicroEmacs to reinitialize the type by freeing off any strings currently stored, note that the "*string*" argument is not used in this case. Resetting the next-line type safe guards against duplicate strings being added to it, a common problem if MicroEmacs is reinitialized.

Note that "*buffer-name*" should be considered a prefix, so if set to "`*grep*`" any buffer whose name starts `*grep*`, such as "`*grep*2`" will match.
Following is a typical output from grep:

```
foo.c: 45:      printf("hello world\n") ;
foo.c: 46:      printf("hello again\n") ;
```

If we replace the file name with "`%f`" and the line number with "`%l`", this becomes:

```
%f: %l:      printf("hello world\n") ;
```

get-next-line works on a left to right basis, as soon as it has enough information from the line it does not need to continue. Therefore the previous example can be reduced to just "`%f: %l:`". This is the string argument that should be given for the above example, i.e.:

```
add-next-line "*grep*" "%f: %l:"
```

get-next-line takes the given string and replaces the "`%f`" with **$file-template**(5) and the "`%l`" with the **$line-template**(5) and then uses the resultant string as a regular expression search string to find the next location. Crudely these could be set to "`foo.c`" and "`45`" respectively to find the first example, but this would fail to find any other. As a result the templates are usually magic search strings which will match any file and line number.
Similarly, following is an example output of the **gcc**(1) compiler:

```
basic.c:522: warning: 'jj' might be used uninitialized in this command
display.c:833: warning: implicit declaration of function 'ScreenPutChar'
```

In this case the **add-next-line** given needs to be:

```
add-next-line "*compile*" "%f:%l:"
```

If a negative numerical argument is given to **add-next-line** the given 'next-line' is ignored, this can be useful when some warnings are to be ignored. For example a common warning from gcc is given when a variable might be used uninitialized, given as follows:

```
bind.c:578: warning: 'ssc' might be used uninitialized in this function
```

These warnings can be ignored using the following:

```
-1 add-next-line "*compile*" ...
    ... "%f:%l: warning: '.*' might be used uninitialized in this function"
```

Some versions of **grep**(1) give the file name first and then the lines on the following lines. This is not a major problem as **get-next-line** remembers the last file name. The only problem occurs when skipping some parts of the list at which point the last file name parsed may not be the current file. Following is an example output of such a **grep** and the setup required:

```
File foo.c:
Line 45:      printf("hello world\n") ;
Line 46:      printf("hello again\n") ;
```

The configuration to locate the lines is defined as:

```
0 add-next-line "*grep*"
add-next-line "*grep*" "File %f:"
add-next-line "*grep*" "Line %l:"
```

## NOTES

The reinitialize command format of this command changed in January 2001, the format changed from:

```
add-next-line "*grep*" ""
```

## SEE ALSO

**$file-template**(5), **$line-template**(5), **cc**(1), **compile**(3), **get-next-line**(2), **grep**(1), **grep**(3).

Ref: add-next-line(2) File: m2cmd007.2 Date: 2022/06/12

## NAME

add-spell-rule - Add a new spelling rule to the dictionary

## SYNOPSIS

*n* **add-spell-rule** [ "*rule-letter*" "*base-ending*" "*remove*" "*derive-ending*" "*sub-rules*" ]

## DESCRIPTION

**add-spell-rule** adds a new spelling rule to the spelling checker. The rules effectively define the prefix and suffix character replacements of words, which is given a single character identifier, *rule-letter*, used in conjunction with the language dictionary. The letter conventions are defined by the **Free Software Foundation** GNU **ispell**(1) and more recently **myspell**(1) and **hunspell**(1) package.

**add-spell-rule** is used in the MicroEmacs language spelling initialization macro files named lsr<*language*>.emf, e.g. `lsrenus.erf`, `lsrelgr.erf` supplied in the MicroEmacs spelling or macros directory.

The command uses the numeric argument *n* to control the addition of a rule to the speller, as follows:

`0` **add-spell-rule**
: Removes all existing rules and re-initializes. This is, by convention, explicitly called before instantiating a new set of rules.

`-1` **add-spell-rule** "*rule-letter*" "*base-ending*" "*remove*" "*derive-ending*" "*sub-rules*"

`-2` **add-spell-rule** "*rule-letter*" "*base-ending*" "*remove*" "*derive-ending*" "*sub-rules*"
: Adds a prefix rule, an argument of -1 indicates that this prefix rule cannot be used with a suffix rule. An argument of -2 indicates it can be used in conjunction with any base word suffix rule which allows used with a prefix rule (e.g. argument of 2).

  "*rule-letter*" is any character in the range 0x21 to 0xff, all rules of the given letter must be a prefix rule and of the same type (i.e. same argument **n**). The start of a base word must match the given "*base-ending*" regular expression string for the rule to be applied; the "*remove*" must be a fixed string if not empty and must explicitly match the start of the *base-ending* and is removed from the start of the base word before the prefix "*deriv-ending*" is prefixed to the word. Example, for the American language;-

```
-2 add-spell-rule "R" "[^e]" "" "re" ""   ; As in label > relabel
-2 add-spell-rule "R" "e" "" "re-" ""     ; As in enter > re-enter
```

  The prefix rule of type 'R' can be applied to any base word which has rule 'R' enabled, and it prefixes "re" or "re-" to the word. The "*sub-rules*" argument can be used to list additional rules that can be applied to the base word if this rule has been applied, as MicroEmacs only supports a single prefix these rules must be suffixes and no other base word suffix can also be applied.

`1` **add-spell-rule** "*rule-letter*" "*base-ending*" "*remove*" "*deriv-ending*" "*sub-rules*"

2 **add-spell-rule** "*rule-letter*" "*base-ending*" "*remove*" "*deriv-ending*" "*sub-rules*"

Add suffix rules. An argument of 1 indicates that this suffix rule cannot be used with a prefix rule. An argument of 2 indicates it can be used in conjunction with any prefix rule which can also be used with a suffix rule (i.e. argument of -2).

"*rule-letter*" is any character in the range 0x21 to 0xff, all rules of the given letter must be a suffix rule and of the same type (i.e. same argument **n**). The end of a base word must match the given "*base-ending*" regular expression string for the rule to be applied, the "*remove*" string must be a fixed string if not empty and the must exactly match the end of the "*base-ending*". The "*remove*" string is removed first from the end of the base word before the "*deriv-ending*", which must also be a fixed string, is appended. Example, for the American language;-

```
2 add-spell-rule "N" "e" "e" "ion" ""     ; As in create > creation
2 add-spell-rule "N" "y" "y" "ication" "" ; As in multiply > multiplication
2 add-spell-rule "N" "[^ey]" "" "en" ""   ; As in fall > fallen
```

The suffix rule of type 'N' can then be applied to any base word which has rule 'N' enabled, and it can be used with prefixes, e.g. with rule R above to derive "*recreation*" from "*create*". A rule which cannot be used with prefixes, i.e.:

```
1 add-spell-rule "V" "e" "e" "ive" ""    ; As in create > creative
1 add-spell-rule "V" "[^e]" "" "ive" ""  ; As in prevent > collectpreventive
```

While some prefix words are legal, such as "*recreative*" but some are not, such as "*prevent*" where "*preventive*" is correct but "*repreventive*" is not. The "*sub-rules*" argument can be used to list additional rules that can be applied to the base word if this rule has been applied, MicroEmacs supports the addition up-to two suffixes and a single prefix. Sub-rules can be used to simply rules and reduces the number of rules required, for examle:

```
2 add-spell-rule "G" "" "" "ing" "S"  ; As in list > listing
2 add-spell-rule "S" "" "" "s" ""     ; As in pen > pens
```

Would automatically derive a "ings" suffix whenever a base word allows a "ing" suffix. This can also be used to reduce the number of erroneous derived words, for example:

```
-2 add-spell-rule "U" "" "" "un" ""     ; As in bend > unbend
2 add-spell-rule "A" "" "" "able" "US"  ; As in loan > loanable
2 add-spell-rule "S" "" "" "s" ""       ; As in pen > pens
```

These rules allow a base word of "*drink*" with flags "AS" to derive the correct words "*drinks*", "*drinkable*", "*drinkables*" "*undrinkable*" and "*undrinkables*" but not the erroneous words "*undrink*" or "*undrinks*".

## SPECIAL RULES

Following are special forms of add-spell-rule used for tuning the spell support, note that an argument can not be given:

**add-spell-rule** "-" "<*y*—*n*>"

Enables and disables the acceptance of hyphens joining correct words. By default the phrase "out-of-date" would be accepted in American even though the phrase does not exist in the American dictionary. This is because the three words making up the phrase are correct and by default hyphens joining words are allowed. Some Latin language such as Spanish do not use this concept so this feature can be disable.

**add-spell-rule** "*#*" "*score*"

Sets the maximum allowed error score when creating a spelling guess list. When comparing a dictionary word with the user supplied word, **spell** checks for differences, each difference or error is scored in the range of 20 to 27 points, once the maximum allowed score has been exceeded the word is ignored. The default guess error score is 60, allowing for 2 errors.

**add-spell-rule** "*" "*regex*"

Adds a correct word in the form of a regex if a word being spell checked is completely matched by the **regex** the word is deemed to be correct. For example, the following rule can be used to make the spell-checker allow all hex numbers:

```
add-spell-rule "*" "0[xX][[:xdigit:]]+"
```

This will completely match the words "`0x0`", "`0xff`" etc but not "`0x00z`" as the whole word is not matched, only the first 4 letters.

## NOTES

The **ispell-convert** macro in **spellutl.emf** can be used to convert **ispell**(1), **myspell**(1) and **hunspell**(1) compliant affix (.aff) and dictionary (.dic) files to MicroEmacs. The format of the dictionary is a list of base words with each word having a list of rules which can be applied to that word. Therefore the list of words and the rules used for them are linked e.g.

```
aback
abaft
abandon/DGRS
abandonment/S
abase/DGRS
abasement/S
abash/DGS
abashed/U
abate/DGRS
achieve/GS
achieve/RSfDG
```

where the "`/...`" is the valid list of rules for that word.

MicroEmacs uses a proprietary binary format for dictionaries which allows for rapid initialisation, this format only supports a single copy of a base word so a single word requires multiple entries to define different rule combinations, such as for *achieve* above, it separates the rule groups with the 0x01 character which is not a valid rule.

MicroEmacs does not support the full set of **hunspell** features, some are unlikely to be required, such as *COMPLEXPREFIXES* as MicroEmacs does not support right-to-left languages such as Arabic. Other features such as support for compounds may be added at some point in the future, see **spell.c** for the current list of unsupported futures.

## SEE ALSO

**add-dictionary**(2), **spell**(2), **spell-buffer**(3), **spell-word**(3), **ispell**(1).

## NAME

append-buffer - Write contents of buffer to end of named file

## SYNOPSIS

*n* **append-buffer** "*file-name*"

## DESCRIPTION

**append-buffer** is used to write the contents of the current buffer into an EXISTING file. Use **save-buffer**(2) if the buffer is to over-write the existing file already associated with the buffer. Use **write-buffer**(2) if the buffer is to be written out to a new file, or to replace an existing file.

**append-buffer** writes the contents of the current buffer to the named file *file-name*. But unlike **write-buffer**(2) the action of the write does not change the attributes associated with the file (if it exists), it also does not effect the stats of the current buffer.

On writing the file, append-buffer ignores the **time**(2m) and **backup**(2m) mode settings. The current buffer will not be time stamped and a backup will not be created for "*file-name*". If the buffer contains a **narrow**(2m) it will automatically be removed before saving so that the whole buffer is saved and restored when saving is complete

The argument *n* is a bit based flag, where:

**0x01** Enables validity checks (default). These include a check that the given file already exist, if not confirmation of writing is requested from the user. Without this flag the command will always succeed wherever possible. If "*file-name*" does not exist the buffer is written out in a similar fashion to using the command **write-buffer**(2).

**0x02** Disables the expansion of any narrows (see **narrow-buffer**(2)) before appending the buffer.

**0x04** Truncate the existing file before writing out the contents of the buffer. This means that the file will consist solely of the contents of the buffer, but it will still have the file attributes of the original file.

If *n* is not specified then the default argument of 1 is used.

## EXAMPLE

The following example appends the current buffer onto the end of a file, creating the file if it does not exists

```
append-buffer "things_to_do.txt"
```

The following example truncates the users email file while maintaining the file attributes. This is taken from **vm**(3) where it is used to remove the current mail from the system mail box.

```
find-buffer "*vm-empty-buffer"
-1 buffer-mode "ctrlz"
5 append-buffer %vm-mail-src
delete-buffer $buffer-bname
```

Note that the macro ensures that **ctrlz**(2m) mode is removed. If it was enabled then the file written would not be empty.

## SEE ALSO

**write-buffer**(2), **save-buffer**(2).

Ref: append-buffer(2) File: m2cmd123.2 Date: 2022/06/12

## NAME

backward-delete-tab - Delete white space to previous tab-stop

## SYNOPSIS

**backward-delete-tab** (**S-tab**)

## DESCRIPTION

**backward-delete-tab** deletes all white characters left of the cursor back to the previous tab stop or non-white space, the deleted text is not added to the kill buffer.

## SEE ALSO

**tab**(2), **$tab-width**(5), **$buffer-tab-width**(5).

Ref: backward-delete-tab(2) File: m2cmd037.2 Date: 2022/06/12

## NAME

beginning-of-buffer - Move to beginning of buffer/file end-of-buffer - Move to beginning/end of buffer/file

## SYNOPSIS

**beginning-of-buffer** (**esc** <)
**end-of-buffer** (**esc** >)

## DESCRIPTION

**beginning-of-buffer** places the cursor at the beginning of the buffer/file.

**end-of-buffer** places the cursor at the end of the buffer/file.

## NOTES

**beginning-of-buffer** is typically bound to **home**.
**end-of-buffer** is typically bound to **end**.

## SEE ALSO

**beginning-of-line**(2), **end-of-line**(2).

Ref: beginning-of-buffer(2) File: m2cmd012.2 Date: 2022/06/12

## NAME

beginning-of-line - Move to beginning of line
end-of-line - Move to end of line

## SYNOPSIS

**beginning-of-line** (**C-a**)
**end-of-line** (**C-e**)

## DESCRIPTION

**beginning-of-line** places the cursor at the beginning of the line.

**end-of-line** places the cursor at the end of the line.

## SEE ALSO

**beginning-of-buffer**(2), **end-of-buffer**(2).

Ref: beginning-of-line(2) File: m2cmd026.2 Date: 2022/06/12

## DEFAULT KEY BINDINGS

The default key bindings are presented below in four alphabetical lists, one for single key bindings and one for each of the 4 bound prefixes (`esc`, `C-x`, `C-h` & `C-c`). See Key Names for a list of valid key names.

### Single-Key Sequences

`backspace` backward-delete-char Delete the previous character.
`delete` forward-delete-char Delete character under the cursor.
`down` forward-line Move to next line.
`end` end-of-buffer Move to the end of the buffer.
`esc` prefix 1 Meta character prefix.
`f1` osd Open top main menu.
`home` beginning-of-buffer Move to the start of the buffer.
`insert` buffer-mode Toggle over-write mode.
`left` backward-char Move backward one character (left).
`page-down` scroll-down Move forward by one screen.
`page-up` scroll-up Move backward by one screen.
`return` newline Insert a new line.
`right` forward-char Move forward one character (right).
`tab` tab Insert a tab character.
`up` backward-line Move to previous line.

`S-backspace` backward-delete-char Delete the previous character.
`S-delete` forward-delete-char Delete character under the cursor.
`S-tab` backward-delete-tab Delete white space to previous tab-stop.

`C-a` beginning-of-line Move to beginning of line.
`C-b` backward-char Move backwards by one character
`C-c` prefix Control character prefix.
`C-d` forward-delete-char Delete character under the cursor.
`C-e` end-of-line Move to end of line.
`C-f` forward-char Move forward one character (right).
`C-g` abort-command Abort current command.
`C-h` prefix Control character prefix.
`C-i` insert-tab Insert tab character.
`C-k` kill-line Delete from cursor to the end of the line.
`C-l` recenter Redraw screen with current line in the center.
`C-m` newline Insert a new line.
`C-n` forward-line Move to next line (down).
`C-o` insert-newline Open up a blank line.
`C-p` backward-line Move to previous line (up).
`C-q` quote-char Insert literal character.
`C-r` isearch-backward Start incremental search backwards.
`C-s` isearch-forward Start incremental search forwards.
`C-t` transpose-chars Transpose two letters.
`C-u` universal-argument Repeat the next command *n* times (default is 4).
`C-v` scroll-down Move forward by one screen.
`C-w` kill-region Delete a marked region.
`C-x` prefix Control character prefix.

C-y yank Restore what was copied or deleted.
C-z scroll-up Move backward by one screen.
C-_ undo Undo the previous edit.
C-space set-mark Set start of a region.
C-down forward-line Move forward five lines.
C-left backward-word Move one word backward.
C-page-down scroll-next-window-down Scroll next window down a page.
C-page-up scroll-next-window-up Scroll the next window up a page.
C-right forward-word Move one word forward.
C-up backward-line Move backward 5 lines.
A-e file-browser Browse the file system.
A-r replace-all-string Replace string with new string in a list of files.
A-down scroll-down Scroll the current window down one line.
A-left scroll-left Scroll the current window left one character.
A-right scroll-right Scroll the current window right one character.
A-up scroll-up Scroll the current window up one line.

### esc Prefix Sequences

esc ! pipe-shell-command Pipe a shell command to a buffer.
esc % query-replace-string Search and replace a string - with query.
esc $ spell-word Spell a word.
esc . set-mark Set the start of a region.
esc / execute-file Execute script lines from a file.
esc < beginning-of-buffer Move to the start of the buffer.
esc > end-of-buffer Move to the end of the buffer.
esc ? help Help - high level introduction to MicroEmacs.
esc @ pipe-shell-command Pipe a shell command to a buffer.
esc [ backward-paragraph Goto the beginning of the paragraph.
esc \ ipipe-shell-command Incrementally pipe a shell command to a buffer.
esc ] forward-paragraph Move forward one paragraph
esc ^ delete-indentation Join 2 lines deleting white spaces.
esc b backward-word Move one word backwards
esc c capitalize-word Capitalize first letter of a word
esc d forward-kill-word Delete word the cursor is on.
esc e set-encryption-key Reset the encryption key.
esc f forward-word Move one word forward.
esc g goto-line Goto a line.
esc i tab Insert a tab character.
esc k global-bind-key Bind a key to a command or macro.
esc l lower-case-word Lowercase word.
esc m global-mode Toggle a global mode.
esc n forward-paragraph Move forward one paragraph
esc o fill-paragraph Reformat (fill) current paragraph.
esc p backward-paragraph Goto the beginning of the paragraph.
esc q ifill-paragraph Reformat (fill) current paragraph.
esc r replace-string Search and replace text (no query).
esc t find-tag Find a tag.
esc u upper-case-word Uppercase word.
esc v scroll-down Move to the previous screen.

esc w copy-region Copy region to the kill buffer.
esc x execute-named-command Execute the named command.
esc y reyank Kill current yank data and restore previous kill buffer data.
esc z quick-exit Save all buffers and exit.
esc buffer-mode Remove edited status from current buffer.
esc backspace backward-kill-word Delete the word under the cursor.
esc esc expand-abbrev Expand an abbreviation.
esc space set-mark Set the start of a region.
esc C-c count-words Count words in a region.
esc C-f goto-matching-fence Reposition the cursor at an opposing bracket.
esc C-g abort-command Abort current command.
esc C-i tab Insert tab character.
esc C-k global-unbind-key Unbind a key from a command or macro
esc C-n change-buffer-name Rename current buffer.
esc C-r query-replace-string Search and replace with query.
esc C-v scroll-next-window-down Scroll next window down a page.
esc C-w kill-paragraph Delete current paragraph.
esc C-z scroll-next-window-up Scroll the next window up a page.
esc A-r query-replace-all-string Query replace string in a list of files.

### C-x Prefix Sequences

C-x # filter-buffer Filter the buffer through a shell filter.
C-x ( start-kbd-macro Start recording a keyboard macro.
C-x ) end-kbd-macro Stop recording a keyboard macro.
C-x / isearch-forward Start incremental search forwards.
C-x 0 delete-window Delete the current window.
C-x 1 delete-other-windows Delete other windows.
C-x 2 split-window-vertically Split the current window into two.
C-x 3 split-window-horizontally Split current window horizontally into two.

C-x 5-1 create-frame Create a new internal frame.
C-x 5-2 create-frame Create a new external frame.
C-x 8 find-cfile Find and load an encrypted file for editing.
C-x 9 find-bfile Find and load a file for binary editing.
C-x < scroll-left Scroll the window left by one screen width.
C-x = buffer-info Show cursor position information
C-x > scroll-right Scroll the window right by one screen width.
C-x ? describe-key Describe binding of command to key.
C-x @ pipe-shell-command Pipe a shell command to buffer.
C-x [ scroll-up Move backward by one screen.
C-x ] scroll-down Move forward by one screen.
C-x ^ grow-window-vertically Enlarge the current window by a line.
C-x ` get-next-line Find the next command line.
C-x a goto-alpha-mark Move the cursor to an alphabetic mark.
C-x b find-buffer Switch window to a buffer.
C-x c shell Start a new command processor.
C-x e execute-kbd-macro Execute a macro.
C-x h hunt-forward Continue search in forward direction.

C-x k delete-buffer Delete buffer.
C-x m buffer-mode Toggle a local buffer mode.
C-x n change-file-name Rename current buffer file name.
C-x o next-window Move to the next window.
C-x p previous-window Move to the previous window.
C-x q kbd-macro-query Query keyboard macro.
C-x r search-backward Search in a reverse direction.
C-x s search-forward Search in a forward direction.
C-x u undo Undo the previous edit.
C-x v set-variable Assign a new value to a variable.
C-x w resize-window-vertically Resize the window.
C-x x next-buffer Switch to the next buffer.
C-x z grow-window-vertically Enlarge the current window.
C-x { shrink-window-horizontally Shrink current window horizontally.
C-x } grow-window-horizontally Enlarge current window horizontally.
C-x C-a set-alpha-mark Mark the current position with an alphabetic mark.
C-x C-b list-buffers Display buffer list.
C-x C-c save-buffer-exit-emacs Exit MicroEmacs.
C-x C-e execute-kbd-macro Execute a macro.
C-x C-f find-file Find a file and load into buffer.
C-x C-g abort-command Abort current command.
C-x C-h hunt-backward Resume search in backwards direction.
C-x C-i insert-file Insert file into the current buffer.
C-x C-l lower-case-region Lowercase region.
C-x C-n scroll-down Scroll the current window down one line.
C-x C-o delete-blank-lines Delete blank lines about the cursor.
C-x C-p scroll-up Scroll the current window up one line.
C-x C-q rcs-file Interact with RCS to check in/out a file.
C-x C-r read-file Read a file from disk.
C-x C-s save-buffer Save current file to disk.
C-x C-t transpose-lines Swap adjacent lines.
C-x C-u upper-case-region Uppercase region.
C-x C-v view-file Read a file for viewing (read only).
C-x C-w write-buffer Write a file to disk witn new name.
C-x C-x exchange-point-and-mark Exchange cursor with mark position.
C-x C-y insert-file-name Insert filename into current buffer.
C-x C-z shrink-window-vertically Reduce size of current window.

### C-h Prefix Sequences

C-h a command-apropos List commands involving a concept.
C-h b describe-bindings Show current command/key binding.
C-h c list-commands List available commands.
C-h d describe-variable Describe current setting of a variable.
C-h k describe-key Describe keyboard binding.
C-h v list-variables List defined variables.
C-h C-c help-command Display command help information.
C-h C-i help-item Display item help information.
C-h C-v help-variable Display variable help information.

## NAME

buffer-bind-key - Create local key binding for current buffer
buffer-unbind-key - Remove local key binding for current buffer

## SYNOPSIS

*n* **buffer-bind-key** "*command*" "*key*"
*n* **buffer-unbind-key** "*key*"

## DESCRIPTION

**buffer-bind-key** creates a key binding local to the current buffer, binding the command *command* to the keyboard input *key*. This command is particularly useful in conjunction with **major-mode** file hook (see **major-mode**(2)) allowing local key bindings dependent upon the context of the buffer.

The message line input is not effected by the current buffers local bindings.

**buffer-unbind-key** unbinds a user created local key binding, this command effects only the current buffer. If a negative argument is given to **buffer-unbind-key** then all the current buffer's bindings are removed.

## NOTES

The prefix commands cannot be rebound with this command.

Key response time linearly increases with each local binding added.

## SEE ALSO

**global-bind-key**(2), **ml-bind-key**(2), **osd-bind-key**(2), **major-mode**(2), **global-unbind-key**(2).

Ref: buffer-bind-key(2) File: m2cmd084.2 Date: 2026/01/17

## NAME

buffer-info - Status information on current buffer position

## SYNOPSIS

*n* **buffer-info** (**C-x =**)

## DESCRIPTION

**buffer-info** reports on the current and total lines and characters of the current buffer. It also gives the hexadecimal code of the character currently under the cursor.
The output of the command is displayed on the message line e.g.

```
Line 1845/3955 Col 0/0 Char 78267/167172 (46%) Win Line 99/48 ACol 0/0 char 10 (0x0a)
```

**$result**(5) is set to the same output string unless a numeric value of zero is given to the command in which case the values are given in a MicroEmacs list format and not printed to the message line; this information can be useful in macros.

## SEE ALSO

**$result**(5), **$mode-line**(5), **about**(2).

Ref: buffer-info(2) File: m2cmd014.2 Date: 2022/06/12

## NAME

buffer-mode - Change a local buffer mode
named-buffer-mode - Change a named buffer mode
add-mode - Set a local buffer mode
delete-mode - Remove a local buffer mode
unmark-buffer - Remove buffer change flag

## SYNOPSIS

*n* **buffer-mode** "*mode*" (**C-x m**)
*n* **named-buffer-mode** "*buffer-name*" "*mode*"
**add-mode** "*mode*"
**delete-mode** "*mode*"
**unmark-buffer**

## DESCRIPTION

**buffer-mode** changes the state of a given buffer mode, affecting only the current buffer. A buffer's mode affects the behavior of MicroEmacs. The **about**(2) command gives a list of the current global and buffer modes. Refer to Operating Modes for a description of the buffer modes.
The argument *n* when given, has the following meaning:

| Delete | Add | toggle | Mode |
|---|---|---|---|
| -1 | 1 | 0 | Use "mode" |
| -2 | 2 | 130 | autosv |
| -3 | 3 | 131 | backup |
| -4 | 4 | 132 | binary |
| -5 | 5 | 133 | cr |
| -6 | 6 | 134 | crypt |
| -7 | 7 | 135 | ctrlz |
| -8 | 8 | 136 | cursor |
| -9 | 9 | 137 | edit |
| -10 | 10 | 138 | exact |
| -11 | 11 | 139 | fence |
| -12 | 12 | 140 | hide |
| -13 | 13 | 141 | indent |
| -14 | 14 | 142 | justify |
| -15 | 15 | 143 | lf |
| -16 | 16 | 144 | lock |
| -17 | 17 | 145 | magic |
| -18 | 18 | 146 | nact |
| -19 | 19 | 147 | narrow |
| -20 | 20 | 148 | over |
| -21 | 21 | 149 | pipe |
| -22 | 22 | 150 | rbin |
| -23 | 23 | 151 | tab |
| -24 | 24 | 152 | time |
| -25 | 25 | 153 | undo |
| -26 | 26 | 154 | view |
| -27 | 27 | 155 | wrap |
| -28 | 28 | 156 | xlate |

Note that when omitted the default argument is 0, i.e. prompt for and toggle a mode.

**named-buffer-mode** changes the state of a given buffer mode for a given buffer which may not be the current buffer.
**add-mode** and **delete-mode** are macros which use buffer-mode to add and remove a buffer mode. **unmark-buffer** is also a macro which removes the edit flag from the current buffer.

They are defined for backward compatibility with MicroEMACS v3.8 and can be found in meme3_8.emf; add-mode is defined as follows:

```
define-macro add-mode
    ; Has the require mode been given as an argument, if so add it
    !force 1 buffer-mode @1
    !if &not $status
        ; No - use 1 buffer-mode to add a mode
        !nma 1 buffer-mode
    !endif
!emacro
```

## NOTES

When a buffer is created it inherits the current global mode state.

## SEE ALSO

Operating Modes, **global-mode**(2), **about**(2), **&bmode**(4).

Ref: buffer-mode(2) File: m2cmd006.2 Date: 2026/02/16

## NAME

capitalize-word - Capitalize word
lower-case-word - Lowercase word (downcase)
upper-case-word - Uppercase word (upcase)
lower-case-region - Lowercase a region (downcase)
upper-case-region - Uppercase a region (upcase)

## SYNOPSIS

*n* **capitalize-word** (**esc c**)
*n* **lower-case-word** (**esc l**)
*n* **upper-case-word** (**esc u**)
**lower-case-region** (**C-x C-l**)
**upper-case-region** (**C-x C-u**)

## DESCRIPTION

**capitalize-word** capitalizes the next *n* words.

**lower-case-word** changes the next *n* words to lower case.

**upper-case-word** changes the next *n* words to upper case.

**lower-case-region** changes all alphabetic characters in the marked region to lower case (see **set-mark**(2)).

**upper-case-region** changes all alphabetic characters in the marked region to upper case

## SEE ALSO

**set-mark**(2).

Ref: capitalize-word(2) File: m2cmd015.2 Date: 2022/06/12

## NAME

change-buffer-name - Change name of current buffer

## SYNOPSIS

*n* **change-buffer-name** "*buffer-name*" (**esc C-n**)

## DESCRIPTION

**change-buffer-name** changes the name of the current buffer to *buffer-name*. Buffer names must be unique as they act as the identity handle. By default the buffer name is derived from the buffer's file name excluding the path. This can lead to conflicts, when editing files with the same name and different paths, in which case a counter is appended to the end of the buffer name to make the name unique. For example:

```
File Name               Buffer Name

/etc/file.c             file.c
/tmp/file.c             file.c<1>
```

The numeric argument *n* is a bit-mask and is interpreted as follows:

`0x01` By default, or an argument is specified with bit 1 set, **change-buffer-name** fails if a buffer with the given name already exists. This behavior can be changed by specifying an argument with the first bit cleared, e.g. 0, in which case if a buffer with that name already exists then a counter as appended.

`0x02` Forces the buffer to adopt the given name *buffer-name*, changing the name of any existing buffer if required.

## SEE ALSO

**$buffer-fname**(5), **change-file-name**(2), **delete-buffer**(2).

Ref: change-buffer-name(2) File: m2cmd088.2 Date: 2022/06/12

## NAME

change-file-name - Change the file name of the current buffer

## SYNOPSIS

**change-file-name** "*file-name*" (**C-x n**)

## DESCRIPTION

**change-file-name** changes the file name of the current buffer to *file-name*. A validity check is made on the given file name and if found to be invalid (e.g. its a directory) the name is rejected.

## SEE ALSO

**change-buffer-name**(2), **write-buffer**(2).

Ref: change-file-name(2) File: m2cmd017.2 Date: 2022/06/12

## NAME

change-font - Change the screen font

## SYNOPSIS

*[X-Windows]*
*n* **change-font** [ "*font-name*" ]
*[macOS-App]*
*n* **change-font** [ "*font-name*" ]
*[IBM-PC / MS-DOS]*
**change-font** "*mode-no*" "*spec*"
*[Microsoft Windows]*
*n* **change-font** [ *charSet* ]
*n* **change-font** [ *charSet* "*font-name*" *weight width height* ]

## DESCRIPTION

**change-font** is a platform specific command which allows the displayable font to be queried and modified. The selection of font is determined by the monitor resolution and the capabilities of the graphics adapter.

This command is available on all systems except termcap. While MS-DOS version does not support the concept of different fonts, it does (or at least the graphics card does) support the concept of changing screen resolution, which has the effect of changing the font. Each platform takes different arguments and are considered independently, as follows:

### X-Windows

The X-Windows UNIX environments accept a single argument which is a fully qualified font name. Simply give the font X name and the font will change if it is available. The window size changes to attempt to retain the same number of rows and columns so ensure that when changing to a larger font then there is enough room (or a way) to resize a window which is larger than the actual screen.

On systems that support FreeType fonts, if the font name does not start with a '−' it is identified as a FreeType font, otherwise, if the font starts with a '−' it is treated as an X11 core font name. If the name is an empty string ("") the default FreeType monospace font is used.

FreeType font names can include attributes of the form `:<name>=value>` at the end of the font name, attributes include `:lang` and `:style` etc. When specifying the font size the `:size=#` attribute should be used, rather than the shorter '`<font-name>-#`', so that supporting macros can more reliably identify the name and size. MicroEmacs uses the `:weight` and `:slant` attributes when bit **0x10** of **$system**(5) is set, see **Use Font Styles** in **user-setup**(3) for more information.

The X core font string describes the attributes of the font in terms of it's size name etc. as follows:

*- foundry - family - weight - slant - width - adstyl - pixels - point - hres - vres - space - av - set*

Where

*foundry* The type of foundry that digitized and supplied the font.

*family* Font Family.

*weight* Modifies the appearance of the font, the *weight* is usually **medium** or **bold**.

*slant* Determines the orientation of the font. *slant* is usually **r**oman (upright), **i**talic or **o**blique.

*width* Describes the proportionate width of the font. Typical widths include **normal**, **condensed**, **narrow**, **double**.

*adstyl* Describes the additional style of the font, such as serif or sans serif etc. This is often an empty field.

*pixels* Pixel size of the font

*point* The resolution of the font in tenths of a **dpi** (i.e. dpi*10)

*hres* Horizontal resolution of the font in dpi.

*vres* Vertical resolution of the font in dpi.

*space* The spacing of the font. Typical spacing values include **m**onospaced (i.e. fixed width), **p**roportional and **c**haracter cell. Proportional fonts should be avoided as these will create rendering issues.

*av* Mean width of all font characters, measured in tenths of a pixel.

*set* Character set - this is made up of two fields, the character set standard and the set identifier, e.g. **iso8859-1**.

The default font used by MicroEmacs is the first font that is iso8859, of family fixed (then clean), of point size 14, 13 or 12 of medium or regular weight, i.e. the following are tried in order:

```
-*-fixed-medium-r-*-*-*-140-*-*-*-*-iso8859-*
-*-fixed-regular-r-*-*-*-140-*-*-*-*-iso8859-*
-*-fixed-medium-r-*-*-*-130-*-*-*-*-iso8859-*
...
-*-fixed-regular-r-*-*-*-120-*-*-*-*-iso8859-*
-*-clean-medium-r-*-*-*-140-*-*-*-*-iso8859-*
...
-*-clean-regular-r-*-*-*-120-*-*-*-*-iso8859-*
-*-fixed-regular-r-*-*-*-120-*-*-*-*-*-*
...
```

Try running **xfontsel**(1) in an xterm if struggling to find a suitable font. The font may also be changed in your **.Xdefaults** file by inserting the line:

```
MicroEmacs.font "-misc-fixed-medium-r-normal--13-*-*-*-c-80-iso8859-1"
```

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** Prompt for the new font name, if left as an empty string the default font, as described above, is selected. If this bit is not set then the current font is used.

**0x02** When set the new font is not applied, this bit is useful when information about a font is wanted.

**0x04** When set the command does not alter the value of variable **$result**(5). When not set **$result** is set to the following information about the font in a MicroEmacs list format:

```
"|<flg>|||<pixel-width>|<pixel-height>||<font-size>|<font-name>|"
```

The <**flg**> is an bitwise integer value where bit 1 indicates the font is not fixed width, i.e. the font spacing is proportional; the <flg> value is only given when the font name is given (i.e. bit **0x01** of argument **n** is set). The <**font-size**> will be set to 0 for X11 core fonts, or the font size for FreeType fonts, this field is empty on systems that do not support FreeType fonts.

**0x08** Not used by this platform.

**0x10** Not used by this platform.

**0x20** When set and the reported font height and sum of the ascent and descent differ, the smaller of the two values is used, however this may cause rendering issues ('debris' in the window). When clear, the larger value is used to minimise rendering issues.

### macOS-App

The macOS native app, distinct from the X11 based **mew** as it uses macOS windowing and drawing APIs directly, support a composite *font-name* consisting of two parts, the font name and size. The format of the string is "<*font-name*>`:size=`<*font-size*>", if the <*font-name*> is omitted the system's default monospaced font is used, if the `:size=`<*font-size*> is omitted a suitable size will be calculated based on the size of the primary monitor. Note that only monospaced fonts should be used, proportionally fonts will not render correctly.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** Prompt the user for the new font name, either using a macOS font selection dialog (see bit **0x08**) or by entering the font name manually. If this bit is not set then the current font is used.

**0x02** When set the new font is not applied, this bit is useful when information about a font is wanted.

**0x04** When set the command does not alter the value of variable **$result**(5). When not set **$result** is set to the following information about the font in a MicroEmacs list format:

```
"|<flg>|||<pixel-width>|<pixel-height>||<font-size>|<font-name>|"
```

The <**flg**> is an bitwise integer value where bit 1 indicates the font is not fixed width, i.e. the font spacing is proportional. The <**font-size**> is set to the given or calculated font size.

**0x08** When set with bit **0x01** a macOS font selection dialog is opened to allow the user to select their required font.

**0x10** Not used by this platform.

**0x20** Not used by this platform.

### IBM-PC / MS-DOS

MS-DOS may only change the screen resolution, the standard screen resolution is either 80 columns by 25 rows or 80 by 50. A more advanced graphics card can typically support up to 132 by 60, MicroEmacs in theory has no limit but it has only been tested up to this size.

The main problem with MS-DOS machines is that there is no standard and this is no exception. The graphics mode needed to get a 132 by 60 screen (if available) varies from one card to the next so MicroEmacs needs to know the graphic mode number your card uses to get your required screen resolution.

MicroEmacs can also attempt a little bit of magic to double the number of rows on the screen for a given screen resolution. This is how 50 lines are obtained from the standard 25 line mode 3. If the value of "*spec*" is non-zero then this is attempted, to the authors knowledge this will either work or not depending on the direction of the wind and no harm will befall the users equipment. However the author also quickly disclaims anything and everything, the user uses this at their own peril, like everything else.

MicroEmacs attempts to determine the new screen width and depth itself, in case this fails the commands **change-frame-width**(2) and **change-frame-depth**(2) may be used to correct the problem.
Following are the standard MS-DOS text modes:

```
change-font "2" "0"       ; Simple monochrome or EGA monitor, 80 by 25.
change-font "3" "0"       ; Simple EGA/VGA monitor, again 80 by 25.
change-font "3" "1"       ; Simple EGA/VGA monitor using spec, 80 by 50.
```

Most Trident cards support the following text mode:

```
change-font "86" "0"      ; Sweet 132 by 60
```

A Diamond Stealth supports the following mode:

```
change-font "85" "1"      ; Nice 132 by 50
```

Cirrus video cards (1MB) seem to support:

```
change-font "84" "1"      ; PT-526 (132x50)
```

Time to start digging out your graphics card manual!

### Microsoft Windows

The Microsoft Windows console version does not support the changing of fonts as the Windows Console does not officially support such a feature. However the **change-font** command can be used to inform MicroEmacs of the type of character set to use for copy text to and from the system clipboard.

The Microsoft Windows environments utilize font files to drive the display. When **change-font** is invoked, by default a font dialog is presented to the user to allow the font to be selected, the dialog only lists fixed width fonts as proportially based fonts are not supported due to the way in which MicroEmacs renders to the screen.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** Prompt the user for the new font details, either using a Windows font selection dialog (see bit **0x08**) or by entering the font details manually, see below. If this bit is not set then the current font is used.

**0x02** When set the new font is not applied, this bit is useful when information about a font is wanted.

**0x04** When set the command does not alter the value of variable **$result**(5). When not set **$result** is set to the following information about the font in a MicroEmacs list format:

```
"|<flg>|<char-set>|<weight>|<pixel-width>|<pixel-height>|<width>|<height>|<font-name>|"
```

The <**flg**> is an bitwise integer value where bit 1 indicates the font is not fixed width, bit 2 indicates it is a vector font and bit 4 that it is a TrueType font. Note that the <**width**> and <**height**> values are in logical units and usually differ to the actual size in pixels, see **width** and **height** below for more details.

**0x08** When set with bit **0x01** a Windows font selection dialog is opened to allow the user to select their required font.

**0x10** When set with bit **0x08** the Windows font selection dialog lists all fonts, not just fixed width ones. Use with care, most proportional fonts are unusable.

The default value when **n** is not specified is 9.

When entering the font details manually, the following information is required:

*charSet* The type of character set required, this is an integer value of:

```
  0 - ANSI or Western (True Type etc)
161 - Greek
162 - Turkish
204 - Russian
255 - OEM (or bitmapped)
```

*font-name*

The name of the font - maximum of 32 characters, use fixed mono fonts only. Proportional fonts may be specified but the cursor will not align with the characters on the screen.

An empty name ("") may be specified in which case no further details are requested and the default font is used for the given character set.

Note that, depending on the version of Windows, **Courier New** may not actually be a fixed mono font as might be expected.

*weight* The weight of the font. The values are defined as:

```
0 - Don't care (Automatically selected).
1 - Thin
2 - Extra Light
3 - Light
4 - Normal
5 - Medium
6 - Semi-Bold
7 - Bold
8 - Extra-Bold
9 - Heavy
```

Note that you may request a weight and it is not honored. Typically 4 and 7 are honored by most font definitions. 4 is typically used.

*width* The width of the font. Specifies the average width, in logical units, of characters in the requested font. If this value is zero, the font mapper chooses a ”closest match” value. The ”closest match” value is determined by comparing the absolute values of the difference between the current device’s aspect ratio and the digitized aspect ratio of available fonts.

Note that if the width is specified as zero then the height should be specified and the width will be automatically selected.

*height* The height of the font. Specifies the desired height, in logical units, of the requested font’s character cell or character. (The character height value is the character cell height value minus the internal-leading value.) If this value is greater than zero, the font mapper matches it against available character cell height values; if this value is zero, the font mapper uses a default height value when it searches for a match; if this value is less than zero, the font mapper matches it against available character height values.

Note: as with the weight the width and height may not be honored if the font cannot support the specified width/height in which case the closest matching height is automatically selected

### Notes on the Standard Windows Configuration

For releases prior to ’99, the **Terminal** font is the standard MS-DOS font used for the MS-DOS window. This is an OEM fixed width character set which contains all of the conventional symbols found in the DOS shell.

Releases of MicroEmacs post ’99 may utilise any of the windows fonts, typically `Courier New` or `Lucida Console` are used, these provide the best screen rendering of characters. `Lucida Console` is slightly better with a smaller font size as this allows a ’`1`’ (one) and ’`l`’ (lower case L) to be distinguished.

The **Terminal** fonts are the same as shown in the DOS window the last 2 arguments are the width x height, the terminal equivalents (Bit Mapped) are commented here.

**7x12** Regular weight seems to offer the best resolution for 14/15” monitors.

**6x8** Regular weight is more suitable for 17-21” monitors which offer better resolutions.

The best options for the fonts are defined as follows:

```
;Standard Terminal Fonts - standard weight
;change-font 0 "Terminal" 4  4  6
change-font 0 "Terminal" 4  6  8
;change-font 0 "Terminal" 4  8  8
;change-font 0 "Terminal" 4  5 12
;change-font 0 "Terminal" 4  7 12
;change-font 0 "Terminal" 4  8 12
;change-font 0 "Terminal" 4 12 16
;change-font 0 "Terminal" 4 10 18

;Standard Terminal Fonts - heavy weight
;change-font 0 "Terminal" 7  4  6
;change-font 0 "Terminal" 7  6  8
;change-font 0 "Terminal" 7  8  8
;change-font 0 "Terminal" 7  5 12
;change-font 0 "Terminal" 7  7 12
```

```
;change-font 0 "Terminal" 7  8 12
;change-font 0 "Terminal" 7 12 16
;change-font 0 "Terminal" 7 10 18
```

The "**Courier New**" font is not actually a fixed mono font as might be expected.

## SEE ALSO

**change-frame-width**(2), **change-frame-depth**(2), **$result**(5), **user-setup**(3).

Ref: change-font(2) File: m2cmd027.2 Date: 2026/05/25

## NAME

change-frame-depth - Change the number of lines on the current frame
change-frame-width - Change the number of columns on the current frame

## SYNOPSIS

*n* **change-frame-depth** [ "*depth*" ]
*n* **change-frame-width** [ "*width*" ]

## DESCRIPTION

**change-frame-depth** changes the depth of the current frame, if the numeric argument *n* is given then the frame depth is changed by *n* lines. If *n* is not specified the user is prompted for the new *depth* and the frame depth will be changed to this value. It is assumed that the screen can draw the requested *n* lines and MicroEmacs draws the lines at the users peril.

A change in depth causes all of the internal windows currently displayed in the frame to be re-sized, the vertical position of the windows are modified to match the new screen dimension, the horizontal position of the windows remains unaltered. If the window is down-sized and the currently displayed windows are not able to fit into the new screen space then all windows are deleted with the exception of the current window.

**change-frame-width** changes the width of the current frame, if the numeric argument *n* is given then the frame width is changed by *n* characters. If *n* is not specified the user is prompted for the new *width* and the frame width will be changed to this value. It is assumed that the screen can draw the requested *n* columns and MicroEmacs draws them at the users peril. The windows are reorganized as **change-frame-depth** working horizontally rather than vertically.

## NOTES

Within windowing environments such as **X-Windows** and **Microsoft Windows** these commands cause the canvas window to be re-sized to accommodate the change in screen size.

In MS-DOS and UNIX Termcap environments the physical size of the screen is determined by the characteristics of the display adapter. **change-frame-depth** may be used to correct anomalies (usually on portables) in the displayable screen area and the graphics mode. e.g. In DOS the graphics mode utilizes 50 lines, and only 47 lines are viewable. In this case change the screen depth to 47 and MicroEmacs will not utilize the remaining lines which are not viewable.

## SEE ALSO

**$frame-depth**(5), **$frame-width**(5).

Ref: change-frame-depth(2) File: m2cmd018.2 Date: 2024/05/21

## NAME

change-window-depth - Change the depth of the current window
grow-window-vertically - Enlarge the current window (relative change)
shrink-window-vertically - Shrink the current window (relative change)
resize-window-vertically - Resize the current window (absolute change)

## SYNOPSIS

*n* **change-window-depth** [ "*depth*" ]
*n* **grow-window-vertically**
*n* **shrink-window-vertically**
*n* **resize-window-vertically**

## DESCRIPTION

**change-window-depth** changes the depth of the current window, if the numeric argument *n* is given then the window depth is changed by *n* lines. If *n* is not specified the user is prompted for the new *depth* and the window depth will be changed to this value. The command aborts if the requested size cannot be achieved (the window becomes too small or a neighbouring one does).

## NOTES

Commands **grow-window-vertically**, **shrink-window-vertically** and **resize-window-vertically** were replaced by the new **change-window-depth** command in April 2002. Following are macro implementations of the old commands:

```
define-macro grow-window-vertically
    @# change-window-depth
!emacro

define-macro shrink-window-vertically
    &neg @# change-window-depth
!emacro

define-macro resize-window-vertically
    !if &not @?
        !abort
    !endif
    change-window-depth @#
!emacro
```

## SEE ALSO

**change-window-width**(2), **resize-all-windows**(2), **split-window-vertically**(2).

Ref: change-window-depth(2) File: m2cmd063.2 Date: 2022/06/12

## NAME

change-window-width - Change the width of the current window
grow-window-horizontally - Enlarge current window horizontally (relative)
shrink-window-horizontally - Shrink current window horizontally (relative)
resize-window-horizontally - Resize current window horizontally (absolute)

## SYNOPSIS

*n* **change-window-width** [ "*width*" ]
*n* **grow-window-horizontally**
*n* **shrink-window-horizontally**
*n* **resize-window-horizontally**

## DESCRIPTION

**change-window-width** changes the width of the current window, if the numeric argument *n* is given then the window width is changed by *n* characters. If *n* is not specified the user is prompted for the new *width* and the window width will be changed to this value. The command aborts if the requested size cannot be achieved (the window becomes too small or a neighbouring does).

## EXAMPLE

Refer to `mouse.emf` for an example of window growth using the mouse to manipulate the size of the windows.

## NOTES

Commands **grow-window-horizontally**, **shrink-window-horizontally** and **resize-window-horizontally** were replaced by the new **change-window-width** command in April 2002. Following are macro implementations of the old commands:

```
define-macro grow-window-horizontally
    @# change-window-width
!emacro

define-macro shrink-window-horizontally
    &neg @# change-window-width
!emacro

define-macro resize-window-horizontally
    !if &not @?
        !abort
    !endif
    change-window-width @#
!emacro
```

## SEE ALSO

**change-window-depth**(2), **resize-all-windows**(2), **split-window-horizontally**(2).

Ref: change-window-width(2) File: m2cmd152.2 Date: 2022/06/12

## NAME

command-apropos - List commands involving a concept

## SYNOPSIS

*n* **command-apropos** "*string*" (**C-h a**)

## DESCRIPTION

**command-apropos** compiles a list of all commands with *string* in their name, also giving their current key bindings. Only commands that are not hidden are displayed, if a numeric argument of 0 is supplied then all commands are displayed, irrespective of their visibility.

## EXAMPLE

To find all of the commands with "`command`" in their name space then issue the command "`C-h a command`" which generates a list of commands such as:

```
abort-command ................. "C-g"
                                "esc C-g"
                                "C-x C-g"
command-apropos ............... "C-h a"
command-complete
execute-named-command ......... "esc x"
help-command .................. "C-h C-c"
ipipe-shell-command ........... "esc \\"
list-commands ................. "C-h c"
pipe-shell-command ............ "esc !"
                                "esc @"
                                "C-x @"
shell-command
```

## SEE ALSO

**describe-bindings**(2).

Ref: command-apropos(2) File: m2cmd009.2 Date: 2022/06/12

## NAME

command-wait - Conditional wait command

## SYNOPSIS

*n* **command-wait**

## DESCRIPTION

When a positive argument *n* is given **command-wait** waits for *n* milliseconds before returning, this wait cannot be interrupted. If a negative argument is given, **command-wait** waits for -*n* milliseconds but the command will return if the user interrupts with any input activity (i.e. presses a key).

When no argument or an argument of 0 is given **command-wait** waits until either the calling command's **.wait** command variable is undefined or it is set to false (0). When no argument is given events such as user input, screen updates etc. are still processed, this interface is best used when a main macro must wait and process input until an exit criteria has been met, the user input is best processed by setting the **$buffer-input**(5) variable to a second macro. The macro **gdiff**(3) uses this command in this way, also see a comprehensive example below.

When no argument is given user input and screen update events are ignored, this is typically used when MicroEmacs is used in pipe mode, see option **-p** of **me**(1), see the example below.

## EXAMPLE

The following macro code will display a message on the screen for a fixed 5 seconds:

```
16 screen-poke 10 10 0 "Hello World!"
5000 command-wait
```

Similarly the following macro code will display a message for up to 5 seconds or till the user presses a key:

```
16 screen-poke 10 10 0 "Hello World!"
-5000 command-wait
```

The following example shows how command-wait can be used to run a system process making MicroEmacs wait until the process has completed before continuing, yet giving feedback from the process to the user and allowing the user to kill the launched process. The process run is a recursive directory listing which will take a long time and give a lot of feed back, to kill the process with a prompt use delete-buffer (bound to '`C-x k`') or use abort-command (bound to '**C-g**') to kill immediately:

```
0 define-macro test-ipipe-macro
    ; add extra macro code to process output here
    !if &not @#
        ; the process has finished
        set-variable .test.wait 0
    !endif
!emacro

0 define-macro test-input-macro
    !if &seq @cck "redraw"
        @# screen-update
        !return
    !elif &seq @cck "callback"
        !force execute-named-command @cc
        !return
    !elif &seq @cck "idle-pick"
```

```
        !abort
    !elif &seq @cc "delete-buffer"
        !force !force set-variable #l0 @mc1 "Abort test [y/n]? " "nNyY"
        !if &and &iseq #l0 "y" .test.wait
            !force ipipe-kill
        !endif
    !elif &seq @cc "abort-command"
        !force ipipe-kill
        set-variable .test.wait 0
    !endif
!emacro

define-macro test
    set-variable #l0 &con &band $system 0x100 "dir /s c:\\" "ls -R /"
    set-variable .wait 1
    !force !force !force 0x82 ipipe-shell-command #l0 "*test*" test-ipipe-macro
    set-variable #l2 $status
    !if &band #l1 1
        !force 0 delete-buffer "*cc-cmdline*"
    !endif
    find-buffer "*test*"
    set-variable $buffer-input test-input-macro
    !force !force !force command-wait
    set-variable $buffer-input ""
    ml-write "[test complete]"
!emacro
```

The following macro code demonstrates how MicroEmacs can be used as a command-line tool, piping the output of another sub-process as it goes. In this example MicroEmacs executes a system directory listing command and prints the output to stdout using ml-write.

```
ml-write "Loading pipetest.emf"

define-macro pipetest-ipipe
    !if &not @#
        end-of-buffer
    !else
        goto-alpha-mark "I"
    !endif
    ml-write &spr "In pipetest-ipipe %d %d %d" @# .line $window-line
    set-variable #l0 $window-line
    set-variable $window-line .line
    !while &les $window-line #l0
        -1 ml-write @wl
        forward-line
    !done
    set-variable .line $window-line
    !if &not @#
        -2 ml-write "Process finished"
        set-variable .start-up.wait 0
    !endif
!emacro

define-macro start-up
    ml-write "Got into start-up"
    set-variable .wait 1
    set-variable .pipetest-ipipe.line 1
    set-variable #l0 &con &band $system 0x100 "dir" "ls -l"
    -2 ml-write &cat "About to execute: " #l0
    0xc0 ipipe-shell-command #l0 "*test*" pipetest-ipipe
    0 command-wait
    ml-write "start-up continues"
    exit-emacs
!emacro

ml-write "Loading pipetest.emf complete"
```

To run this example, save the above macro code to `pipetest.emf` and then run:

**me -n -p @pipetest.emf**

Use the **-P** option for additional debug lines, printed to stderr. Windows users must use a console capable version of MicroEmacs, i.e. mec32.exe or mecw32.exe.

## NOTES

When the given wait time is smail the system's clock frequency can play a large part in the accuracy, for example on Windows the documented smallest time for **SetTimer**(3) is 10 milisconds (see **USER_TIMER_MINIMUM**) but in practice tends to be more like 16ms. The following code can be run to test the granularity of a system:

```
set-variable #g0 $unix-time
1 command-wait
set-variable #g1 $unix-time
1 command-wait
set-variable #g2 $unix-time
1 command-wait
set-variable #g3 $unix-time
1 command-wait
set-variable #g4 $unix-time
list-variables
```

The values of `#g0` to `#g5` should increase by just over 0.001 (1ms), any more indicates a limitation of the system clock. This issue can be demonstrated by the following example which attempts to create a stable 'frame-rate' by allowing for the time taken by the code on each loop, the desired frequency can be adjusted by the changing the first variable:

```
; change the following variable to change the target delay (in ms)
set-variable #l9 100
set-variable #l8 0
set-variable #l7 $unix-time
set-variable #l1 -1
set-variable #l2 0
!repeat
  set-variable #l0 $unix-time
  ; Do work here...
  ml-write &spr "Loop: %4d %s" &inc #l1 1 &mid #l0 7 6
  set-variable #l8 &add #l8 #l2
  ; This gets the current time and removes the time at the start of our loop work, this is removed from
  !iif &les &set #l2 &sub #l9 &fmu 1000 &fsub $unix-time #l0 1  set-variable #l2 1
  #l2 command-wait
!until &equ #l1 100
set-variable #l6 $unix-time
set-variable #l8 &add #l8 #l2
ml-write &spr "Result: %d iterations took %.3fsec, average time %.2fms, average value for command-wait %
```

When the desired frequency is large or the system clock frequency is large this code is accurate, but if the clock frequency is, say, 16ms then a desired frequency of 20ms will typically have a large 12ms error caused by the system clock (`command-wait` will end up waiting 2 times 16ms on each loop). If accuracy is required, use an appropriate OS, such as any UNIX system.

the use of **create-callback**(2) intead of **command-wait** typically creates a better user experience as it allows the system to continue to functioning normally while waiting for the next loop.

## SEE ALSO

**ml-write**(2), **create-callback**(2), **$time**(5), **$unix-time**(5), **$buffer-input**(5).

## NAME

compare-windows - Compare buffer windows, ignore whitespace.
compare-windows-exact - Compare buffer windows, with whitespace.

## SYNOPSIS

*n* **compare-windows**
**compare-windows-exact**

## DESCRIPTION

**compare-windows** compares the textural content of ALL the current windows from their current cursor position. These commands are generally used to locate the next difference in the windows displayed. Returns `TRUE` if the buffers of the windows do not differ from the current position to the end of the file (inclusive), else returns `FALSE` setting the cursor of each buffer to the first point of difference.

The argument *n* can be used to change the default behavior of compare-windows, *n* is a bit based flag where:

**0x01** When enabled the compare operation ignores white-space differences, when clear an exact white-space match is performed.

**0x02** When enabled the compare operation is case-insensitive, when clear an exact letter match is performed.

When omitted a case-sensitive compare which ignores white-space differences is performed.

**compare-windows-exact** is a macro short cut for *0 compare-windows*, forcing an exact white-space & letter-case comparison.

## NOTES

A window can be excluded from **compare-windows** by setting bit `0x08` of its **$window-flags**(5) variable.

## SEE ALSO

**diff**(3), **diff-changes**(3), **gdiff**(3), **$window-flags**(5).

Ref: compare-windows(2) File: m2cmd020.2 Date: 2022/06/12

## COMPATIBILITY

JASSPA MicroEmacs is based on the original version of **MicroEMACS** produced by Danial Lawrence at revision 3.8, the source files were obtained in approximately 1990. The exact origin of the files is unknown. In that period of time the source files have undergone an awful lot of change, without reference to the subsequent releases made of MicroEMACS by Danial Lawrence (due to no network access). As a result the JASSPA version of **MicroEmacs** does not include any modifications or features that may have been implemented since. This version of **MicroEmacs** has been tailored to suite the requirements of a small group of individuals who have used the editor on a daily basis across a limited number of platforms, for a variety of very different tasks and operating requirements.

This version of MicroEmacs is biased towards UNIX environments, MS-DOS and Microsoft Windows ports have been performed however they are heavily influenced by UNIX and inherit UNIX characteristics wherever possible. The intention is that programmers, and alike, may move across platforms using a common editor environment without being frustrated by the idiosyncrasies of different platforms. The most noticeable platform is the Microsoft Windows platform which mimics the X-Windows cut and paste mechanism within the MicroEmacs environment. If you want a Windows style environment then use **Notepad**(1) or **Wordpad**(1), this editor is not for you !!

The gross changes to **MicroEmacs** are summarized as follows:

- Macro language interpreter re-written allowing an unlimited number of named macros to be supported. The macro implementation allows new commands to be created by the user, as opposed to continually extending the underlying command set. The named macros are transparent to the user, appearing as built in commands on the command line. Macro command set significantly increased. Support for global, buffer and register variables within the macro language.
- Display drivers re-written providing color hilighting support on most platforms. A macro interface allows information to be written directly to the display canvas allowing the screen to be annotated with additional transient information.
- Support for X-Window screen type in UNIX environments. Microsoft Window's environments (3.x, '95, NT) treated in the same was as X-Windows - this may be unorthodox for existing Window's users, UNIX users will find it more comfortable.
- Introduction of integrated spell checker. Support includes correction word guessing, word auto-correction and double word detection. Ignore and personal dictionaries supported.
- Horizontal window splitting.
- Introduction of scroll bars on all platforms that support a mouse. The scroll bar implementation is platform independent.
- Command and file completion available on all platforms. Most commands support a command history allowing previous command invocations to be recalled.
- Session history file kept, allowing the previous edit session to be reinstated.
- Undo capability, allows previous edits to be undone when mistakes are made.

- Backup capability, Includes a periodic timed backup while an editing session is in progress. The timed backup is automatically recovered by the next session in situations where the system (or editor) crashes.
- A regular expression incremental search becomes the default search forward mechanism.
- Support for abbreviation files allowing frequently used constructs to be automatically expanded.
- Automatic time stamping of files, allowing the edit time to be automatically maintained in the source file(s).
- Introduction of an electric 'C' mode. Editor intelligently handles the layout of 'C' files (under user control).
- Improved documentation text mode providing left/right/center and both justification methods with inclusion for bullet points. Automatic justification may be continually performed as text is entered, thereby maintaining the paragraph in the correct format.
- Integrated on-line help facilities. All commands are documented on-line. New macros may be documented within the macro files and become part of the help system.
- File type determination system, based on either the file name or embedded file text allows file type specific macros (hooks) to be applied, thereby configuring the editor into the correct mode for the file type.
- Introduction of special MicroEmacs search path allowing all of the standard configuration files to be utilized from a shared directory.

The name space of JASSPA MicroEmacs differs from the original MicroEMACS and has become more compliant with the GNU implementation of Emacs. A list of the original MicroEMACS verses the new command name set is as follows, executing the compatibility macro file `meme3_8.emf` will create macro versions of these commands:

**add-global-mode** => global-mode
**add-mode** => buffer-mode
**apropos** => command-apropos
**backward-character** => backward-char
**begin-macro** => start-kbd-macro
**beginning-of-file** => beginning-of-buffer
**bind-to-key** => global-bind-key
**buffer-position** => buffer-info
**case-region-lower** => lower-case-region
**case-region-upper** => upper-case-region
**case-word-capitalize** => capitalize-word
**case-word-lower** => lower-case-word
**case-word-upper** => upper-case-word
**change-screen-depth** => change-frame-depth
**change-screen-width** => change-frame-width
**clear-message-line** => 0 ml-write
**ctlx-prefix** => prefix 2
**delete-global-mode** => global-mode

**delete-mode** => buffer-mode
**delete-next-character** => forward-delete-char
**delete-next-word** => forward-kill-word
**delete-previous-character** => backward-delete-char
**delete-previous-word** => backward-kill-word
**end-macro** => end-kbd-macro
**end-of-file** => end-of-buffer
**execute-command-line** => execute-line
**execute-macro** => execute-kbd-macro
**execute-macro-#** => *Deleted*
**file-name-insert** => insert-file-name
**forward-character** => forward-char
**grow-window** => grow-window-horizontally
**handle-tab** => tab
**i-shell** => shell
**incremental-search** => isearch-forward
**kill-to-end-of-line** => kill-line
**meta-prefix** => prefix 1
**move-window-down** => scroll-down
**move-window-up** => scroll-up
**name-buffer** => change-buffer-name
**next-line** => forward-line
**next-page** => scroll-down
**next-paragraph** => forward-paragraph
**next-word** => forward-word
**open-line** => insert-newline
**pipe-command** => pipe-shell-command
**previous-line** => backward-line
**previous-page** => scroll-up
**previous-paragraph** => backward-paragraph
**previous-word** => backward-word
**quote-character** => quote-char
**redraw-display** => recenter
**restore-window** => goto-position
**reverse-incremental-search** => isearch-backward
**save-file** => save-buffer
**save-window** => set-position
**scroll-next-down** => scroll-next-window-down
**scroll-next-up** => scroll-next-window-up
**search-reverse** => search-backward
**select-buffer** => find-buffer
**set** => set-variable
**shrink-window** => shrink-window-vertically
**split-current-window** => split-window-vertically
**top-bottom-switch** => *Deleted*
**transpose-characters** => transpose-chars
**unbind-key** => global-unbind-key
**update-screen** => screen-update
**write-message** => ml-write

Ref: Compatibility(2) File: m2cmd201.2 Date: 2025/08/28

## NAME

copy-region - Copy a region of the buffer

## SYNOPSIS

*n* **copy-region** (**esc w**)

## DESCRIPTION

**copy-region** copies all the characters between the cursor and the mark set with the **set-mark**(2) command into the kill buffer (so they can later be yanked elsewhere).

If the last command also entered text into the kill buffer (or the **@cl**(4) variable is set to one of these commands) the **copy-region** text is appended to the last kill.

The numeric argument *n* determines how the copy operates when a narrow region, defined by **narrow-buffer**(2), is encountered. The default operation, when the number argument is `1`, is to copy all of the text contained within the narrowed region. When the numeric argument is `0` then narrowed region is NOT copied and the text is copied as it appears in the buffer.

If a negative argument is given, the current current region is not copied, instead the current kill buffer is copied to the system clipboard (Windows only).

## USAGE

To copy text from one place to another, using the **copy-region** command, the following operations are performed:

- Move to the beginning of the text you want to copy.
- Set the mark there with the set-mark (**esc-space**) command.
- Move the point (cursor) to the end of the text.
- Use **copy-region** to copy the region you just defined. The text will be saved in the kill buffer. (If you accidentally delete the text use yank (**C-y**) immediately or undo (**C-x u**) to restore the text).
- Move the point to the place you want the text to appear.
- Use the yank (**C-y**) command to copy the text from the kill buffer to the current point.

Repeat the last two steps to insert further copies of the same text.

## NOTES

Windowing systems such as X-Windows and Microsoft Windows utilize a global windowing kill buffer allowing data to be moved between windowing applications (*cut buffer* and *clipboard*, respectively). Within these environments MicroEmacs automatically interacts with the windowing systems kill buffer, the last MicroEmacs **copy-region** entry is immediately available for a paste operation into another windowing application.

The numeric argument *n* was introduced in the '04 release.

## SEE ALSO

**exchange-point-and-mark**(2), **kill-region**(2), **narrow-buffer**(2), **set-mark**(2), **yank**(2).

Ref: copy-region(2) File: m2cmd022.2 Date: 2025/08/28

## NAME

count-words - Count the number of words in a region

## SYNOPSIS

*n* **count-words** (**esc C-c**)

## DESCRIPTION

**count-words** counts the number of words in the current buffer and between the **set-mark**(2) position and the current cursor position. The command also gives statistics on the average characters per word, the number of characters and the number of lines. The output appears on the message line in a format such as:

```
Buffer:  3959 Words (4.7), 23394 Chars, 274 Lines Region:
154 Words (5.2), 983 Chars, 10 Lines
```

The **$result**(5) variable is also set to the same output string.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When clear the command gives the information in the form of a MicroEmacs list and does not print it to the message line; this interface is more useful to macros.

**0x02** When set whole buffer information is included, i.e. the first half, when clear this information is not computed or included.

**0x04** When set buffer region information is included, i.e. the second half; if the current buffer does not have a region an error is returned unless bit **0x08** is also set.

**0x08** When set no error is thrown if buffer region information is requested but the buffer does not have a region, if bit **0x01** is clear `-1` is returned for all 4 values.

When *n* is not specified, the default value is `0x0f`, i.e. print both whole buffer and region information without error.

## SEE ALSO

**$result**(5), **buffer-info**(2), **set-mark**(2).

Ref: count-words(2) File: m2cmd023.2 Date: 2024/10/06

## NAME

create-callback - Create a timer callback

## SYNOPSIS

*n* **create-callback** "*command*"

## DESCRIPTION

**create-callback** creates a timer based callback command. The given *command* is called back in *n* milliseconds time. This can be used by the user to monitor system events (such as incoming mail). The command is called only once, but if the command creates a callback of itself a loop is created.

If a negative argument *n* is given any pending callback for *command* is cancelled.

## EXAMPLE

The following example creates a callback that is invoked every 10 minutes.

```
define-macro Example-callback
    ml-write "It was 10 minutes since you last saw me!"
    600000 create-callback Example-callback
!emacro
Example-callback
```

The following example reimplements the test given in help page for **command-wait**(2), using **create-callback** in place of **command-wait**:

```
; change the following variable to change the target delay (in ms)
set-variable #g9 100
set-variable #g8 0
set-variable #g7 $unix-time
set-variable #g1 -1
set-variable #g2 0
define-macro Callback-test
  set-variable #g0 $unix-time
  set-variable #g8 &add #g8 #g2
  !if &equ &inc #g1 1 100
    ml-write &spr "Result: %d iterations took %.3fsec, average time %.2fms, average value for command-wa
  !else
    ; Do work here...
    ml-write &spr "Loop: %4d %s" #g1 &mid #g0 7 6
    ; This gets the current time and removes the time at the start of our work, this is then removed fro
    !iif &les &set #g2 &sub #g9 &fmu 1000 &fsub $unix-time #g0 1  set-variable #g2 1
    #g2 create-callback Callback-test
  !endif
!emacro

Callback-test
```

Global register variables are being used for ease but should not be for any code needing to be reliable!

## NOTES

A call-back cannot interrupt while MicroEmacs is active, instead the call-back is delayed until MicroEmacs becomes inactive. MicroEmacs is considered to be inactive when it is waiting for user input, this could be during the execution of another macro. If a command or macro requires no user input then once execution has started, it cannot be interrupted by a call-back macro.

The resolution of the clock is platform dependent, some platforms limit the minimum timer period to 10 milliseconds.

MicroEmacs does not guarantee to service the callbacks within any set time constraints, the resultant callback intervals may be of a slightly different duration than requested.

When a callback macro is executed, the key given by **@cck**(4) is "`callback`. If the current buffer has a **$buffer-input**(5) command set, this command will be called instead of the callback command with **@cc** and **@cck** set appropriately. It is the responsibility of the input macro to deal with the callback.

## SEE ALSO

**$auto-time**(5), **define-macro**(2).

Ref: create-callback(2) File: m2cmd024.2 Date: 2025/09/09

## NAME

create-frame - Create a new frame

## SYNOPSIS

*n* **create-frame** (**C-x 5 1**, **C-x 5 2**)

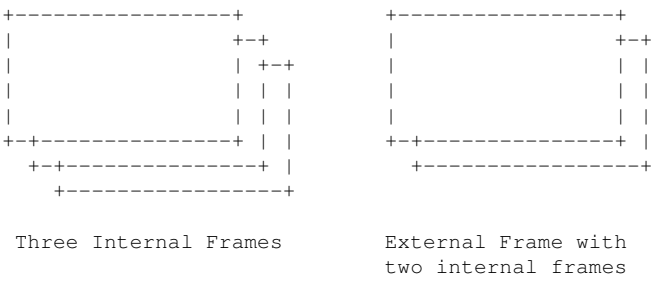
## INTODUCTION

A **frame** in MicroEmacs is the screen canvas on which all of the windows are attached. MicroEmacs allows the creation of new frames, creating a new blank canvas, which may be populated with new windows. Frames may be swapped and the window layout for a given frame is preserved until the frame is displayed again. There are two types of frame:

Internal
: An internal frame uses the same display window (or console) hiding any other frames. On creation, any existing frame is hidden and a new frame blank takes its place. Internal frames are supported on all platforms.

External
: An external frame creates a new display window allowing both the original frame and new frame to be visible in different windows. External frames are only available on windowing platforms such as X-Windows and Microsoft Windows. External frames may include internal frames which are paged.

This is shown in the following diagram:




The user may move to the next frame using **next-frame**(2) which swaps the current frame for the next frame. External frames may also be accessed via the window manager by selecting the window.

A frame may be subsequently deleted with **delete-frame**(2).

## DESCRIPTION

**create-frame** creates a new frame for the current MicroEmacs session, the new frame is assigned a unique identity **$frame-id**(5) which exists for the life of the frame.

The numeric argument *n* defines the type of frame to be created. An argument of `2` creates an external frame, an internal frame is created with an argument of `1`. Where no argument is specified an external frame is created by default on systems which support them otherwise an internal frame is created.

## NOTES

MicroEmacs is not multi-threaded and only one frame may be active at any one time (the complexity of being able to run a command in one frame while editing in another would rapidly lead it away from the 'Micro' status). Therefore if a command is left active (such as a search) in one frame and the focus is changed to another frame the input is *sent* to the frame with the active command and the message `[NOT FOCUS]` appears in the message-line of the frame with the focus.

**create-frame** is useful in macros that rely on a window layout, as the macro can preserve the users current window layout by creating a new internal frame in which to run.

## SEE ALSO

**delete-frame**(2), **next-frame**(2), **$frame-id**(5).

Ref: create-frame(2) File: m2cmd161.2 Date: 2022/06/12

## NAME

define-help - Define help information

## SYNOPSIS

**define-help** "*string*" ["*section*"]

*Free form text*

**!ehelp**

## DESCRIPTION

**define-help** provides a mechanism to define help information for commands and variables within macro files. The command allows user defined macros to be documented with help information that is accessible from the command line via the normal help commands such as **help-item**(2).

The help information is typically embedded in the macro file with the macro command that it is documenting. When the macro file is loaded then the help information is loaded and integrated into the existing help database.

*string* is the name of the item that is being defined, *section* defines what section the item belongs to. Following is a table of standard MicroEmacs sections:

| | |
|---|---|
| 1 | MicroEmacs command line arguments. |
| 2 | Built-in commands. |
| 2m | MicroEmacs buffer modes. |
| 3 | Macro commands. |
| 4 | Macro language commands. |
| 5 | MicroEmacs variables. |
| 8 | MicroEmacs file formats. |

When *section* is omitted is defaults to the general section which is usually used for the higher level help pages.

Text following the **define-help** line contains the help information, this is a free form text area that is reproduced when the help information is requested. The end of the text area is delimited by a **!ehelp** construct. The help text is usually displayed using a special hilighting scheme to control the colors and hyper-text links to other help pages. As a result the text may contain escape ('^[') key sequences, see **ehf**(8) for more information on the format.

## EXAMPLE

The following example is a define-help representation for the **paragraph-to-line**(3) macro.

```
define-help "paragraph-to-line" "3"

^[cENAME^[cA

     paragraph-to-line - Convert a paragraph to a single line
$a


^[cESYNOPSIS^[cA
```

```
     n paragraph-to-line


^[cEDESCRIPTION^[cA

     paragraph-to-line  reduces  each of the  next n  paragraphs  of text to a
     single  line.  This  is  used to  prepare  a  document  to go into a word
     processor environment where end of line marks represent paragraph marks.


^[cENOTES^[cA

     This command is a macro defined in format.emf.


^[cESEE ALSO^[cA

     ^[ls^[lm^[cGfill-paragraph(2)^[cA^[le.

!ehelp
```

## SEE ALSO

**ehf**(8), **help-item**(2), **define-macro**(2), **help-command**(2), **help-variable**(2).

Ref: define-help(2) File: m2cmd093.2 Date: 2022/06/12

## NAME

define-macro - Define a new macro

## SYNOPSIS

*n* **define-macro** *macro-name*

*Macro body*

**!emacro**

## DESCRIPTION

**define-macro** starts the definition of an macro named *macro-name*, only used within macro files or buffers. After the above header line, the body of the macro is added, one command or expression on a line. The macro definition is completed by the !emacro directive.

The numeric argument *n*, specified as zero, defines the macro as private such that it does not appear on a command completion list. A zero argument is generally used on helper macro's that form part of a larger macro. If the argument is omitted, or non-zero, then the macro appears in the command completion list.

See **execute-file**(2) for a complete definition and examples of the MicroEmacs macro language.

Once the macro has been defined, it becomes indistinguishable from a standard MicroEmacs command, i.e. **execute-named-command**(2) (`esc x`) can be used to execute the macro and **global-bind-key**(2) can be used to globally bind the command to a key combination.

There are no restrictions on the number of macros that may be defined, provided that the name space is managed properly. Consideration must be given as to when any additional macros that are created are loaded into MicroEmacs. We usually like start-up to be rapid and macros are loaded as and when requested by the user, or by the buffer major-modes as new files are loaded (see **major-mode**(2) and **define-macro-file**(2)).

User defined macros may be documented with on-line help by including a **define-help**(2) construct within the macro file.

## EXAMPLE

The following are two standard macros provided with MicroEmacs. The first is a macro called **clean**, this strips trailing white space from the ends of lines in a file and removes blank lines from the end of the file.

```
define-macro clean
    ;
    ; Prepare to clean up file.
    ; Remember line & magic mode
    set-variable #l0 $window-line
    set-variable #l1 &not &bmod magic
    !if #l1
        1 buffer-mode "magic"
    !endif
    ;
    ; Get rid of trailing white space on EOL
    beginning-of-buffer
    replace-string "[\t ]+$" "\\0"
    beginning-of-buffer
```

```
    replace-string "[ ]+\t" "\t"
    ;
    ; Strip trailing blank lines.
    end-of-buffer
    backward-line
    !while &and &gre $window-line 1 &sequal @wc "\n"
        kill-line
        backward-line
    !done
    ;
    ; Clean up - restore buffer modes etc.
    ; Move back to starting line & restore original magic mode
    !force goto-line #l0
    !if #l1
        -1 buffer-mode "magic"
    !endif
    screen-update
    ml-write "Cleaned up file."
!emacro
```

The second example converts all of the <tab> characters in the file to their <SPACE> character equivalent.

```
;
; tabs-to-spaces.
; Convert all of the tabs to spaces.
define-macro tabs-to-spaces
    ; Remember line
    set-variable #l0 $window-line
    beginning-of-buffer
    !force search-forward "\t"
    !while $status
        set-variable #l1 $window-acol
        backward-delete-char
        &sub #l1 $window-acol insert-space
        !force search-forward "\t"
    !done
    goto-line #l0
    screen-update
    ml-write "[Converted tabs]"
!emacro
```

Both of these commands are available from the command line, they are indistinguishable from the built in commands.

Macros may also be nested, as shown in the next example, this macro contains a **define-macro** within itself, when executed the macro creates another macro dynamically - dynamic macros are generally given a prefix of **%** and are highlighted differently in **describe-bindings**(2).
The following example is taken from the **alarm**(3) macro, executing **alarm** the user is prompted for a message, and the time interval before the alarm expires in hours and minutes. It then creates a new macro with a callback so that the new macro will be called at the correct time.

```
!if &seq %alarm-numb "ERROR"
    set-variable %alarm-numb 0
    set-variable %osd-alarm &pinc %osd 1
!endif

define-macro alarm
    set-variable %alarm-numb &add %alarm-numb 1
    set-variable #l0 &cat "%alarm-" %alarm-numb
    !force set-variable #l2 @3
    !if &not $status
        set-variable &ind #l0 @ml "Message"
        set-variable #l1 @ml "Hours"
        set-variable #l2 @ml "Minutes"
    !else
        set-variable &ind #l0 @1
        set-variable #l1 @2
```

```
    !endif
    set-variable #l1 &mul 60000 &add &mul 60 #l1 #l2
    define-macro #l0
        !bell
        set-variable #l0 &add &len &ind @0 10
        osd %osd-alarm 0 "bat" 9 3
        osd %osd-alarm 1 ""
        osd %osd-alarm 2 "c" "ALARM"
        osd %osd-alarm 3 ""
        osd %osd-alarm 4 "" &ind @0
        osd %osd-alarm 5 ""
        osd %osd-alarm 6 "Bcf" " OK " f void
        %osd-alarm osd
    !emacro
    #l1 create-callback #l0
!emacro
```

## SEE ALSO

Refer to **!return**(4) and **!abort**(4) for details macro termination.

**!emacro**(4), **major-mode**(2), **define-macro-file**(2), **define-help**(2), **describe-bindings**(2), **execute-file**(2), **execute-named-command**(2), **global-bind-key**(2), **insert-macro**(2), **start-kbd-macro**(2).

## NAME

define-macro-file - Define macro file location

## SYNOPSIS

**define-macro-file** "*file-name*" ["*macro-name*" "*macro2-name*" ...]

## DESCRIPTION

Macros are loaded as late as possible using an on-demand mechanism, this speeds up the load time of MicroEmacs, it also keeps the startup file clean since macros are not defined within the start-up file. Only when the user first executes a macro defined via **define-macro-file** is the file loaded, the macro becomes defined and is executed. Subsequent calls to the macro will not reload the file as the macro will now be fully defined.

**define-macro-file** binds macros (*macro-name* ...) to a file name (*file-name*). This operation informs MicroEmacs which file should be loaded when *macro-name* is first executed. The *macro-name* arguments may be omitted if the file contains only one exported macro which has the same name as *file-name*.

Alternatively the macro file may contain many macros all of which can be defined by a single call to **define-macro-file**, listing all macros on the same line after the *file-name*. If a *macro-name* is given then the default macro *file-name* is not created, if a macro of that name does exist it must be added to the *macro-name* list.

## EXAMPLE

The following definitions are found in the `me.emf` start-up file:

```
0 define-macro-file utils ascii-time
define-macro-file format clean sort-lines-ignore-case tabs-to-spaces ...
define-macro-file cvs cvs cvs-status cvs-update cvs-commit cvs-log-file ...
define-macro-file abbrev expand-abbrev-handle expand-iso-accents ...
define-macro-file misc symbol check-line-length alarm time
define-macro-file search replace-all-string query-replace-all-string
define-macro-file tools compile grep rgrep which diff diff-changes
define-macro-file hkdirlst file-browser file-browser-close
define-macro-file comment comment-line uncomment-line comment-to-end-of-line
define-macro-file spell spell-word spell-buffer spell-edit-word find-word
define-macro-file games Metris Patience Triangle Mahjongg Match-It
define-macro-file fattrib file-attrib
define-macro-file osd osd-main
define-macro-file gdiff
define-macro-file calc
define-macro-file draw
```

Hilighting a number of entries as examples; macro file **calc** is defined with no macro definition, the macro is assumed to be **calc**. The file **tools.emf** contains multiple macros **compile**, **grep**, **diff** and **diff-changes**; all can be defined by a single **define-macro-file** entry.

## NOTES

- Macro files are searched for in the current directory and along the **$search-path**(5).
- The macro file is not loaded unless a binding has been defined using **define-macro-file**.
- Any other macros that exist in the *file-name* macro file become defined when the

entry point macro is loaded and are available for use. This is potentially useful as a single *entry* macro may be defined using **define-macro-file**, when invoked other helper macros become available.

## SEE ALSO

**major-mode**(2), **define-macro**(2), **$search-path**(5), **start-up**(3).

Ref: define-macro-file(2) File: m2cmd147.2 Date: 2026/07/25

## NAME

delete-blank-lines - Delete blank lines about cursor

## SYNOPSIS

**delete-blank-lines** (**C-x C-o**)

## DESCRIPTION

**delete-blank-lines** deletes all the blank lines before and after the current cursor position. Note that the deleted lines are not added to a kill buffer.

## SEE ALSO

**delete-indentation**(3), **clean**(3), **kill-line**(2).

Ref: delete-blank-lines(2) File: m2cmd030.2 Date: 2022/06/12

## NAME

delete-buffer - Delete a buffer

## SYNOPSIS

*n* **delete-buffer** "*buffer-name*" (**C-x k**)

## DESCRIPTION

**delete-buffer** disposes of buffer *buffer-name* in the editor and reclaim the memory. This does not delete the file that the buffer was read from.

If the buffer has been edited and its name does not start with a '*' then the user is prompted as to whether the changes should be discarded. Also if the buffer has an active process running in it then confirmation is sort from the user before the process is killed.

The argument *n* can be used to change the default behavior of delete-buffer described above, *n* is a bit based flag where:

**0x01** Enables loss of work checks (default). These include a check that the buffer has not been modified, if so the user is prompted. Also if a process is running then user must confirm that the process can be killed. If this flag is not supplied then the buffer is killed without any user prompts (useful in macros).

## SEE ALSO

**next-buffer**(2).

Ref: delete-buffer(2) File: m2cmd031.2 Date: 2022/06/12

## NAME

delete-dictionary - Remove a spelling dictionary from memory

## SYNOPSIS

*n* **delete-dictionary** ["*dictionary*"]

## DESCRIPTION

**delete-dictionary** removes the given *dictionary* from memory, where *n* is a bitwise flag determining the removal mode, defined as follows:

**0x01** Prompt the user before loosing any changes (except to the ignore dictionary).

**0x02** Delete all the dictionaries other than the ignore dictionary.

**0x04** Delete the ignore dictionary.

If the argument does not have bit 0x02 or 0x04 set, which specify the dictionaries to be deleted, the user is prompted for the "*dictionary*". The default argument is 1.

## NOTES

The ignore dictionary is a temporary dictionary that exists in memory for duration of the MicroEmacs session; the dictionary holds words that have been ignored during any previous spell checks (see **spell**(2)). All of the words that have been ignored may be discarded with:

```
4 delete-dictionary
```

i.e. **esc 4 esc x delete-dictionary**.

## SEE ALSO

**spell-buffer**(3), **add-dictionary**(2), **save-dictionary**(2), **spell**(2).

Ref: delete-dictionary(2) File: m2cmd041.2 Date: 2022/06/12

## NAME

delete-frame - Delete the current frame

## SYNOPSIS

*n* **delete-frame** (**C-x 5 0**)

## DESCRIPTION

**delete-frame** deletes the current frame and the next is displayed. If there is no next frame then the command fails.

## SEE ALSO

**create-frame**(2), **next-frame**(2), **$frame-id**(5).

Ref: delete-frame(2) File: m2cmd162.2 Date: 2026/06/19

## NAME

delete-registry - Delete a registry tree

## SYNOPSIS

**delete-registry** "*root*"

## DESCRIPTION

**delete-registry** deletes a registry node *root* from the registry, any children belonging to the node are also deleted.

## DIAGNOSTICS

**delete-registry** fails if *root* does not exist.

## SEE ALSO

**copy-registry**(3), **delete-registry**(2), **find-registry**(2), **list-registry**(2), **mark-registry**(2), **read-registry**(2), **set-registry**(2), **&reg**(4), **erf**(8).

Ref: delete-registry(2) File: m2cmd114.2 Date: 2022/06/12

## NAME

delete-some-buffers - Delete buffers with query

## SYNOPSIS

*n* **delete-some-buffers**

## DESCRIPTION

**delete-some-buffers** cycles through all visible buffers (buffers without mode **hide**(2m) set) and prompts the user [**y/n**] as to whether the buffer should be deleted. A **y** response deletes the buffer, a **n** response retains the buffer.

If a **y** response is given, the buffer has been edited, and its name does not start with a '*' then the user is prompted as to whether the changes should be discarded. Also if the buffer has an active process running in it then confirmation is sort from the user before the process is killed.

The argument *n* can be used to change the default behavior of delete-some-buffers described above, *n* is a bit based flag where:

**0x01** Enables all checks (default). These include the initial y/n prompt on each buffer, the buffer has not been modified check, if so the user is prompted. Also if a process is running then user must confirm that the process can be killed. If this flag is not supplied then all visible buffers are killed without any user prompts (useful in macros).

## SEE ALSO

**delete-buffer**(2), **next-buffer**(2), **hide**(2m).

Ref: delete-some-buffers(2) File: m2cmd155.2 Date: 2022/06/12

## NAME

delete-window - Delete the current window
delete-other-windows - Delete other windows

## SYNOPSIS

*n* **delete-window** (**C-x 0**)
*n* **delete-other-windows** (**C-x 1**)

## DESCRIPTION

**delete-window** attempts to delete the current window (remove window from the screen), retrieving the lines for use in the window adjacent to it. The command fails if there is no other window or if the current window is protected from deletion (see **$window-flags**(5)). The deletion protection can be overridden by giving the command a numerical argument *n* of 2.

The window deletion policy is determined by the formation of the windows displayed on the screen. The bias is for the *previous* window (above) the current window to be merged when split vertically, and for the left window to be merged when split horizontally.

**delete-other-windows** deletes all of the other windows, the current window becomes the only window, using the entire available screen area. Windows can be protected from deletion by using the $window-flags variable, giving the command a numerical argument *n* of 2 overrides this protection.

## SEE ALSO

**set-position**(2), **grow-window-vertically**(2), **resize-window-vertically**(2), **split-window-horizontally**(2), **split-window-vertically**(2), **$window-flags**(5).

Ref: delete-window(2) File: m2cmd036.2 Date: 2022/06/12

## NAME

describe-bindings - Show current command/key binding

## SYNOPSIS

**describe-bindings** (**C-h b**)

## DESCRIPTION

**describe-bindings** pops up a window with a list of all the named commands, and the keys currently bound to them. Each entry is formatted as:

**keyCode . . . . . command**

**describe-bindings** is buffer context sensitive and shows the bindings for the currently active buffer (i.e. the buffer that is active when the command is invoked). The resultant command list is divided into three sections as follows:

**Buffer Bindings**
: The bindings for the active buffer when **describe-bindings** was invoked. These are the buffer bindings set by **buffer-bind-key**(2).

**Ml Bindings**
: The message line bindings as set by **ml-bind-key**(2).

**Global Bindings**
: Global binding of keys as set by **global-bind-key**(2).

## EXAMPLE

The following is an example of the displayed output from **describe-bindings**. This was invoked while editing buffer **m2fun038.2** which is the **Nroff** file for this manual page; the local bindings for the buffer are all Nroff related.

```
Buffer [m2cmd038.2] bindings:

    "C-c C-s" ..................... nroff-size
    "C-c C-r" ..................... nroff-roman
    "C-c C-b" ..................... nroff-bold
    "C-c C-i" ..................... nroff-italic
    "C-c C-c" ..................... nroff-mono
    "C-c C-o" ..................... nroff-para
    "esc o" ....................... nroff-para
    "esc q" ....................... nroff-para
    "C-c b" ....................... nroff-bold-block
    "C-c i" ....................... nroff-italic-block
    "C-c C-h" ..................... nroff-swap-hilight
    "C-c &" ....................... nroff-add-padding
    "C-x &" ....................... nroff-remove-padding
    "C-c C-p" ..................... nroff-prev
    "C-mouse-drop-1" .............. nroff-tag

Ml bindings:

    "esc esc" ..................... tab

Global bindings:

    "C-a" ......................... beginning-of-line
    "C-b" ......................... backward-char
```

```
"C-c" ........................ 4 prefix
"C-d" ........................ forward-delete-char
"C-e" ........................ end-of-line
"C-f" ........................ forward-char
"C-g" ........................ abort-command
"C-h" ........................ 3 prefix
"C-i" ........................ insert-tab
"C-k" ........................ kill-line
"C-l" ........................ recenter
"C-m" ........................ newline
"C-n" ........................ forward-line
"C-o" ........................ insert-newline
"C-p" ........................ backward-line
"C-q" ........................ quote-char
"C-r" ........................ isearch-backward
"C-s" ........................ isearch-forward
"C-t" ........................ transpose-chars
"C-u" ........................ universal-argument
"C-v" ........................ scroll-down
"C-w" ........................ kill-region
"C-x" ........................ 2 prefix
"C-y" ........................ yank
"C-z" ........................ scroll-up
"C-_" ........................ undo
"A-e" ........................ file-browser
"A-r" ........................ replace-all-string
"esc C-c" ..................... count-words
"esc C-f" ..................... goto-matching-fence
"esc C-g" ..................... abort-command
"esc C-i" ..................... goto-matching-fence
"esc C-k" ..................... global-unbind-key
"esc C-n" ..................... change-buffer-name
"esc C-r" ..................... query-replace-string
"esc C-v" ..................... scroll-next-window-down
"esc C-w" ..................... kill-paragraph
"esc C-z" ..................... scroll-next-window-up
"esc space" ................... set-mark
"esc !" ....................... pipe-shell-command
"esc $" ....................... spell-word
"esc ." ....................... set-mark
"esc /" ....................... execute-file
"esc <" ....................... beginning-of-buffer
"esc >" ....................... end-of-buffer
"esc ?" ....................... help
"esc @" ....................... pipe-shell-command
"esc [" ....................... backward-paragraph
"esc \\" ...................... ipipe-shell-command
"esc ]" ....................... forward-paragraph
"esc ^" ....................... delete-indentation
"esc b" ....................... backward-word
"esc c" ....................... compile
"esc d" ....................... forward-kill-word
"esc e" ....................... set-encryption-key
"esc f" ....................... forward-word
"esc g" ....................... goto-line
"esc i" ....................... tab
"esc k" ....................... global-bind-key
"esc l" ....................... lower-case-word
"esc m" ....................... global-mode
"esc n" ....................... forward-paragraph
"esc o" ....................... fill-paragraph
"esc p" ....................... backward-paragraph
"esc q" ....................... fill-paragraph
"esc r" ....................... replace-string
"esc t" ....................... find-tag
"esc u" ....................... upper-case-word
"esc v" ....................... scroll-up
"esc w" ....................... copy-region
"esc x" ....................... execute-named-command
"esc y" ....................... reyank
```

```
"esc z" ..................... quick-exit
"esc ~" ..................... -30 buffer-mode
"esc A-r" ................... query-replace-all-string
"C-x C-a" ................... set-alpha-mark
"C-x C-b" ................... list-buffers
"C-x C-c" ................... save-buffers-exit-emacs
"C-x C-e" ................... execute-kbd-macro
"C-x C-f" ................... find-file
"C-x C-g" ................... abort-command
"C-x C-h" ................... hunt-backward
"C-x C-i" ................... insert-file
"C-x C-l" ................... lower-case-region
"C-x C-o" ................... delete-blank-lines
"C-x C-q" ................... rcs-file
"C-x C-r" ................... read-file
"C-x C-s" ................... save-buffer
"C-x C-t" ................... transpose-lines
"C-x C-u" ................... upper-case-region
"C-x C-v" ................... view-file
"C-x C-w" ................... write-buffer
"C-x C-x" ................... exchange-point-and-mark
"C-x C-y" ................... insert-file-name
"C-x C-z" ................... shrink-window-vertically
"C-x #" ..................... filter-buffer
"C-x (" ..................... start-kbd-macro
"C-x )" ..................... end-kbd-macro
"C-x /" ..................... isearch-forward
"C-x 0" ..................... delete-window
"C-x 1" ..................... delete-other-windows
"C-x 2" ..................... split-window-vertically
"C-x 3" ..................... next-window-find-buffer
"C-x 4" ..................... next-window-find-file
"C-x 5" ..................... split-window-horizontally
"C-x 9" ..................... find-bfile
"C-x <" ..................... scroll-left
"C-x =" ..................... buffer-info
"C-x >" ..................... scroll-right
"C-x ?" ..................... describe-key
"C-x @" ..................... pipe-shell-command
"C-x [" ..................... scroll-up
"C-x ]" ..................... scroll-down
"C-x ^" ..................... grow-window-vertically
"C-x `" ..................... get-next-line
"C-x a" ..................... goto-alpha-mark
"C-x b" ..................... find-buffer
"C-x c" ..................... shell
"C-x e" ..................... execute-kbd-macro
"C-x h" ..................... hunt-forward
"C-x k" ..................... delete-buffer
"C-x m" ..................... buffer-mode
"C-x n" ..................... change-file-name
"C-x o" ..................... next-window
"C-x p" ..................... previous-window
"C-x q" ..................... kbd-macro-query
"C-x r" ..................... search-backward
"C-x s" ..................... search-forward
"C-x u" ..................... undo
"C-x v" ..................... set-variable
"C-x w" ..................... resize-window-vertically
"C-x x" ..................... next-buffer
"C-x z" ..................... grow-window-vertically
"C-x {" ..................... shrink-window-horizontally
"C-x }" ..................... grow-window-horizontally
"C-h C-c" ................... help-command
"C-h C-i" ................... help-item
"C-h C-v" ................... help-variable
"C-h a" ..................... command-apropos
"C-h b" ..................... describe-bindings
"C-h c" ..................... list-commands
"C-h d" ..................... describe-variable
```

```
"C-h k" ..................... describe-key
"C-h v" ..................... list-variables
"backspace" ................. backward-delete-char
"delete" .................... forward-delete-char
"down" ...................... forward-line
"end" ....................... end-of-buffer
"esc" ....................... 1 prefix
"f1" ........................ menu
"home" ...................... beginning-of-buffer
"insert" .................... 141 buffer-mode
"left" ...................... backward-char
"mouse-drop-1" .............. mouse-drop-left
"mouse-drop-2" .............. yank
"mouse-drop-3" .............. menu
"mouse-pick-1" .............. mouse-pick-left
"mouse-pick-2" .............. void
"mouse-pick-3" .............. void
"page-down" ................. scroll-down
"page-up" ................... scroll-up
"redraw" .................... screen-update
"return" .................... newline
"right" ..................... forward-char
"tab" ....................... tab
"up" ........................ backward-line
"S-backspace" ............... backward-delete-char
"S-delete" .................. forward-delete-char
"S-tab" ..................... backward-delete-tab
"C-down" .................... 5 forward-line
"C-left" .................... backward-word
"C-mouse-drop-1" ............ mouse-control-drop-left
"C-mouse-pick-1" ............ set-cursor-to-mouse
"C-page-down" ............... scroll-next-window-down
"C-page-up" ................. scroll-next-window-up
"C-right" ................... forward-word
"C-up" ...................... 5 backward-line
"A-down" .................... 1 scroll-down
"A-left" .................... 1 scroll-left
"A-right" ................... 1 scroll-right
"A-up" ...................... 1 scroll-up
"esc backspace" ............. backward-kill-word
"esc esc" ................... expand-abbrev
"C-c g" ..................... grep
```

Note that both internal commands and macro commands are shown in the list.

## SEE ALSO

**buffer-bind-key**(2), **command-apropos**(2), **describe-key**(2), **describe-variable**(2), **global-bind-key**(2), **list-commands**(2), **ml-bind-key**(2).

Ref: describe-bindings(2) File: m2cmd038.2 Date: 2022/06/12

## NAME

describe-key - Report keyboard key name and binding

## SYNOPSIS

*n* **describe-key** "*key*" (**C-x ?**)

## DESCRIPTION

**describe-key** allows a key to be typed and it will report the name of the command bound to that key (if any) and the internal key-code. This command is useful when trying to locate the identity of keyboard keys for binding. The **$result**(5) variable is set to the bound command or an empty string if the key is not bound.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When this bit is not set all messages to the message-line are suppressed enabling it to be used in macro code as a way of determining if a key is currently bound.

**0x02** If this bit is set then message-line bindings are included in the lookup rather than buffer bindings.

**0x04** If this bit is set then current buffer bindings are not included in the key lookup.

## SEE ALSO

**command-apropos**(2), **global-bind-key**(2), **buffer-bind-key**(2), **ml-bind-key**(2), **describe-bindings**(2), **describe-variable**(2), **$result**(5).

Ref: describe-key(2) File: m2cmd039.2 Date: 2023/06/29

## NAME

describe-variable - Describe current setting of a variable

## SYNOPSIS

**describe-variable** (**C-h v**)

## DESCRIPTION

**describe-variable** describes the current setting of the given variable (**%**, **:** and **$** variables), returning `ERROR` if the variable is undefined. If a `$` variable is not found then it is tested for an environment variable, i.e.

```
describe-variable $PATH
```

returns your environment `$PATH` setting. This is the easiest and best way of determining the current platform from within a Macro file.

The returned value of any undefined variable is the string `ERROR`.

## NOTES

Completion is enabled on the command line for variable names.

## SEE ALSO

**describe-key**(2), **help-variable**(2), **set-variable**(2).

Ref: describe-variable(2) File: m2cmd040.2 Date: 2022/06/12

## NAME

directory-tree - Draw the file directory tree

## SYNOPSIS

*n* **directory-tree** ["*directory*"]

## DESCRIPTION

**directory-tree** creates or manipulates a view of the file systems directory structure. The command is quite complex to use directly so is largely used by macros such as **file-browser**(3).

The argument *n* is a bit based flag which is used to control the command, where the bits have the following meaning:

**0x01** If set, the focal directory of the command is set by the given "*directory*" argument. Otherwise the argument is not required and the command must be executed within the "**directory**" buffer; the current line sets the focal directory.

**0x02** Specifies that the current line in resultant "**directory**" window should be set to the focal directory. If this bit is not set then the current line will be the last selected directory, or if none have been selected, the first line in the buffer.

**0x04** Specifies that any evaluations required during the commands operation should be performed. Without this flag an open operation on a directory which has not previously been evaluated will not be perform an evaluation and the results will likely be incomplete.

**0x08** Specifies that the current focal directory should be opened. This means that sub-directories within the current focal directory will also be drawn in the directory tree.

**0x10** Specifies that the current focal directory should be closed. This means that sub-directories within the current focal directory will not be drawn in the directory tree.

**0x20** Specifies that the current focal directory's open state should be toggled. This means that if the sub-directories are currently hidden they will now be drawn and vice-versa.

**0x40** When specified any directory opened will be re-evaluated, ensuring the accuracy of the information.

**0x80** Enables a recursive behavior, for example if this flag was specified with the open then not only will the focal directory be opened, but all of it's children, and their children etc. Note that if the Evaluation flag is not specified then only the already evaluated directories can be opened.

**0x100** & **0x200**
If both bits **0x100** and **0x200** are set then the current filter mode is toggled; if only bit **0x100** is set then the filter mode is disabled (not filtered), if only bit **0x200** is set then the filter mode is enabled, and if neither is set then the filter mode is left unchanged.

directory-tree creates a new buffer "**directory**" and draws the known directory tree. Every drawn directory is preceded by a character flag giving the user an indication of the directory state, where:

**?** Directory has not been evaluated.

**-** Directory has been evaluated and is visible.

**+** Directory has been evaluated but is currently hidden.

Directories which have been evaluated and found to have no children use the '-' **$box-chars**(5) instead of a '-' character.

On UNIX platforms, if a directory is a symbolic link to another directory, the link name is given after the directory name.

If the filter mode is enabled, default, then only directories which do not match a **file-ignore**(5) postfix are listed. All non-hidden directories are listed when the filter mode is disabled.

## EXAMPLE

The best example of the use of directory-tree is **file-browser**(3) which can be found in hkdirlst.emf.

## SEE ALSO

**file-browser**(3), **$file-ignore**(5), **$box-chars**(5).

Ref: directory-tree(2) File: m2cmd032.2 Date: 2023/06/29

## NAME

exchange-point-and-mark - Exchange the cursor and marked position

## SYNOPSIS

**exchange-point-and-mark** (**C-x C-x**)

## DESCRIPTION

**exchange-point-and-mark** moves the cursor to the current marked position (see **set-mark**(2)) in the current window and moves the mark to where the cursor was. This is very useful in finding where a mark was, or in returning to a position previously marked.

The current buffer mark line and column position may be obtained from within a macro using the variables **$window-mark-line**(5) and **$window-mark-col**(5).

## SEE ALSO

**set-mark**(2), **copy-region**(2), **$window-mark-col**(5), **$window-mark-line**(5).

Ref: exchange-point-and-mark(2) File: m2cmd043.2 Date: 2022/06/12

## NAME

execute-buffer - Execute script lines from a buffer
execute-line - Execute a script line from the command line

## SYNOPSIS

*n* **execute-buffer** "*buffer-name*"
*n* **execute-line** ["*command-line*"]

## DESCRIPTION

**execute-buffer** executes script lines in the named buffer *buffer-name*. If the buffer is off screen and an error occurs during execution, the cursor is left on the line causing the error.

If numeric argument *n* is given to **execute-buffer** then macro argument **@?**(4) will be set to 1 and **@#**(4) to *n*, otherwise **@?** and **@#** will be set to 0 and 1 respectively. These variables can be used by the macro code during the execution of the buffer.
**execute-line** executes a in script line entered from the command line, typically this command is used in macros. If bit 2 of the given numeric argument **n** is set then the command returns an error if there is more text given in the command-line than is executed, for example

```
3 execute-line "set-variable #g0 1 2"
```

Will throw an error as `" 2"` is not used, this can be useful in tracking down macro errors. Note the **@?** and **@#** macro argument variables are also set.

## SEE ALSO

**execute-file**(2), **execute-string**(2), **execute-named-command**(2).

Ref: execute-buffer(2) File: m2cmd044.2 Date: 2022/06/12

## NAME

execute-file - Execute script lines from a file

## SYNOPSIS

*n* **execute-file** "*file*" (**esc /**)

## DESCRIPTION

**execute-file** executes script lines from the given *file* in succession, this is the normal way to execute a MicroEmacs script. The command prompts for a file name, and will then search for <*file*>[.emf] in the search path. If the file is found then the file is loaded and the buffer is executed *n* times.

If numeric argument *n* is given then macro argument variable **@?**(4) will be set to 1 and @#(4) to *n*, otherwise **@?** and **@#** will be set to 0 and 1 respectively. These variables can be used by the macro code during the execution of the file.

## SEE ALSO

**execute-buffer**(2), **execute-line**(2), **execute-named-command**(2), **execute-string**(2), **@?**(4), @#(4).

Ref: execute-file(2) File: m2cmd045.2 Date: 2025/08/28

## NAME

execute-kbd-macro - Execute a keyboard macro

## SYNOPSIS

*n* **execute-kbd-macro** (**C-x e**)

## DESCRIPTION

**execute-kbd-macro** executes a keyboard macro. The entire sequence of recorded key-strokes is repeated starting at the current point. The result is exactly as if you were retyping the sequence all over again. A numeric argument *n* prefixing the **execute-kbd-macro** command repeats the stored key-strokes *n* times.

Keyboard macros are recored with **start-kbd-macro**(2); recording is terminated with **end-kbd-macro**(2).

## SEE ALSO

**end-kbd-macro**(2), **kbd-macro-query**(2), **name-kbd-macro**(2), **start-kbd-macro**(2).

Ref: execute-kbd-macro(2) File: m2cmd046.2 Date: 2022/06/12

## NAME

execute-named-command - Execute a named command

## SYNOPSIS

*n* **execute-named-command** "*command-string*" esc x

## DESCRIPTION

**execute-named-command** command prompts the user for the name of a command to execute and then executes the command *n* times. MicroEmacs offers command completion and history facilities, see **ml-bind-key**(2).

## SEE ALSO

**execute-buffer**(2), **describe-bindings**(2), **ml-bind-key**(2).

Ref: execute-named-command(2) File: m2cmd047.2 Date: 2025/08/28

## NAME

execute-string - Execute a string as a command

## SYNOPSIS

*n* **execute-string** "*string*"

## DESCRIPTION

**execute-string** executes the given *string n* times as if it is being typed. This is the writable format of a keyboard macro, it can be placed in any **emf** file. Any characters may form the *string* (unprintables as \xXX) and key-strokes that are bound to a command will execute that command. This command is used by macros to store user defined keyboard macros.

## EXAMPLE

The following example uses keyboard strokes with **execute-string** in a macro to format **nroff**(1) text located between `.` commands:

```
define-macro nroff-para
    beginning-of-line
    !if &not &sequal @wc "."
        1 buffer-mode "magic"
        execute-string "\CXS^\\.\CM\CB\CM\CX\CH\CN\CM"
        -1 fill-paragraph
        execute-string "\CD\CX\CH\CN\CD\CXH\CB"
    !endif
    forward-line
!emacro
```

**execute-string** has the advantage that execution is very fast as the amount of parsing and decoding to be performed is limited. The disadvantage is that you cannot quickly discern which operations are being performed !!

## NOTES

Try to avoid using named key, such as "`up`" and "`return`", as the keyboard macro equivalent is not readable and is likely to change in future releases.

For this reason the following special abbreviations may be used

**\E** The "**escape**" key.

**\N** The "**return**" key.

**\T** The "**tab**" key.

**\b** The backspace character (0x08).

**\d** The delete character (0x7f).

**\e** The escape character (0x1b).

**\f** The form-feed character (0x0c).

**\n** The carriage-return character (0x0a).

**\r** The line-feed character (0x0d).

## SEE ALSO

**buffer-abbrev-file**(2), **global-abbrev-file**(2), **insert-macro**(2), **name-kbd-macro**(2), **start-kbd-macro**(2).

Ref: execute-string(2) File: m2cmd048.2 Date: 2022/06/12

## NAME

exit-emacs - Exit MicroEmacs

## SYNOPSIS

*n* **exit-emacs** [*exitCode*]

## DESCRIPTION

Exit MicroEmacs back to the operating system. If no argument *n* is given and there are any unwritten, changed buffers, the editor prompts the user to discard changes.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

`0x01` Prompting the user before saving any files, writes out all changed buffers to the files they were read from. Saves all changed dictionaries, killing any running commands and exits the editor.

`0x02` Writes out all changed buffers to the files they were read from, saves all changed dictionaries, killing any running commands and exits the editor. The behavior is modified by flags `8` and `16`.

`0x04` Define the exit code *exitCode* from MicroEmacs, the default when omitted is 0.

`0x08` Discard changed buffers (except the history and registry) and delete any backup files associated with the edited buffer(s). This mode does not operate when bit `1` is set.

`0x10` Discard changed buffers (except the history and registry) and create a auto save file(s) with the changes from unsaved buffers. This mode does not operate when bit `1` is set.

`0x20` Forces the program exit status to be 1 (non-zero), which is interpreted by any parent task as an error status.

If an argument of `0` is specified then MicroEmacs exits immediately.

**exit-emacs** may be used to reproduce the behavior of **save-buffers-exit-emacs**(2) and **quick-exit**(2) as follows:

```
3 exit-emacs == save-buffers-exit-emacs
2 exit-emacs == quick-exit
```

## MACRO INTERACTION

If the macro **shut-down**(3) is defined then it is invoked by the editor when it is closed. A user defined shutdown action may be associated with the close operation. It is the callers responsibility that the macro terminates correctly otherwise the editor will hang.

## NOTES

All buffers with a name starting with a '$\star$' are assumed to be system buffers (i.e. ***scratch***) and are not saved.

Modes `8` and `16` may be used where the user wishes to discard edited buffers on exiting and does not want to confirm the exit. Dictionaries should be configured to auto-save in **user-setup**(3).

An emergency exit command which forces the editor to quit may be implemented as `0x28 exit-emacs` i.e. Force quit and exit with failed status. Internally this is what MicroEmacs uses to perform an emergency quit operation caused by signal fault etc.

## SEE ALSO

**quick-exit**(2), **save-buffers-exit-emacs**(2), **shut-down**(3).

Ref: exit-emacs(2) File: m2cmd049.2 Date: 2026/08/20

## NAME

expand-abbrev - Expand an abbreviation

## SYNOPSIS

**expand-abbrev**

## DESCRIPTION

**expand-abbrev** expands an abbreviation to an alternate form. The abbreviation must be an alpha-numeric string and the cursor must be one position to the right of the abbreviation (which must not be alpha-numeric) when this command is called. If the abbreviation is found, it is deleted and the alternate form is inserted leaving the cursor at the end of the insertion unless `\p` is used. If not found, a space is inserted.

## SEE ALSO

**buffer-abbrev-file**(2), **global-abbrev-file**(2), **expand-abbrev-handle**(3), **eaf**(8).

Ref: expand-abbrev(2) File: m2cmd051.2 Date: 2022/06/12

## NAME

file-op - File system operations command

## SYNOPSIS

*n* **file-op** [ ( [ "*from-file*" "*to-file*" ] ) —

( ["*delete-file*"] ) — ( ["*dir-name*"] ) —
( [ "*chmod-file*" "*mode*" ] ) ]

## DESCRIPTION

**file-op** can be used to perform numerous file system operations. The given argument *n* must be used to determine the required operation, the value is a bit based flag denoting the operation as follows:

**0x010** Log-off and close down the current ftp connect (not a file system operation but functionality was required and it had to go somewhere).

**0x020** When this bit is set the command functionality is changed to delete-file, the single argument *delete-file* is deleted. Note this can also delete a directory if the directory is empty.

**0x040** When this bit is set the command functionality is changed to move-file, the specified *from-file* is moved to *to-file*.

**0x080** When this bit is set the command functionality is changed to copy-file, the specified *from-file* is copied to *to-file*.

**0x100** When this bit is set the command functionality is changed to making a new directory, the specified *dir-name* is the name of the new directory. A file or directory of the given name must not already exist.

**0x200** When this bit is set the command functionality is changed to a change mode type of operation. *chmod-file* is the name of the file (or directory) to change and *mode* is the integer mode to apply. Note that the *mode* value is operating system specific. The file attributes may be retrieved with the **&stat**(4) command.

Only one operation can be performed per invocation. The following bits in the given argument *n* can be used to effect the behaviour of these operations:

**0x01** Enables validity checks, these include a check that the proposed file does not already exist, if so confirmation of writing is requested from the user. Also MicroEmacs checks all other current buffers for one with the proposed file name, if found, again confirmation is requested. Without this flag the command will always succeed wherever possible.

**0x02** Creates a backup of any file about to be deleted or over-written. Set help on **$buffer-backup**(5) for backup file-name generation.

**0x04** When validity checks are enabled (see bit 0x01 above) and this bit is set, if a problem is found the command aborts rather than prompting the user. Also see the error return codes in the notes section below.

**0x80** When this bit is set then the files time stamp is preserved.

## NOTES

**http** and **https** files are not supported except as the source file when copying. **ftp**, **ftpe** and **ftps** files are fully supported, however note that if moving a file from and to the same ftp site M-S will attempt to use an ftp rename command, if this fails try moving the file locally and then move if back to the ftp site.

The command is used by **file-browser**(3) and **ftp**(3) which provides an easy to use interfaces for file manipulation.

When the command fails an error code is returned in the **$result**(5) variable, which will be one of the following:

**0** User aborted.

**1** General error.

**2** Failure due to a read error.

**3** Failure due to a write error.

**4** Failure due to a directory creation error.

**5** Failure due to a file or directory deletion error.

**6** Failure due to file write checks, i.e. file already exists etc.

**7** Failure due to directory creation checks, i.e. the directory or file already exists.

**8** Failure due to file delete checks, i.e. file is read only.

**9** Failure due to source and destination files are URLs.

## SEE ALSO

**file-browser**(3), **ftp**(3), **find-file**(2), **write-buffer**(2), **&stat**(4), **$result**(5), **$temp-name**(5).

Ref: file-op(2) File: m2cmd158.2 Date: 2023/02/21

## FILE HOOKS

**NOTE:** In January 2026 the **add-file-hook** command was replaced with a new **major-mode**(2) commad with the main objective of consolidating the major-mode concept. As a result, this page contains out of date information and needs updating.

File hooks provide a mechanism to automatically invoke a set of macros for a given buffer type when the following events occur:

- Loading of a file into a buffer
- Moving into a buffer (i.e. making a buffer current)
- Moving out of a buffer (i.e. making another buffer current)
- Deleting an active buffer

The file hook selection (see below) is performed on the file name / extension and on the textual content of the buffer using add-file-hook.

Refer to Language Templates for a description of how the file hooks are used to define a new template for a new text format.

The hook macros allow buffer modes and highlighting, applicable to the text type of the file, to be applied to the buffer. In addition, the associated hook macros may be located in a separate file and are loaded on demand when the file reading determines that a set of hook macros are required.
Consider a file hook definition of the form;

```
add-file-hook ".c .h" "fhook-c"
```

which binds the file hook **fhook-c** to any files that are loaded with the extension **.c** and **.h**. The operations undertaken by MicroEmacs are defined as follows when a file `foo.c` is loaded:

- Attempt to load file `foo.c`, if `foo.c` is not found then create a new buffer and assign file name `foo.c`.
- If `foo.c` is found then load file into buffer. Search the first line(s) of the buffer for magic hook text (*add-file-hook* with argument).
- If magic hook was not found then determine hook name from the file extension (*add-file-hook* information).
- If a hook command is located, assign the file hook **fhook-c** to the buffer, assign the buffer entry (begin) hook macro of **bhook-c**; assign a buffer exit hook of **ehook-c**.
- If the macro **fhook-c** is undefined then execute the macro file **hkc.emf** from the MicroEmacs home directory in an attempt to load the macro. If the file **myc.emf** is defined, then the modifications to the language template are applied after **hkc.emf** is loaded.
- If the macro **fhook-c** is (now) defined then `foo.c` is TEMPORARILY made the current buffer and the file hook macro **fhook-c** is executed to completion and the previous current buffer is restored. [*TEMPORARY* here implies that no buffer hooks are executed on the flip in/out of `foo.c`].

- The current buffer is officially swapped to `foo.c`. At this point the *ehook* of the old current buffer is executed (while its still current) and then `foo.c` is swapped in to become the current buffer; the begin buffer hook *bhook-cmode* is then executed for `foo.c` (if it exists).
- If the user moves to another buffer execute the end hook macro **ehook-cmode** (if it exists) and move to the new buffer, executing it's begin hook.
- If the user subsequently returns to buffer `foo.c` execute the previous buffers end hook macro, set the current buffer to *foo.c* and execute the begin hook macro **bhook-c** (if it exists).
- If the user kills buffer `foo.c`, if `foo.c` is the current buffer then an alternative buffer is made current, `ehook` and `bhook` executed as normal. If macro **dhook-c** is defined then `foo.c` is TEMPORARILY made the current buffer and the delete hook macro **dhook-c** is executed to completion and the previous current buffer is restored.

The name of the file hook macro name is important, hook commands must commence with the text **fhook-***mode* where *mode* is an identifier for the operating mode. The name space is decomposed as follows:

- The initial **f** is removed and replaced with **b** for the begin hook macro and **e** for the end hook macro.
- When the **fhook** macro is undefined the *mode* component is removed and the file **hk***mode***.emf** is executed from the MicroEmacs home directory in an attempt to define the macro.

The **fhook-** nomenclature may be omitted provided that the name is less than 6 characters, however the file, begin and end hook macros MUST commence with **f**, **b** and **e** respectively. In addition the macros must be defined as no auto file loading is performed.

### Buffer Hook Variables

The macros bound to a buffer may be interrogated, the variables **$buffer-fhook**(5), **$buffer-bhook**(5), **$buffer-ehook**(5) and **$buffer-dhook**(5) contain the names of any associated macro attached as a macro hooks, defining the *file*, *begin*, *end* and *delete* hooks respectively. If a macro is not bound then the empty string `""` is returned. Setting the variables has the effect of defining the hook and is a method by which the buffer hooks may be affected after the buffer has been loaded.

### Determination of a new file

The *file* hook **fhook-XXX** numeric argument may be used to determine if the file associated with a buffer is a new file created by the user, or an existing file. Typically this distinction is used to determine whether a boiler template is added to the file or not. The macro argument **@#** is defined as zero (0) if this is a new file that has been created, or non-zero otherwise.
The macro argument status is typically tested on entry to the macro as follows:

```
define-macro fhook-mode
    !if &not @#
        ; This is a new file. Do new file things
    !else
        ; This is an existing file
```

```
    !endif
    ; Set up bindings
!emacro
```

An example of a generic **hook** file is given at the end of this section which elaborates on the file hooks.

### Begin and End hooks

The *begin* and *end* hooks are usually used to save and restore global states which require special settings for a particular buffer type. This typically involves saving and restoring global variables which are used by other buffers in a different configuration. For example the following is used to reformat the time stamp string; the time stamp is a global variable **$timestamp**(5) and if it is changed in one buffer, it must be restored ready for another. In this case the old time stamp is retained in a local buffer variable whenever the buffer is entered, the time stamp is then modified for the buffers requirements. On exit from the buffer the old time stamp format is restored to it's former state.

```
0 define-macro bhook-foo
    set-variable .timestamp $timestamp      ; Save old time stamp.
    set-variable $timestamp "19%Y/%M/%D %h:%m:%s"
!emacro

0 define-macro ehook-foo
    set-variable $timestamp .bhook-foo.timestamp
!emacro
```

Note that in both cases the **define-macro**(2) invocation is defined as zero, this merely hides the macro from the command line since both are private macros not normally invoked by the user.

## FILE HOOK SELECTION

MicroEmacs may be reconfigured to operate in different modes (referred to a *Major Modes* in GNU **emacs**(1)) using the macro file hooks. The file hooks allow the working environment to be customized for the editing of text of a particular sort, by importing text specific macros, key rebinding and highlighting.

MicroEmacs, by default, loads a file into a buffer with default global modes with no highlighting. There are no mode specific key bindings, variable settings, macros or highlights, buffer interaction behaves in it's default state. The state of the buffer interaction may be modified through the use of the buffer modes (see Operating Modes), for example the 'C' programming language automatic indent **indent**(2) changes the characteristics of the `tab` character and performs language specific indentation of statements. When a text specific set of highlighting rules are applied to the buffer, the text becomes emphasized through the use of color applied selectively to the text i.e. comments, keywords, strings are shown in different colors, allowing them to be differentiated without studying the content.

Setting the operating mode of the buffer would be tedious to perform from the command line, instead MicroEmacs uses three different prioritized criteria to endeavor to select the correct operating mode. The operating mode is applied to the buffer by execution of a set of file specific macros, referred to a hook commands. The selection criteria of the hook commands is performed as follows, ordered in lowest to highest priority:

**File Name**

MicroEmacs uses the filename and/or the file extension to select a start-up hook command. File names and extensions are bound to a set of macro hooks in a space separated list e.g.

add-file-hook "**c cpp**" "**fhook-cmode**"
add-file-hook "**doc txt README**" "**fhook-doc**"

The space separated list of names are interpreted as either file extensions or filenames. In this case any file with the extension **.c**, **.cpp** is bound to a file hook called **fhook-cmode** e.g. `foo.c`. Similarly files with the extension **.doc** or **.txt** are interpreted as plain text documents and are bound to **fhook-doc**. e.g. `foo.txt`. The entry **README** that exists in the documentation hook list may refer to a file `README` and also `foo.README`, both cases invoke the document hook.

The file selection is the lowest priority selection criteria but usually satisfies most mode selection requirements.

**Magic Strings**

There are cases when file extensions may be omitted from files, typically these files include an identifier, or magic string, on the first line of the file which is used to identify the file to the operating system or application e.g. shell scripts under UNIX. MicroEmacs automatically interrogates the top of every file that is loaded to locate some form of identification string. The identification strings are defined in a similar way to the file name hooks, except instead of defining a file extension the location and text content of the identifier is defined:

1 add-file-hook "**#!/bin/sh**" "**fhook-shell**"
1 add-file-hook "**#!/usr/local/bin/wish**" "**fhook-tcl**"

In this case, any file that commences with "**#!/bin/sh**" is interpreted as a shell script and invokes the shell hook **fhook-shell**. Where the identifier does not appear on the first non-blank line, the argument may be increased to the number of lines to be searched. Also it the magic sting should be search for without **exact**(2m) mode then the argument should be negated, e.g.

-4 **add-file-hook** "<html>" "**fhook-html**"

invokes **fhook-html** whenever "`<html>`", "`<HTML>`" etc. is found in the first 4 lines of a file header, e.g.:

```
<!-- Comment line -->
<HtMl>
```

A match on a string identifier is assigned a higher priority than the file extension. It is recommended that magic strings are only used where there are no predefined file extensions, or conflicts exist between files with the same extension containing data interpreted in a different context.

**Explicit Strings**

The last method allows an explicit identifier string to be embedded into the text of the file informing MicroEmacs which mode it should adopt. GNU Emacs supports this (see **Major Mode** in the GNU Emacs documentation) type of operation by insertion of strings of the form:

**-*-** *mode* **-*-**

Where *mode* represents the major mode within GNU Emacs. The same format as used by **Magic Strings** can be used to find and extract the *mode*, e.g.:

**-1** add-file-hook "**-[*!]-[ \t]nroff.*-[*!]-**" "**fhook-nroff**"

The definition would detect the GNU Emacs mode defined in an Nroff file e.g.

```
.\" -*- nroff -*- "
.TH man 1
.SH NAME
...
```

It should be stressed that the **-*-** syntax belongs to GNU Emacs and NOT MicroEmacs, MicroEmacs provides a mechanism to locate, extract and interpret the string. The **-*-** syntax should only be applied to files if it is known that the *mode* is a GNU mode.

A MicroEmacs specific string is also provided, defined as:

**-!-** *mode* **-!-**

where *mode* is an arbitrary string defined by *add-file-hook*. User defined modes may be created and assigned to files with this syntax, this does not conflict with the GNU Emacs command. For example to assign a new mode *mymode* to a file we would define the following:

**-1** add-file-hook "**-!-[ \t]mymode.*-!-**" "**fhook-mymode**"

Files containing a the following identifier would be loaded with *mymode* hook:

```
# -!- mymode -!-
#
# Last Modified: <120683.1014>
```

## FILE HOOK SCRIPTS

The buffer hook files **hk***name***.emf** typically follow a standard layout, and are generally associated with hi-lighting as follows, **mode** in this case is the name of the file mode associated with the file:

```
!if &seq .hilight.mode "ERROR"
    set-variable .hilight.mode &pinc .hilight.next 1
!endif
;
; Define the hilighting
;
0 hilight .hilight.mode 1               $global-scheme
hilight .hilight.mode 2 "\*\*"          .scheme.comment
hilight .hilight.mode 4 "\"" "\"" "\\"  .scheme.string
hilight .hilight.mode 0 "'.'"           .scheme.quote

hilight .hilight.mode 1 "if"            .scheme.keyword
hilight .hilight.mode 1 "elif"          .scheme.keyword
hilight .hilight.mode 1 "else"          .scheme.keyword
...

; Reset the hilighting printer format and define the color bindings.
0 hilight-print .hilight.mode
hilight-print .hilight.mode "i"  .scheme.comment
hilight-print .hilight.mode "b"  .scheme.keyword .scheme.variable
hilight-print .hilight.mode "bi" .scheme.string .scheme.quote
```

```
...

; Define the indentation tokens
0 indent  .hilight.mode 2 10
indent .hilight.mode n "if" 4
indent .hilight.mode s "elif" -4
indent .hilight.mode s "else" -4
indent .hilight.mode o "endif" -4
indent .hilight.mode n "while" 4
...

define-macro fhook-mode
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "mode"
    !endif
    set-variable $buffer-hilight .hilight.mode
    set-variable $buffer-indent .hilight.mode
    1 buffer-mode "time"
    1 buffer-mode "indent"
    buffer-abbrev-file "mode"
!emacro
```

The previous example shows how the **fhook-mode** numeric argument is used to determine if this is a new file. If the argument **@#** is zero then this is interpreted as a new file, in this case a standard template is inserted (from file **mode.etf**) and the generic strings such as `$YEAR$` replaced with construction information. The template is generally used for standard headers and skeleton text body.

In addition an abbreviation file **mode.eaf** (see **eaf**(8)) is bound to the buffer using the **buffer-abbrev-file**(2) command and the buffer hi-lighting enabled by assignment of the **$buffer-hilight**(5) variable.

## MODIFYING FILE HOOKS

The standard hook files supplied with MicroEmacs should not be modified, changes to the file hooks may be applied using a separate macro file called **my***XXX.emf*, this is automatically executed after the **hk***XXX.emf* file is executed.

The extended hook functions may be defined company wide, or by the user, to over-ride some of the standard hook functions, or to extend the syntax of the base files with locally defined extensions. As an example, consider the following file **myc.emf** which extends the basic **hkc.emf** file set of hi-lighting tokens for the 'C' Language.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;
;  Created By    : Steven Phillips
;  Created       : Thu Jun 18 15:34:05 1998
;  Last Modified : <230798.0854>
;
;  Description   Extension hilighting for the 'C' language.
;
;  Notes         Define the locally defined 'C' library types and definitions
;                as extensions to the 'C' programming language.
;
;  History
;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; MicroEmacs specific tokens
hilight .hilight.c 1 "LINE"           .scheme.type
hilight .hilight.c 1 "BUFFER"         .scheme.type
hilight .hilight.c 1 "WINDOW"         .scheme.type
hilight .hilight.c 1 "REGION"         .scheme.type
hilight .hilight.c 1 "KEYTAB"         .scheme.type
hilight .hilight.c 1 "KILL"           .scheme.type
```

```
hilight .hilight.c 1 "KLIST"          .scheme.type
hilight .hilight.c 1 "HILNODE"        .scheme.type
hilight .hilight.c 1 "HILNODEPTR"     .scheme.type
hilight .hilight.c 1 "HILCOLOR"       .scheme.type
hilight .hilight.c 1 "SELHILIGHT"     .scheme.type
hilight .hilight.c 1 "VIDEO"          .scheme.type
hilight .hilight.c 1 "VVIDEO"         .scheme.type
hilight .hilight.c 1 "FRAMELINE"      .scheme.type
hilight .hilight.c 1 "IPIPEBUF"       .scheme.type
hilight .hilight.c 1 "DIRNODE"        .scheme.type
hilight .hilight.c 1 "UNDOND"         .scheme.type
hilight .hilight.c 1 "meVARLIST"      .scheme.type
hilight .hilight.c 1 "meVARIABLE"     .scheme.type
hilight .hilight.c 1 "meCMD"          .scheme.type
hilight .hilight.c 1 "meAMARK"        .scheme.type
hilight .hilight.c 1 "meABREV"        .scheme.type
hilight .hilight.c 1 "meMACRO"        .scheme.type
hilight .hilight.c 1 "meNARROW"       .scheme.type
hilight .hilight.c 1 "meREGISTERS"    .scheme.type
hilight .hilight.c 1 "meSTAT"         .scheme.type
hilight .hilight.c 1 "osdITEM"        .scheme.type
hilight .hilight.c 1 "osdDIALOG"      .scheme.type
hilight .hilight.c 1 "osdCHILD"       .scheme.type
hilight .hilight.c 1 "meSCROLLBAR"    .scheme.type
hilight .hilight.c 1 "osdCONTEXT"     .scheme.type
hilight .hilight.c 1 "osdDISPLAY"     .scheme.type
hilight .hilight.c 1 "RNODE"          .scheme.type
hilight .hilight.c 1 "REGHANDLE"      .scheme.type
hilight .hilight.c 1 "meDIRLIST"      .scheme.type
hilight .hilight.c 1 "meNAMESVAR"     .scheme.type
hilight .hilight.c 1 "meDICTADDR"     .scheme.type
hilight .hilight.c 1 "meSPELLRULE"    .scheme.type
hilight .hilight.c 1 "meDICTWORD"     .scheme.type
hilight .hilight.c 1 "meDICTIONARY"   .scheme.type
hilight .hilight.c 1 "meMODE"         .scheme.type
```

## SEE ALSO

Operating Modes, Language Templates, **add-file-hook**(2), **indent**(2).

## NAME

fill-paragraph - Format a paragraph

## SYNOPSIS

*n* **fill-paragraph** (**esc o**)

## DESCRIPTION

**fill-paragraph** this takes all the text in the current paragraph (as defined by surrounding blank lines, or a leading indent) and attempts to fill it from the left margin to the current fill column as defined by **$buffer-fill-col**(5). When an argument *n* is supplied *n* paragraphs are filled. If *n* is positive then MicroEmacs performs indented filling (i.e. indentation for a bullet mark etc). If *n* is negative then indented filling is disabled. If no argument *n* is supplied then the paragraph is filled and the *point* and *mark* positions are retained. This allows paragraphs to be filled, whilst in the middle of the paragraph and the word position is maintained.

If **justify mode** is enabled the variable **$buffer-fill-mode**(5) determines how the paragraph is filled (i.e. *left*, *right*, *both* or *center*). The variable **$fill-eos-len**(5) determines the trailing space used after a period ( **.** ) character (the trailing characters are specified by **$fill-eos**(5)), typically defined as 2.

A set of characters defined by **$fill-bullet**(5) enable bullet markers to be placed in the text at the beginning of the paragraph causing the left margin to be moved to the right of the bullet. The search depth for fill to locate a bullet character is defined by **$fill-bullet-len**(5). When the paragraph is formatted and one of the bullet characters is encountered then the user is prompted as to whether the paragraph should be indented following the marker or not. The point of indentation is shown with a <<<< marker.

Filling is automatically disabled on paragraphs which start with characters in the **$fill-ignore**(5) set.

The simple text formatting is generally used for mail messages, ASCII text README files etc.

## EXAMPLE

The following examples show how the text is formatted with indented filling enabled and both justification enabled:

```
This  is  regular  text  that  is on the
margin

    This is a regular  paragraph that is
    offset  from  the  margin.  Note how
    MicroEmacs retains the indent.

    * With  the  introduction  of one of
      the  special  characters,  in this
      case a  bullet,  a  format  of the
      paragraph  offsets  the text  from
      the bullet.

    1) Numbered  lists  are  the  same.
       Note that the  paragraphs are all
       separated with a blank line.

    1. Numbered  lists  ending  with  a
       period.

    label - Or labeled lists, separated
```

```
        with a dash.

>  '>' might be an ignore
>  character so it skips the paragraph
>
>         it is up to the user to
>   format these.
```

## SEE ALSO

**$buffer-fill-col**(5), **$buffer-fill-mode**(5), **$fill-bullet**(5), **$fill-bullet-len**(5), **$fill-col**(5), **$fill-eos**(5), **$fill-eos-len**(5), **$fill-ignore**(5), **$fill-mode**(5), **ifill-paragraph**(3), **paragraph-to-line**(3).

Ref: fill-paragraph(2) File: m2cmd052.2 Date: 2025/08/28

## NAME

filter-buffer - Filter the current buffer through an O/S command

## SYNOPSIS

**filter-buffer** (**C-x #**)

## DESCRIPTION

**filter-buffer** executes one operating system command, using the contents of the current buffer as input, sending the results back to the same buffer, replacing the original text.

This would typically be used in conjunction with **sort**(1), **awk**(1) or **sed**(1) to translate the contents of the buffer.

## SEE ALSO

**pipe-shell-command**(2).

Ref: filter-buffer(2) File: m2cmd053.2 Date: 2022/06/12

## NAME

find-file - Load a file

## SYNOPSIS

*n* **find-file** "*file-name*" (**C-x C-f**)

## DESCRIPTION

**find-file** finds the named file *file-name*. If it is already in a buffer, make that buffer active in the current window, otherwise attempt to create a new buffer and read the file into it.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** If the file does not exist and this bit is not set the command fails at this point. If the file does not exist and this bit is set (or no argument is specified as the default argument is `1`) then a new empty buffer is created with the given file name, saving the buffer subsequently creates a new file.

**0x02** If this bit is set the file will be loaded with **binary**(2m) mode enabled. See help on **binary** mode for more information on editing binary data files.

**0x04** If this bit is set the file will be loaded with **crypt**(2m) mode enabled. See help on **crypt** mode for more information on editing encrypted files.

**0x08** If this bit is set the file will be loaded with **rbin**(2m) mode enabled. See help on **rbin** mode for more information on efficient editing of binary data files.

**0x10** If this bit is set then the file's major-mode evaluation is not performed (i.e. the **$buffer-major-mode**(5) is not set and the file hook is not executed).

Text files are usually thought of as named collections of text residing on disk (or some other storage medium). In MicroEmacs the disk based versions of files come into play only when reading into or writing out buffers. The link between the physical file and the buffer is through the associated file name.

MicroEmacs permits full file names, i.e. you can specify:

```
disk:\directories\filename.extension
```

or (UNIX)

```
/directories/filename.extension
```

If the disk and directories are not specified, the current buffers disk/ directory is used. Several points should be noted in respect to the methods that MicroEmacs utilizes in the handling of files:

- Without explicitly saving the buffer(s) to file, all edits would be lost upon leaving MicroEmacs - you are asked to confirm whenever you are about to lose edits.
- MicroEmacs has a mechanism for "protecting" your disk-based files from overwriting when it saves files. When instructed to save a file, it proceeds to dump the file to disk, making a backup of the existing file when **backup**(2m) mode is enabled.

- Auto-saving files can be performed on edited buffers by setting the **$auto-time**(5) variable. The file is saved in the same place with a '#' appended to the file name. This can be used directly by the user or in the unlikely event of MicroEmacs crashing (or system crash), the files are automatically recovered next time it is edited.

If you do not wish to perform any edits but merely browse the file(s), add the **view**(2m) mode to the buffer or ask for the file to be read in for viewing only.

### RCS Support

If the file does not exist and the variable **$rcs-file**(5) is set then the existence of the RCS file is tested. If the rcs file exists then it will be checked out using a command-line created from the variable **$rcs-co-com**(5). If the check-out is successful then this file is loaded.

This raw interface for supporting file revision control systems has been adapted to support SCCS and Visual Source Safe see help on variable **$rcs-file** for more information and examples.

### HTTP Support

MicroEmacs supports http file loading, this is available by default on UNIX systems and most windows platforms (socket libraries not available on all win95 machines or earlier). When available a http file can be loaded by simply executing **find-file** and giving the http file name, i.e. "`http://user:password@address:port/file`". Only the `http://`, `address` and `/file` components are mandatory, the rest can usually be omitted. e.g.:

```
find-file "http://www.jasspa.com/index.html"
```

See help page on **%http-proxy-addr**(5) for information on HTTP proxy server support.

The progress of the HTTP transfer, and the HTTP commands issued, may be viewed in the `*http-console*` buffer. This is popped up depending on the setting of the **%http-flags**(5) variable.

### HTTPS Support

MicroEmacs can be compiled to support https using OpenSSL, when available the interface is the same as for http except the prefix is `https://`. When loading https URLs the libcrypt and libssl OpenSSL dynamic libraries must be available at runtime, if they are not available the function returns an error. The **%http-flags**(5) variable can be used to alter the behaviour, in particular the '**i**' flag which can be used to ignore certificate errors.

### FTP support

MicroEmacs supports ftp file loading, this is identical to http except the prefix `ftp://` is used as opposed to `http://`. The user name and password defaults to *anonymous* and "*ftp@example.com*" in the absence of both these fields. If the user name is supplied but not the password the password will be prompted for; this can be useful as the password will not be stored or written to the history file. Connection is by default on port 21.

```
find-file "ftp://<me>:<password>@members.xoom.com/jasspa/index.html"
```

See also **ftp**(3).

The progress of the FTP transfer, and the FTP commands issued, may be viewed in the `*ftp-console*` buffer. This is popped up depending on the setting of the **%ftp-flags**(5) variable.

### Secure FTP support

There are two standards for secure FTP, explicit and implicit.

Explicit ftp typically uses the default port 21 and the initial connection is established without encryption, the client then explicitly requests secure comms, hopefully before the user name and password are submitted. MicroEmacs supports explicit ftp by using a non-standard `ftpe://` URL prefix, this allows an explicit ftp URL to be distinguish from an open ftp request. Note that the ftp server may refuse the request to make the connection secure, this is not considered a failure by MicroEmacs so the ***ftp-console*** should be checked if secure comms is required.

Like **HTTPS**, implicit ftp requires a secure connect to be established up front, by default it uses port 990. MicroEmacs supports implicit ftp by using the standard `ftps://` URL prefix (not to be confused with sftp which is used to denote file transfer through an ssh connection).

## NOTES

The base name part (i.e. not the path) of `file-name` can contain wild-card characters compatible with most file systems, namely:

**?** Match any character.

**[abc]** Match character only if it is *a*, *b* or *c*.

**[a-d]** Match character only if it is *a*, *b*, *c* or *d*.

**[^abc]** Match character only if it is not *a*, *b* or *c*.

***** Match any number of characters.

If the name matches more than one file, a buffer will be created for each matching file. Note that these are not the same wild-card characters used by regex.

For *ftp* and *http* then a ftp console window is opened up to show the progress of the transfer (when configured), this is described in **ftp**(3).

The automatic loading of very large files can be avoided by setting the variable **$file-size-prompt**(5) to an appropriate size.

## SEE ALSO

**auto**(2m), **binary**(2m), **crypt**(2m), **rbin**(2m), **time**(2m), **view**(2m), **buffer-mode**(2), **file-attrib**(3), **file-op**(2), **find-hfile**(3), **ftp**(3), **next-window-find-file**(2), **major-mode**(2), **page-bfile**(3), **page-file**(3), **read-file**(2), **save-buffer**(2), **view-file**(2), **write-buffer**(2), **$buffer-major-mode**(5), **$file-size-prompt**(5), **$rcs-file**(5), **%ftp-flags**(5), **%http-flags**(5), **%http-proxy-addr**(5), **%sock-send-timeout**(5).

Ref: find-file(2) File: m2cmd056.2 Date: 2026/05/11

## NAME

find-registry - Index search of a registry sub-tree.

## SYNOPSIS

**find-registry** "*root*" *index*

## DESCRIPTION

**find-registry** performs an indexed search of a registry sub-tree allowing the caller to determine the names of the children that exist as sub-nodes of the specified node. *root* defines position in the hierarchy whose children are to be determined. *index* is a value from `0..n` and identifies the index number of the child node. The name of the child node is returned in **$result**(5) if one exists, otherwise an error status is returned.

## EXAMPLE

The following example comes from `addrbook.emf` and shows how **find-registry** is used to iterate through entries in the address book. Note that **find-registry** is used with **!force**(4) and the **$status**(5) of the call is tested to determine if the invocation succeeded.

```
!force find-registry "/AddressBook/Names" #l0
!if $status
    set-variable #l1 $result
    76 insert-string "_"
    2 newline
    insert-string &spr "Section: %s" #l1
    newline
    ; Iterate through all of the entries.
    set-variable #l2 0

    !repeat
        !force #l2 ab-buffer
        !if $status
            set-variable #l2 &add #l2 1
        !endif
    !until &not $status
    set-variable #l0 &add #l0 1
    !goto next
!endif
```

## SEE ALSO

**copy-registry**(3), **delete-registry**(2), **list-registry**(2), **mark-registry**(2), **read-registry**(2), **set-registry**(2), **&reg**(4), **erf**(8).

Ref: find-registry(2) File: m2cmd072.2 Date: 2022/06/12

## NAME

find-tag - Find tag, auto-load file and move to tag position

## SYNOPSIS

*n* **find-tag** "*string*" (**esc t**)

## DESCRIPTION

**find-tag** finds the current or given tag (*string*) in a **tags** file and goes to the given point, loading the file if necessary. The tag is either the current word under the cursor or a user supplied word if the cursor is not in a word. The buffer containing the tag is popped up in another window and the cursor moved to the tag in the new window.

A **tags** file is usually created by an external program (e.g. **ctags**(1)) which stores word references (or tags) and the name of the file containing the tag, with a search string to go to its local. It is an indexing system which is often used in programming.

The argument *n* can be used to change the default behavior of find-tag described above, *n* is a bit based flag where:

**0x01** Use popup-window to display the tag in a different window (default) when this flag is not given the current window is used to display the tag.

**0x02** Disable the use of the current cursor position to determine the tag. Instead the tag must always be supplied through "*string*".

**0x04** Find the next definition of the last tag (multiple tag support). This feature can only be used if multiple tag support is enabled (see flag '**m**' of variable **%tag-option**(5)) and **find-tag** has already been successfully executed. In this situation the last invocation of find-tag defines the current tag and executing again with an argument of `4` will jump to the next definition of the current tag or return the message "`[No more "<current>" tags]`".

The next tag is typically bound to `M-C-t`.

The **tags** file is, by default, assumed to reside in the current directory of the currently viewed file. The user variable **%tag-option**(5) may be specified with a value of '`r`' (recursive) and '`c`' (continue) flags, which ascends the directory tree from the current directory and attempts to locate a *recursively* generated tags file at a higher directory level. Recursive tag files are generally easier to maintain where project source files are located in a number of project sub-directories, and enable the whole of the project tree to be taggable.

Two user variables must be defined before **find-tag** will execute, if either **%tag-file**(5) or **%tag-template**(5) are not defined the error message "**[tags not setup]**" is signaled.

## NOTES

A **tags** file may be generated by MicroEmacs from the menu (*Tools->XX Tools->Create Tags File*). Alternatively a **tags** file may be generated by the **ctags**(1) utility. This is typically standard on UNIX platforms. For Windows and DOS platforms then the **Exuberant Ctags** is recommended, this is available from:

```
https://ctags.sourceforge.net/
```

A MicroEmacs compatible tags file may be generated using the command line "`ctags -N --format=1 .`" cataloging the current directory. To generate **tags** for a directory tree then use "`ctags -NR -format=1 .`". Refer to the **Exuberant Ctags** documentation for a more detailed description of the utility.

## SEE ALSO

**%tag-file**(5), **%tag-option**(5), **%tag-template**(5), **generate-tags-file**(3), **ctags**(1).

Ref: find-tag(2) File: m2cmd057.2 Date: 2023/02/21

## NAME

forward-char - Move the cursor right backward-char - Move the cursor left

## SYNOPSIS

*n* **forward-char** (**C-f**)
*n* **backward-char** (**C-b**)

## DESCRIPTION

**backward-char** moves the cursor *n* characters to the left. Move to the end of the previous line if the cursor was at the beginning of the current line.

**forward-char** moves the cursor *n* characters to the right. Move to the beginning of the next line if the cursor was already at the end of the current line.

For both commands, a negative value reverses the sense of movement as expected.

## NOTES

**backward-char** is also bound to **left**.
**forward-char** is also bound to **right**.

## SEE ALSO

**forward-line**(2), **backward-line**(2).

Ref: forward-char(2) File: m2cmd010.2 Date: 2026/06/19

## NAME

forward-delete-char - Delete next character at the cursor position
backward-delete-char - Delete previous character at the cursor position

## SYNOPSIS

*n* **forward-delete-char** (**C-d**)
*n* **backward-delete-char** (**backspace**)

## DESCRIPTION

**forward-delete-char** deletes the next *n* characters from the current cursor position. If the cursor is at the end of a line, the next line is joined onto the end of the current line. If a numeric argument is given the character(s) are added to the kill buffer, otherwise the kill buffer remains unaltered.

**backward-delete-char** deletes the next *n* characters immediately to the left of the cursor (e.g. more conventionally backspace). If the cursor is at the beginning of a line, this joins the current line to the end of the previous line. If an argument is given then the character is added to the kill buffer, otherwise the kill buffer remains unaltered.

Where the numeric argument is `0` the command has no effect. A negative argument changes the direction.

## NOTES

**forward-delete-char** is also bound to **delete** and **S-delete**.

**backward-delete-char** is also bound to **S-backspace**.

## HISTORY

As of December 2004 the mode **letter**(2m) which controlled whether the deleted character is added to the kill buffer was removed. Letter mode used to be disabled by default, the behavior of the old letter mode, when enabled, may be achieved through the key bindings:

```
1 global-bind-key backward-delete-char "backspace"
1 global-bind-key forward-delete-char "delete"
```

The use of the numeric argument of 1 in the global binding has the same effect as the existing **letter**(2m) mode (when enabled).

## SEE ALSO

**backward-char**(2), **backward-kill-word**(2), **forward-char**(2), **forward-kill-word**(2), **kill-paragraph**(2), **kill-region**(2).

Ref: forward-delete-char(2) File: m2cmd034.2 Date: 2023/06/29

## NAME

forward-kill-word - Delete next word at the cursor position
backward-kill-word - Delete previous word at the cursor position

## SYNOPSIS

*n* **forward-kill-word** (**esc d**)
*n* **backward-kill-word** (**esc backspace**)

## DESCRIPTION

**forward-kill-word** deletes the next *n* words starting at the current cursor position, the deleted text is added to the kill buffer. See **forward-word**(2) for a description of word boundaries. If no argument *n* is given and the current character is a non-word character then only the characters from the current position to the start of the next word are removed. If the argument *n* is 0 the command has no effect. If a negative argument is specified, *n* words are deleted backwards (i.e. this is the same as **-n backward-kill-word**).

**backward-kill-word** deletes the previous *n* words before the cursor, the deleted text is added to the kill buffer. The numeric argument has the same effect as with **forward-kill-word**.

## NOTES

**backward-kill-word** is also bound to **esc backspace**.

The negative argument is typically used by macro scripts where the kill buffer is more precisely controlled.

## HISTORY

The interpretation of the numeric argument was changed in the 2005 release to kill *n* words in a forwards or backwards direction. In previous releases then the negative argument controlled the kill buffer behavior. If kill buffer control is required then **kill-region**(2) should be used instead.

## SEE ALSO

**backward-delete-char**(2), **forward-delete-char**(2), **forward-word**(2), **kill-region**(2), **yank**(2).

## NAME

forward-line - Move the cursor to the next line
backward-line - Move the cursor to the previous line

## SYNOPSIS

*n* **forward-line** (**C-n**)
*n* **backward-line** (**C-p**)

## DESCRIPTION

**forward-line** moves the cursor down *n* lines, default 1. If the line is not on the current screen then display the next page and move to the line.

**backward-line** moves the cursor up *n* lines, if the line is not on the current screen then display the previous page and move to the line.

For both invocations a negative value reverses the sense of movement as expected.

## SEE ALSO

**backward-word**(2), **forward-word**(2), **scroll-down**(2), **scroll-up**(2).

Ref: forward-line(2) File: m2cmd092.2 Date: 2022/06/12

## NAME

forward-paragraph - Move the cursor to the next paragraph
backward-paragraph - Move the cursor to the previous paragraph

## SYNOPSIS

*n* **forward-paragraph** (**esc n**)
*n* **backward-paragraph** (**esc p**)

## DESCRIPTION

**forward-paragraph** moves forwards *n* paragraphs. The cursor is placed at the end of the *n*th paragraph at the start of the blank paragraph separation line. The default value of *n* is 1, if *n* is negative then the direction is backwards.

**backward-paragraph** similarly moves backwards *n* paragraphs. The cursor is placed at the start of the blank paragraph separation line. The default value of *n* is 1, if *n* is negative then the direction is forwards.

## DIAGNOSTICS

The following errors can be generated, in each case the command returns a FALSE status:

**[end of buffer]**
: When moving forwards, the given argument *n* was greater that the number of remaining paragraphs, the cursor is left at the end of the buffer.

**[top of buffer]**
: When moving backwards, the given argument *n* was greater than the number of paragraphs before the cursor, the cursor is left at the beginning of the buffer.

## NOTES

- For both invocations a negative value reverses the sense of movement as expected.
- A paragraph break is defined as a blank line.

## SEE ALSO

**backward-line**(2), **forward-line**(2), **kill-paragraph**(2), **scroll-down**(2), **scroll-up**(2).

Ref: forward-paragraph(2) File: m2cmd094.2 Date: 2026/02/28

## NAME

forward-word - Move the cursor to the next word
backward-word - Move the cursor to the previous word

## SYNOPSIS

*n* **forward-word** (**esc f**)
*n* **backward-word** (**esc b**)

## DESCRIPTION

**forward-word** places the cursor at the end of the *n*th word from the current position; the default is 1.

**backward-word** places the cursor at the beginning of the *n*th previous word, default 1.

For both commands, a negative value reverses the sense of movement as expected.

## NOTES

Words are distinguished by non-alphanumeric characters and need not be white space such as spaces and tabs.

A character is considered to be part of a word if it is in the **$buffer-mask**(5) character set. The default setting for **$buffer-mask** is "`luh`" which gives a word character set of the alphanumeric characters, i.e. `0-9`, `A-Z`, `a-z`, this may be changed by setting the **$buffer-mask** variable. The character sets (including 4 user character sets `1-4`) may be altered by using the command **set-char-mask**(2).

## SEE ALSO

**backward-line**(2), **backward-paragraph**(2), **forward-line**(2), **forward-paragraph**(2), Locale Support, **$buffer-mask**(5), **set-char-mask**(2).

Ref: forward-word(2) File: m2cmd096.2 Date: 2026/06/19

## NAME

generate-hash - Generate SHA1 or SHA256 hash or HMAC

## SYNOPSIS

*n* **generate-hash** [ "*key*" ] [ "*data*" ]

## DESCRIPTION

**generate-hash** computes the SHA256 hash value of the current buffer's content, as if it were written to file in its current state (i.e. the binary, line and file termination modes are taken into consideration). The hash is then turned into a hex string and the **$result**(5) variable is set to the resultant 64 byte string.

The numeric argument *n* is a bit based flag used to modify the default behaviour of the command, where the bits are defined as follows:

**0x01** If bit 2 is clear this defines whether the hash will be generated for the current region (if clear) or whole buffer (if set). If bit 2 is set then this bit determines whether the input *data* is a hex-string (if set) or plain text.

**0x02** When set the user is prompted for the *data*, in plain text or hex-string form (see bit 0x01) rather than the current region or buffer.

**0x04** When set a HMAC is calculated instead, the user will be prompted for the *key*.

**0x08** When calculating a HMAC, setting this bit indicates the *key* is given in the form of a hex-string.

**0x10** When set a SHA1 based hash/HMAC is calculated instead.

## EXAMPLE

To calculate the hash of a string use:

```
2 generate-hash "Hello world"
```

After executing the SHA256 hash value will be stored in **$result**. To calculate a HMAC-SHA256 of a string use:

```
6 generate-hash "TheKey" "Hello world"
```

After executing **$result** will be set to `1804C1869D6DD76A9A8D00819E6F62E970D981F01FE805F927C`

## NOTES

When using an argument of 2 the data must be a plain text string which means it cannot contain a byte of value 0. If the hash of binary data is required then set bit 1 and convert to a hex-string or create a temporary buffer, enable **rbin**(2m) mode, insert the data as a hex string and then call **generate-hash** with no argument to calculate the buffer's hash.

When calculating the hash of a region, if **binary**(2m) or **rbin**(2m) modes are enabled then the whole of the region's first line is always included , as is the whole of the last line if the region's end point column is greater than 0. If neither of these modes are enabled then the **cr**(2m) and **lf**(2m) line termination modes are taken into account, as is the **ctrlz**(2m) end of file mode.

## SEE ALSO

**$result**(5), **binary**(2m), **rbin**(2m), **cr**(2m), **lf**(2m), **ctrlz**(2m), **buffer-mode**(2), **SHA1**(3), **SHA256**(3), **HMAC-SHA256**(3).

Ref: generate-hash(2) File: m2cmd107.2 Date: 2024/05/08

## NAME

get-next-line - Find the next command line

## SYNOPSIS

*n* **get-next-line** (**C-x ‘**)

## DESCRIPTION

**get-next-line** is typically used in conjunction with the **compile**(3) and **grep**(3) commands to enable the user to step through errors/locations one by one. The command looks for lines in the form defined by **add-next-line**(2) in the order of definition. If a match is found the command attempts to find the next error or warning found from the current location (See **compile**(3)). If the buffer was not found then the next buffer set is searched for, and if found then the next expression from the cursor is automatically located. The command fails if none of the buffers exist, or the end of the buffer is reached.

The argument *n* can be used to change the default behavior of get-next-line described above, *n* is a bit based flag where:

**0x01** Use popup-window to display the buffer containing the next location in a different window (default) when this flag is not given the window containing the ’next’ command output, or the current window if the ’next’ command output is not currently displayed.

**0x02** When set the command returns an error if the current buffer is not a ’next’ command buffer, this effectively forces the command to only get the next line in the current buffer. This flag is used by ipipe output processing macros to highlight error lines in the current buffer.

**0x04** When set the command does not open the file/buffer containing the next location, leaving the ’next’ command buffer current with the cursor at the next line.

**0x08** When set the command does not attempt to get the next line, this can be used to simply go to the most current ’next’ command buffer or, if used in conjunction with bit **0x02**, to confirm the current buffer is a ’next’ command buffer.

## SEE ALSO

**$file-template**(5), **$line-template**(5), **add-next-line**(2), **keep-buffer**(3), **compile**(3), **grep**(3).

Ref: get-next-line(2) File: m2cmd058.2 Date: 2026/08/16

## NAME

global-abbrev-file, buffer-abbrev-file - Set abbreviation file(s).

## SYNOPSIS

*n* **global-abbrev-file** [ "*abbrev-file*" ]
*n* **buffer-abbrev-file** [ "*abbrev-file*" ]

## DESCRIPTION

The abbreviation files allow the user to define a set of short-cut expansion text, whereby a short sequence of chararacters are associated with a longer text segment. When the short sequence is entered, the user may elect to maually expand the sequnce with the associated replacement text. Provision for cursor positioning may be made in the replacement text.

**buffer-abbrev-file** sets the current buffer's abbreviation file (limit of one abbreviation file per buffer). **buffer-abbrev-file** does the minimal amount of work to increase speed at load-up. The first use of **expand-abbrev**(2) attempts to load the abbreviation file at which point errors may be reported.

An argument *n* of zero, forces the buffer abbreviation file to be uncached, such that the next abbreviation that is expanded forces a re-load of the abbreviation file. This is typically only used when an abbreviation file is being constructed and tested.

If argument *n* of 2 is given the command returns the current abbrev file name in **$result**(5).

**global-abbrev-file** assigns a global set of abbreviations accross ALL buffers, such that the abbreviation is available regardless of the current buffer type. The global abbreviation file has a lower presidence than the **buffer-abbrev-file**, hence the currently assigned **buffer-abbrev-file** is searched before the **global-abbrev-file**.

Similarly for **global-abbrev-file**, an argument of zero forces the global abbreviation file to be uncached and re-loaded on the next use.

An abbreviation is a string which is expanded to an alternate form, e.g.

**e.g.** -> **for example**

or

**PI** -> **3.1415926536**

An abbreviation file is an ordinary text file with a strict format, it is loaded only once at the first call to **expand-abbrev**(2), from then on it reminds buffered. An abbreviation file has an abbreviation per line, they cannot use multiple lines. This is not a draw back as the expansion string is executed using **execute-string**(2) so any MicroEmacs command may also be called.

For example the following expansion string inserts the string "`!continue`" and a newline:

"`!abort\r`"

Note that '`\r`' is used instead of '`\n`' as **C-m** is bound to **newline**(2) and not **C-j**. The expansion string can also make use of a few useful abbreviations:

**\p** Mark the current position (expanded to "`C-x C-a P`")

**\P** Move cursor to the marked position (expanded to "`C-x a P`")

See help on **execute-string**(2) for more useful abbreviations.

## EXAMPLE

The abbreviation must be on the left hand side followed by at least 1 space, the expansion string must then be on the same line in quotes. So for the given examples, the abbreviation file would be:

```
|
|e.g. "for example"
|PI   "3.1415926536"
|
```

The following abbreviation could be used for a C *if-else* statement.

```
|
|if "if(\p)\r{\r\r}\relse\r{\r\r}\r\P"
|
```

This is particularly useful for email address, e.g.

```
|
|JA "\"JASSPA\" <support@jasspa.com>"
|
```

The following example is MicroEmacs C-Mode abbreviation file for constructing C files. Remember **\p** is where the cursor is positioned following the expansion.

```
#i "#include <\p>\r\P"
#d "#define "
if "if(\p)\r{\r\r}\r\P"
ef "else if(\p)\r{\r\r}\r\P"
el "else\r{\r\p\r}\r\P"
wh "while(\p)\r{\r\r}\r\P"
sw "switch(\p)\r{\rcase :\rdefault:\r}\r\P"
```

## NOTES

Abbreviation files are given the extension **.eaf** in the MicroEmacs home directory.

One of the easiest ways to create more complex abbreviations is to record a keyboard macro, name it and then insert the resultant macro. See notes on commands **start-kbd-macro**(2), **name-kbd-macro**(2) and **insert-macro**(2).

Try to avoid using named key, such as "`up`" and "`return`", as the keyboard macro equivalent is not readable and is likely to change in future releases.

When using argument `2` to obtain the current abbrev file name, note that this will be the same value as passed initially to the **global-abbrev-file** or **buffer-abbrev-file** command so it may only be a base name. If a full file name is required, the following macro code can be used:

```
2 buffer-abbrev-file
!if &len &set #g0 $result
  set-variable #g1 &find #g0 ".eaf"
  ...
!endif
```

## FILES

**c.eaf** - C-Mode abbreviation file. **emf.eaf** - Macro code abbreviation file.

## SEE ALSO

**execute-string**(2), **expand-abbrev**(2), **insert-macro**(2), **iso-accents-mode**(3), **name-kbd-macro**(2), **start-kbd-macro**(2), **$result**(5), **eaf**(8).

## NAME

global-bind-key - Bind a key to a named command or macro
global-unbind-key - "Unbind a key from a named command or macro"

## SYNOPSIS

*n* **global-bind-key** "*command*" "*key*" (**esc k**)
*n* **global-unbind-key** "*key*" (**esc C-k**)

## DESCRIPTION

**global-bind-key** takes one of the named commands and binds it to a key. Thereafter, whenever that key is struck, the bound command is executed. If an argument *n* is given then the bound command is executed *n* times when the key is struck. (i.e. the command is passed the numeric argument '*n*').

**global-unbind-key** unbinds (detaches) a user entered *key* sequence (i.e. `C-x C-f`) from any command to which it may be bound. This does not work with buffer or message line key bindings, see **buffer-unbind-key**(2) and **ml-unbind-key**(2). If an argument of 0 is given to **global-unbind-key**, only a single key is obtained for the user, if the character is currently bound to the prefix command, the prefix binding and any sub-bindings are removed. **global-bind-key** calls **global-unbind-key** first if the key to be bound is already bound to something else.

If a negative argument is given to **global-unbind-key** then all bindings are removed, **caution** - removing all bindings interactively will render the current MicroEmacs session unusable. This can only be used within macro development where new bindings are created immediately afterwards.

The **global-bind-key** command, currently bound to `esc k`, prompts the user for the named command and the key to which it is to be bound. This help file gives a complete list of all built in commands, and some useful macros, a complete list of all commands and macros can be obtained by using the command completion (type `esc x tab tab`, see **ml-bind-key**(2)) or using the command **describe-bindings**(2).

The mouse buttons are considered to be *keys*, there is a *key* for each button press and release event, use **describe-key**(2) to get the binding key string.

The non-ASCII standard keys such as the cursor keys have 'standard' key names to make cross platform binding support easy. Some systems such as *termcap* do not have fixed key-bindings, for these key the users must use the command **translate-key**(2) to convert the system key binding to the standard key binding.

Permanent changes are done indirectly through the `me.emf` file. This is a file that MicroEmacs reads and executes (see **execute-file**(2)) during startup and hence results in the appearance of a permanent change in the key bindings. The syntax of commands in the `me.emf` file is described under the execute-file command. Of principal concern here are the two commands **global-bind-key** and **global-unbind-key**. The primary difference between the way parameters are passed to these commands in the `me.emf` file is that the keys are not typed in directly (as in the *control-I* key when you want `C-i`) but by symbolic names. Every key has a unique name which can be easily obtained with the current binding by using the command **describe-key**(2).

See help on Key Names for a description of the symbolic naming system and a complete list of valid key names. Also see Bindings for a complete list of default key bindings.

## EXAMPLE

### Alt P

```
global-bind-key "func" "A-p"
```

### Control F2

```
global-bind-key "func" "C-f3"
```

### Shift Alt Left Cursor

```
global-bind-key "func" "A-S-left"
```

### Control Alt Delete

```
global-bind-key "func" "C-A-delete"
```

> Note that binding **Control-Alt-Delete** is not recommended for MS-DOS systems for obvious reasons.

## NOTES

Some ASCII keys, such as <CR> (C-m), <tab> (C-i), <BACKSPACE> (C-h) have non-ASCII key bindings, namely "**return**", "**tab**", "**backspace**" etc. this is to allow separate key-bindings for the real "**C-m**" etc.

Be very careful in binding and unbinding keys since you could get into some very peculiar situations such as being unable to abort out of a command (if you unbind CTRL-G or bind it to something else) or recover from the bad binding/unbinding if you unbind **execute-named-command**(2) or the **global-unbind-key** command. As long as you leave yourself the opportunity to do both of the last two commands, you can recover from disastrous bindings/unbindings.

## SEE ALSO

**buffer-bind-key**(2), **buffer-unbind-key**(2), **describe-bindings**(2), **describe-key**(2), **ml-bind-key**(2), **ml-unbind-key**(2), **translate-key**(2).

Ref: global-bind-key(2) File: m2cmd013.2 Date: 2025/08/28

## NAME

global-mode - Change a global buffer mode
add-global-mode - Set a global buffer mode
delete-global-mode - Remove a global buffer mode

## SYNOPSIS

*n* **global-mode** "*mode*" (**esc m**)
**add-global-mode** "*mode*"
**delete-global-mode** "*mode*"

## DESCRIPTION

**global-mode** changes the state of one of the hereditary global modes. A buffer's modes are initialized to the global modes when first created. This command is very useful in changing some of the default behavior such as case sensitive searching (see the example below). See Operating Modes for a full list and description of modes. Also see **buffer-mode**(2) for a full description of the use of the argument *n*.

The **about**(2) command gives a list of the current global and buffer modes.
**add-global-mode** and **delete-global-mode** are macros defined in meme3_8.emf which use global-mode to add or remove a global mode. They are defined for backward compatibility with MicroEMACS v3.8 and for ease of use; they are simple macros, add-global-mode is defined as follows:

```
define-macro add-global-mode
    ; Has the require mode been given as an argument, if so add it
    !force 1 global-mode @1
    !if &not $status
        ; No - use 1 global-mode to add a mode
        !nma 1 global-mode
    !endif
!emacro
```

## EXAMPLE

The following example globally disables **exact**(2m) and **magic**(2m) modes, if these lines are copied to the user setup file then are searches will be simple and case insensitive by default:

```
-1 global-mode "exact"
-1 global-mode "magic"
```

## NOTES

Globally adding **binary**(2m), **crypt**(2m) and **rbin**(2m) modes is strongly discouraged as any file loaded would be assigned these modes. Instead use the numeric argument of command **find-file**(2) or commands **find-bfile**(3) and **find-cfile**(3).

**auto**(2m), **autosv**(2m), **backup**(2m), **cursor**(2m), **exact**(2m), **magic**(2m), **quiet**(5), **tab**(2m) and **undo**(2m) modes are present on all platforms by default. On Windows and DOS platforms **cr**(2m) and **lf**(2m) is also present and on DOS **ctrlz**(2m) is also present.

## SEE ALSO

Operating Modes, **buffer-mode**(2), **find-bfile**(3), **find-cfile**(3), **about**(2).

Ref: global-mode(2) File: m2cmd005.2 Date: 2026/02/16

## NAME

goto-alpha-mark - Move the cursor to a alpha marked location

## SYNOPSIS

**goto-alpha-mark** "*?*" (**C-x a**)
*0* **goto-alpha-mark**

## DESCRIPTION

**goto-alpha-mark** prompts user for an alpha character and sets the cursor position to the preset location. Alpha marks are specified on a per buffer basis, thus the current buffer is not changed, merely the current location in the buffer. The alpha mark must already be defined using **set-alpha-mark**(2). This functionality is useful for rapidly returning back to locations in large files.

**goto-alpha-mark** with a numeric argument of zero (0) returns a string containing all of the alpha marks that exist in the current buffer. The alpha mark string is returned in **$result**(5), this is an empty string if there are no alpha marks in the buffer.

## SEE ALSO

**set-alpha-mark**(2).

Ref: goto-alpha-mark(2) File: m2cmd059.2 Date: 2022/06/12

## NAME

goto-line - Move the cursor to specified line

## SYNOPSIS

*n* **goto-line** (**esc g**)
**goto-line** "*num*"

## DESCRIPTION

**goto-line** moves the cursor to the specified line in the buffer. The user is prompted for the new line number on the command line, which may be entered as a relative displacement ([+—−]*number*) from the current position, or as an absolute line number (*number*). If the number is preceded by + or − then this is treated as a relative displacement from the current line, otherwise it is an absolute line number.
If a positive argument *n* is supplied, **goto-line** moves to this line, e.g. to move the cursor to line 240:

```
240 goto-line
```

A special case of **goto-line** is operative if an argument of `0` is supplied, argument "*num*" must also be given as above except **goto-line** treats the line number or displacement as an absolute move, i.e. includes *narrowed out* sections when calculating the new position. If the new line lies within a narrowed out section (i.e. a section that has been hidden and is not visible on the screen) the narrow is automatically expanded. See **narrow-buffer**(2) for more information on narrowing.

Supplying a negative argument to goto-line results in an error.

## NOTES

After successfully calling goto-line, variable **$window-line**(5) is set to the required line number.

## SEE ALSO

**goto-alpha-mark**(2), **goto-matching-fence**(2), **narrow-buffer**(2), **$window-line**(5).

Ref: goto-line(2) File: m2cmd060.2 Date: 2022/06/12

## NAME

goto-matching-fence - Move the cursor to matching fence

## SYNOPSIS

*n* **goto-matching-fence** (**esc C-f**)

## DESCRIPTION

**goto-matching-fence** moves the cursor to the opposing fence character of the character currently under the cursor. The set of fence characters include `[ ... ]`, `{ ... }`, `( ... )` and `< ... >`. i.e. if the character under the cursor is '{' then **goto-matching-fence** moves the cursor to the matching closing '}' fence, and visa versa.

**goto-matching-fence** can also be used to move to a matching `' ... '` and `" ... "` fence, but, as the start and end fence is the same, bit `0x04` must be used to indicate whether the current fence is the open or close fence, see below.

**goto-matching-fence** can also be used to move the cursor between the start and end of a C/C++ `/* ... */` style comment. It can also be used to move to matching C/C++ `#if`, `#elif`, `#else` and `#endif` constructs, cycling through them in the given order.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When this bit is set (the default) the terminal bell is rung if the matching fence cannot be found.

**0x02** When this bit is set the command moves the cursor to the matching fence, updates the screen and pauses for **$pause-time**(5) before moving hte cursor back again. When omitted the cursor is simply moved to the matching fence.

**0x04** When this bit is set the fence the cursor is current at is treated as the close fence so **goto-matching-fence** moves backwards through the file looking for the matching open fence. When this bit is omitted the fence is assumed to be the open fence, this bit is only used for `'` and `"` type fences.

When the **fence**(2m) buffer mode is enabled the matching open fence is automatically displayed when the closing fence is typed. The length of time the matching fence is displayed for can be controlled by the **$pause-time**(5) variable.

## SEE ALSO

**fence**(2m), **$pause-time**(5), **goto-line**(2).

Ref: goto-matching-fence(2) File: m2cmd061.2 Date: 2026/06/19

## NAME

help - Help; high level introduction to help
help-command - Help; command information
help-variable - Help; variable information
help-item - Help; item information

## SYNOPSIS

*n* **help** (**esc ?**)
**help-command** "*command*" (**C-h C-c**)
**help-variable** "*variable*" (**C-h C-v**)
**help-item** "*item*" (**C-h C-i**)

## DESCRIPTION

The help commands provide a quick on-line help facility within MicroEmacs without invoking a third party documentation system (e.g. a browser such as **firefox**(1) or **winhelp**(1)).

The on-line help is assisted by a set of macros which enable key words within the help buffers to be located by either tagging (`esc t`) or by selection with the left mouse button. The tag mechanism supports command completion.

**help** provides general help on the philosophy and functionality of MicroEmacs, if an argument *n* of 0 is given to the command it changes the current buffer to the internal help buffer, typically named "`*on-line help*`". This is a hidden system buffer used to store all the on-line help and can be used for a variety of things. Note that access to this buffer must be via the **help** command, not **find-buffer** and the help command will also ensure the system help file is loaded first.

**help-command** describes the purpose of the given *command*.

**help-variable** Describes the purpose of the given *variable*, similar to **help-command**, only for variables.

**help-item** Describes the purpose of any given item, where item could be a command, variable or any aspect of MicroEmacs.

## FILES

The help files are ASCII text files located in the MicroEmacs home directory. The files are defined as follows:

`me.ehf` - Help text file.
`hkehf.emf` - Help macros.

## SEE ALSO

**osd-help**(3), **command-apropos**(2), **describe-bindings**(2), **describe-key**(2), **list-commands**(2), **list-variables**(2).

Ref: help(2) File: m2cmd065.2 Date: 2025/08/28

## NAME

hilight - Manage the buffer hilighting schemes

## SYNOPSIS

*0* **hilight** "*hil-no*" "*flags*" ["*nol*"] ["*buffer-scheme*" ["*trunc-scheme*"]]
**hilight** "*hil-no*" "*type*" "*token*" [["*rtoken*"]
[(["*close*" ["*rclose*"] "*ignore*"])—
(["*continue*"])—
(["*b-hil-no*"])]
"*schemeNum*"
**hilight** "*hil-no*" "*0x200*" "*token*"
**hilight** "*hil-no*" "*0x400*" "*from-col*" "*to-col*" "*schemeNum*"
*-1* **hilight** "*hil-no*" "*type*" "*token*"

## DESCRIPTION

The **hilight** command creates and manages the buffer hilighting, the process of creating a new hilighting scheme is best described in File Language Templates. The command takes various forms as defined by the arguments. Each of the argument configurations is defined as follows:

### Hilight Scheme Creation

Hilight scheme creation uses the following command form:

*0* **hilight** "*hil-no*" "*flags*" ["*nol*"] ["*buffer-scheme*" ["*trunc-scheme*"]]

With an argument of `0`, **hilight** initializes or re-initializes the hilight scheme *hil-no* (1-255). Every buffer has a hilight scheme, the default is 0 which means no hi-lighting and only the **$global-scheme**(5) etc. are used. The hilighting scheme must be defined before use and is used to specify how the buffer is to be hilighted. MicroEmacs supports the following hilighting concepts:

- **hilight string**, a user specified string is hilighted in any color scheme.
- **Tokens**, same as a hilight string except that the string must be enclosed in non alpha-numeric characters.
- **Start-of-line hilights**, the start of the hilight must be the first non-white character of the line.
- **End-of-Line hilights**, the hilight starts from the current position and continues until the end of the line. Optionally, the hilight may continue onto the next line if the current line ends in a given string. A bracket may also be searched for within the line.
- **Bracket hilight**, hi-lights from the current position until the closing bracket token is found.
- **Replace string** , allows the hilight string to be replaced with a different user specified string. (i.e. the displayed representation is different from the buffer contents)

Terminals that cannot display color directly may still be able to take advantage of the hi-lighting. A terminal that has fonts (i.e. *Termcap*) can use them in the same way using the **add-color-scheme**(2) command. The hi-light scheme is also used in printing (see **print-buffer**(2)). If

your terminal cannot display color in any way, it is recommended that hi-lighting is disabled (except when printing) as it does take CPU time.

The "*hil-no*" argument specifies which hi-lighting scheme is being initialized. Once a hilighting scheme has been initialized, hi-light tokens can be added to it and it can be used by setting the current buffer's **$buffer-hilight**(5) variable to "*hil-no*". The "*flags*" argument is a bit based flag setting global hi-light characteristics, where:

**0x01** The hi-light scheme is case insensitive, i.e. the following tokens become equivalent:

```
house == HOUSE == hOuSe
```

When the hilight scheme is attributed as case insensitive then the tokens must **all** be specified in **lower** case.

**0x02** Set a hi-light look-back. During the process of determining the window hilighting then the hilight process has to determine whether the top of the window starts in a hi-light bracket or not. The look-back command tries looking "*nol*" lines backwards for an open bracket. If an open bracket is found then the top of the window is assumed to start with that bracket, else it is assumed that the top of the window is not in a bracket. For example, in 'C', a comment starts with "/*" and ends with "*/" so if the hilight was initialized with

```
0 hilight 1 2 10 $global-scheme
```

of the following, only the first would begin hi-lighted which is correct (assuming the "/*" is 10 or less lines away).

```
 /* ........          /*.........            .........
    ........            .........*/          .........
----------------    ----------------     ---------------- top of
    ........*/          .........            .........    window
```

The optional argument "*buffer-scheme*" specifies the default scheme to use if there is no specific hi-light, when omitted the value of **$global-scheme**(5) is used. The *buffer-scheme* is a reference to a set of foreground and background color pairs previously defined with **add-color-scheme**(2). The last argument "*trunc-scheme*" is also optional and specifies the line truncation scheme, when omitted the value of **$trunc-scheme**(5) is used.

The hi-lighting scheme required is based on the type of file being edited and so is usually directly related to the **major-mode**(2), which is typically automatically determined by the file's extension or content (see **major-mode**(2) for more information on file type identification).

### Hilight Scheme Token Creation

Hilight scheme token creation uses the following command forms:

**hilight** "*hil-no*" "*type*" "*token*" ["*rtoken*"]
[(["*close*" ["*rclose*"] "*ignore*"])—
(["*continue*" ["*rcontinue*"]])—
( ["*b-hil-no*"] ) ]
"*schemeNum*"
**hilight** "*hil-no*" "*0x200*" "*token*"
**hilight** "*hil-no*" "*0x400*" "*from-col*" "*to-col*" "*schemeNum*"

With the default argument of `1`, **hilight** creates a hilight token to be used in hilight color scheme identified by "*hil-no*" (1-255) (see the section on **Hilight Scheme Creation** for a overview of

hi-lighting). The second argument "*type*" specifies the token type and must always be specified, it determines which other arguments required.

Typically the last argument, *schemeNum*, is also required. This identifies the color scheme to use when hilighting the token, defining the foreground, background and selection color schemes. This is an index generated from **add-color-scheme**(2). If the *schemeNum* argument is omitted the default hilght color scheme is used.

The token "**type**" is a bit based flag of which 0, 1 or more of the bits may be set, the effect of the bits are defined as follows:

**0x001** The "*token*" must be surrounded by non-word characters (word characters are typically the alpha-numeric characters), e.g. the following defines "`if`" as a token:

```
hilight 1 1 "if" .scheme.keyword
```

this hilights the 'if' in " `if` " but not in "`aifa`".

**0x002** Color this to the end of the line, often used for comments etc. For example in MicroEmacs macro language a '`;`' character signifies the rest of the line as a comment, hilighting is defined as follows:

```
; this is a comment line
hilight 1 2 ";" .scheme.comment
```

**0x004** This is a bracket token, the closing bracket string "*close*" and an ignore character "*ignore*" must also be supplied. The ignore character indicates that when found it should ignore the next character; this prevents an early end on bracket miss-match. For example, in C a '`"`' character can be inserted into a string by 'protecting' it with a '\' character, such as "*this is a string with a \" in it*". In this example the ignore character should be '\' so the mid string '`"`' is correctly ignored, as follows:

```
hilight 1 4 "\"" "\"" "\\" .scheme.string
```

An empty value, "", effectively disables the ignore feature. If replacing bit `0x040` is set the replacement close bracket "*rclose*" must be supplied.

**0x008** The token has a continuation string, usually used with 0x02 but cannot be used with token types `0x004` and `0x080`. The argument "*continue*" must be supplied and if the replacing bit `0x040` is set the replacement continue string "*rcontinue*" must also be supplied. The best example of its use can again be found in C; macros defined using the `#define` pre-processor construct may be constructed on single or multiple lines. The macro continues onto another line if the current line ends with a backslash '\' character, e.g.:

```
#define a_single_line_macro() printf("hello world\n")

#define a_four_lined_macro()          \
do {                                  \
    printf("hello world\n") ;         \
} while(0)
```

This can be correctly hilighted with the pre-processor scheme using:

```
; use to-end-of-line (2) and continuation (8), i.e. 2+8=10
hilight 1 10 "#" "\\" .scheme.prepro
```

**0x010**

If this is an end of line token (`0x002`) then
    The rest of the line is checked for any valid brackets.
Else if this is a bracket token (`0x004`) then
    This is still searched for after an end of line token is found.
Else Ignored

This feature enables the searching and hilighting of specific brackets contained within a to-end-of-line scheme. For example, consider the following C code:

```
#define My_Token 0x01  /* This is a multi-lined comment
                        * describing My_Token */
```

With the '#' pre-processor hilight (see bit 0x08 above) the #define line would all be hilighted with the pre-process scheme, the comment would be missed causing incorrect hilighting of the next line. Instead this feature may be used by both the pre-processor and comment hilight tokens to correctly hilight the above example:

```
hilight 1 26 "#" "\\" .scheme.prepro
hilight 1 20 "/\\*" "*/" "" .scheme.comment
```

**0x020** This token must be the first non-white character of the line.

**0x040** The token (and closing bracket tokens) are to be replaced by the given replacement strings. This is often utilized when displaying formated text such as MicroEmacs online help **ehf**(8) pages, the output from UNIX **man**(1) etc. In MicroEmacs help pages, the start of bold text is delimited with the character sequence "`\C[cD`" and ends with the character sequence "`\C[cA`", e.g.

```
"the word \C[cDbold\C[cA is in \C[cDbold\C[cA"
```

Obviously the hilight delimiters should not appear so the character sequence may be correctly drawn using a bracket token, starting with "`\C[cD`" and ending with "`\C[cA`", replacing both with an empty string:

```
hilight 1 0x44 "\C[cD" "" "\C[cA" "" "" .scheme.bold
```

**0x080** This is a branch token. When this token is found, the token (or the replace string) is colored using the given color *schemeNum* and then the current hilighting scheme is changed to "*b-hil-no*" (which MUST be defined by the time it is first used). The "*b-hil-no*" hi-light scheme should also contain a branch token which branches back to "*hil-no*" or "`0`" (which branches to **$buffer-hilight**(5)). A branch does not have to branch back to "*hil-no*", it may branch to any other hi-light scheme. The branches are not stacked and there is no limit on the nesting.

**0x100** The token must be at the start of the line.

**0x200** This is an invalid token in its own right, which is used for optimizing a hi-lighting scheme.

This has the second highest precedence (see **0x400**) and all other bits are ignored. Only the first 3 arguments are required. For example, if there are 11 tokens starting with "`delete-`" as with the hi-lighting of this buffer, then adding the token "`delete-`", while invalid in its own right, means that "`delete-`" is only checked for once. This also reduces the size of the internal hilighting tables so if the message "**Table full**" appears, the hilighting scheme should be reduced by removal of the common components.

**0x400** This is a column hilighting token, which allows absolute columns within a window to be hilighted (irrespective of the contents). This bit takes precedence over all other bits and all other bits are ignored. Column highlighting is a different concept to token in that it requires a "*from-col*" and a "*to-col*" column positions and a line will be hilighted in the given scheme between these two columns.

**0x800** The flag is used with bracket tokens (`0x04`) and indicates that the bracket is typically contained on a single line. This information is used by MicroEmacs in trying to avoid hilighting anomalies caused when the start and end tokens of the bracket are the same (e.g. a string's start and end token is `'"'`). Problems arise when the bracket starts on one line and closes on a later line, even with a large look-back, eventually the start bracket will become too far back and only the end bracket is found. But as this is the same as the open token it is mistaken for an open bracket and the strings become out of synch. This test can reset this if further down the file an open and close bracket is found on the same line. For this to have any effect, the hilighting scheme must use look-back (flag `0x02` of **Hilight Creation**).

**0x1000** Flag is typically used with a branch token (`0x04`) and indicates that the hilight exists to the end of the line or is terminated by the new line (i.e. encountering the end of line will removes the branch hilight).

Note that `0x004`, `0x008` and `0x080` are mutually exclusive and more than 1 should not be set in any one hilight token, if 2 or more are set the effect is undefined. Other than this there is no restrictions placed on the types of token used, although strange combinations like `0x006` may lead to unexpected results – hopefully not a core dump, but not guaranteed !

The token and close token of brackets may contain a limited subset of regular expression tokens as follows:

**^** When specified as the first character of the token, the token must be at the start of the line.

**$** The token must be at the end of the line, must be the last character.

**\{** Indicates the start of the hilighted part of the token, only one may be used per token. This token use is different from regex. The **\{** hilight start marker must be at a fixed point from the start of the token, that is a wild card sequence may NOT proceed the "\{" token, i.e. "`A\{ *B`" is correct, "`A *\{B`" is incorrect as it is proceeded by a wild card character.

**\}** Indicates the end of the hilighted part of the token, only one may be used per token. The rest of the token must be matched for it to be used but is not considered part of the token, i.e. hilighting continues on the character immediately after the "\}", not at the end of the token. Similar to the \< token, the length of the rest of the token must be fixed. This token use is different from regex.

**\(.\)** Groups are supported in hilighting, but they must only enclose a single character, closures ('*', '?' and '+') must come after the closure, i.e. use "\(.\)*", not "\(.*\)". Alternatives ("`\|`") are not supported.

Groups are used to capture a single, this single character may be tested again within the string using the syntax `\1..\9` as with regular expressions. Where the inverse character is required then the syntax `\!1..\!9` may be used. This NOT syntax is special to highlighting and is not present in the regular expression general syntax.

**.** Matches any character.

**[...]** Matches a single buffer character to a range of characters, for example to hilight MicroEmacs register variables (i.e. `#g0-#g9`, `#p0-#p9`, `#l0-#l9`) the following regex string may be used:

```
hilight 1 1 "#[gpl][0-9]"
```

This matches a token which starts with a '`#`', followed by a '`g`', '`p`' or '`l`' character and ends with a numerical digit. If the user required the replacement (bit 0x40) of the "#" to "#register" to aid readability, the replacement string some now needs to know whether the second character was a '`g`', '`p`' or '`l`' and which digit. Up to 9 groups ("`\(.\)`") can be use to store a store a single search character, which can be used later in the search string and in the replacement string by using the form "`\#`", where `#` is the range test number counting from the left, e.g. for the given example use:

```
hilight 1 65 "#\\([gpl]\\)\\([0-9]\\)" "#register\\1\\2"
```

The content of the brackets ([...]) include a set of special short cuts and regular expression syntax definitions as follows:

`[abc]` A list of characters.

`[a-z]` A range of characters.

`[-.0-9]`
A combination of character lists and ranges.

`[[:space:]]`
A white space character. See **set-char-mask**(2) for a full description on MicroEmacs character range support.

`[[:digit:]]`
A digit, 0-9.

`[[:xdigit:]]`
A hexadecimal digit, 0-9, a-f, A-F.

`[[:lower:]]`
A lower case letter, by default a-z.

`[[:upper:]]`
An upper case letter, by default A-Z.

`[[:alpha:]]`
A lower or upper case letter.

`[[:alnum:]]`
A lower or upper case letter or a digit.

`[[:sword:]]`
A spell word character.

**[^...]** Matches all characters except the given range of characters, e.g. "`[^[:space:]]`".

`\#` The same character which matched the #th group token. This functionality is best explained using UNIX **man**(1) output as an example, to create a bold character '**X**' it produces "`X\CHX`" where `\CH` is a backspace character thereby overstriking the first '`X`' with another creating a bold character. This can be checked for and simulated in MicroEmacs using the following:

```
hilight 1 64 "\\(.\\)\CH\\1" "\\1" .scheme.bold
```

The use of "`\1`" in the search string ensures that the second character is the same as the first. This is replace by a single character drawn in the bold scheme.

`\!#` The inverse character that matched the #th group token, i.e. NOT `\1`.

**? + *** Matches 0 or 1, 1 or more and 0 or more of the previous character or character range respectively.

`\a \A \w \W \l \L \m \u \U \d \D \h \H \x \s \S \e \g \n \r`
Marches (`a`)lphabetic lower, the special word characters (`w`)ord, (`l`)ower, (`m`)alhpanumeric, (`u`)pper, (`d`)igit, (`h`)exdigit (`xxx`) literal hexdecimal digit, white(`s`)pace, (`e`escape, (`n`)ewline, (`g`}bell, (`r`)formfeed. The uppercase letters are the inverse. (see Regular Expressions).

Following is a list of hilighting regular expression restrictions:

The number of characters to the left of a **\{** and to the right of a **\}** token must be fixed, i.e. the '**?**', '**+**' and '*****' tokens cannot be used before this token. Consider the hilighting of a C function name defined to be a token at the start of a line followed by 0 or more spaces followed by a '('. The following hilight token looks valid but the variable space match is incorrect as it is to the right of the **\}**:

```
hilight 1 0 "^\\w+\\}\\s*(" .scheme.function
```

Instead either the space match must be include in the function token hilighting (which may cause problems, particularly if printing with underlining) or by fixing the number of spaces as follows:

```
; include the spaces in the function hilighting
hilight 1 0 "^\\w+\\s*\\}(" .scheme.function
; or fix the number of spaces to 0, 1 ...
hilight .hilight.c    0 "^\\w+\\}(" .scheme.function
hilight .hilight.c    0 "^\\w+\}\\s(" .scheme.function
```

The **+** and ***** tokens match the longest string and do not narrow, e.g. consider the hilighting of a C goto label which takes the form of an alpha-numerical name at the start of a line followed by a ':' character. The token "`^.*:`" cannot be used as **.** will also match and move past the ending ':', ending only at the end of the line. As no narrowing is performed the final '**:**' in the token will not match and the label will not be hilighted. Instead a character range which excludes a ':' character must be used, e.g. "**^[^:]*:**".

A group should not be followed by a **?** or ***** closure, it should either be a fixed single character or followed by a **+** closure (in which case the last matching character is stored).

Following is a list of hilight type bit / token regex equivalents:

**0x01** "`[^word]\{????\}[^word]`"

**0x02** "`????.*`"

**0x20** "`^\s*\{????`" - (note that this is strictly incorrect as the `\s*` is to the left of the `\{`, it is correctly handled for the ease of use).

**0x100** "`^????`"

### Hilight Scheme Token Deletion

Deleting hilight scheme tokens use the following command form:

*-1* **hilight** "*hil-no*" "*type*" "*token*" With a negative argument **hilight** deletes the given "*token*" from a hi-light color scheme identified by "*hil-no*". The token "*type*" must also be specified to distinguish between normal and column token types.

## EXAMPLE

Examples of different scheme hilighting follows:

### Example 1

Hilighting a MicroEmacs character given in hex form, checking its validity (i.e. "`\x??`" where `?` is a hex digit):

```
hilight 1 0 "\\x[[:xdigit:]][[:xdigit:]]" .hilight.variable
```

Hilighting a C style variable length hex number (i.e. "`0x???`"):

```
hilight 1 1 "0[xX][[:xdigit:]]+" .hilight.variable
```

### Example 2

Replacing a quoted character with just the character (i.e. `'x'` -> `x`)

```
hilight 1 64 "'\\(.\\)'" "\\1" %magenta
```

### Example 3

The following example uses the branch hilighting feature to hilight each window line a different color to its neighbors by cycle through 3 different color schemes:

```
0 hilight .hilight.line1 0                        $global-scheme
  hilight .hilight.line1 0x80 "\\n" .hilight.line2 .scheme.no1
0 hilight .hilight.line2 0                        .scheme.no1
  hilight .hilight.line2 0x80 "\\n" .hilight.line3 .scheme.no2
0 hilight .hilight.line3 0                        .scheme.no2
  hilight .hilight.line3 0x80 "\\n" .hilight.line1 $global-scheme
```

### Example 4

Simulate the hilighting from the output of a UNIX man page (taken from hkman.emf):

```
0 hilight  .hilight.man 0                                $global-scheme
; ignore
hilight .hilight.man 64 ".\CH" ""                        $global-scheme
; normal underline/italic
hilight .hilight.man 64 "_\CH\\(.\\)\\}[^\CH]" "\\1"     .scheme.italic
hilight .hilight.man 64 "\\(.\\)\CH_\\}[^\CH]" "\\1"     .scheme.italic
; bold - first is for nroff -man
hilight .hilight.man 64 "\\(.\\)\CH\\1\\}[^\CH]" "\\1"   .scheme.bold
```

```
hilight .hilight.man 64 "_\CH_\CH_\CH_\\}[^\CH]" "_"     .scheme.header
hilight .hilight.man 64 "\\(.\\)\CH\\1\CH\\1\CH\\1\\}[^\CH]" "\\1" .scheme.header
; bold underline
hilight .hilight.man 64 "_\CH_\CH_\CH_\CH_\\}[^\CH]" "_" .scheme.italic
hilight .hilight.man 64 "_\CH\\(.\\)\CH\\1\CH\\1\CH\\1\\}[^\CH]" "\\1" .scheme.italic
```

This replaces the complex nroff character string with a single hi-lighted character (if you do not believe me then try it!).

## NOTES

MicroEmacs hilight was written with speed and flexibility in mind, as a result the user is assumed to know what they are doing, if not the effects can sometimes be fatal.

Within MicroEmacs macro language then a single backslash "\" is the escape character of the macro pre-processor. When writing a regular expression search string in the macro language then search string backslashes should be escaped. Hence a regular expression defined as "`\x41\\x41`" searches for the character '`A`' ('`A`' is hexadecimal 0x41 i.e. "`\x41`") followed by a single hexadecimal character ("`\\x`) followed by the literal string "`41`".

## HISTORY

The March 2005 release modified the NOT replacement pattern with "`\s!D`" where `D` is the numeric group identifier `1..9`.

The token `0x1000` was undocumented before March 2005 and is now a fully supported hilight mode.

## SEE ALSO

Regular Expressions, File Language Templates, **$buffer-hilight**(5), **major-mode**(2), **add-color-scheme**(2), **print-scheme**(2), **indent**(2), **$system**(5), **print-buffer**(2).

## NAME

hunt-forward - Resume previous search in forward direction hunt-backward - Resume previous search in backward direction

## SYNOPSIS

*n* **hunt-forward** (**C-x h**)
*n* **hunt-backward** (**C-x C-h**)

## DESCRIPTION

**hunt-forward** repeats the last search with the last search string in a forwards direction, from the current cursor position. **magic**(2m) and **exact**(2m) modes are operational.

**hunt-backward** repeats the last search with the last search string in a backwards direction, as per **hunt-forward**.

The numeric argument *n* is interpreted as follows:

**n > 0** The *n*th occurrence of the pattern is located.

**n < 0** The first occurrence of the pattern is located in the next *n* lines.

## DIAGNOSTICS

The command returns a status of `FALSE` if no previous search string has been established, or if the pattern could not be located (or *n*th pattern where *n* occurrences are requested). If the pattern is found within the given search criteria the return status is `TRUE`.

## SEE ALSO

**exact**(2m), **isearch-forward**(2), **magic**(2m), **narrow-search-backward**(3), **narrow-search-forward**(3), **search-backward**(2), **search-forward**(2),
Regular Expressions

Ref: hunt-forward(2) File: m2cmd070.2 Date: 2022/06/12

## NAME

indent - Manage the auto-indentation methods

## SYNOPSIS

*0* **indent** "*ind-no*" "*flags*" [ "*look-back*" ] [ "*indent-scheme*" ]
**indent** "*ind-no*" "*type*" "*token*" [ "*close*" "*ignore*" ] [ "*indent*" ]
**indent** "*ind-no*" "**t**" "*token*" [ "*token*" ] "*indent-scheme*"
**indent** "*ind-no*" "**x**" — "**w**" "*token*" "*indent*" "*indent-scheme*"
**indent** "*ind-no*" "**c-type**" "*indent*"
**indent** "*ind-no*" "**u**" "*comment-cont*"
*2* **indent**
*3* **indent** "*ind-no*" "*change-macro*"

## DESCRIPTION

The **indent** command creates and manages the auto-indenting methods, the process of creating a new indentation method is best described in File Language Templates. The command takes various forms as defined by the arguments.

In the following sections an *indent* length can take one of two forms, it can either be an absolute number in the range of +/-63, note however that a value of -63 has special meaning as it forces a relative indent level to 0 (no indent). Alternatively an indent can be expressed as multiples of the buffer's tab size followed by the letter `t`, see the many example below.

Each of the argument configurations is defined as follows:

### Indentation Method Creation

*0* **indent** "*ind-no*" "*flags*" [ "*look-back*" ] [ "*indent-scheme*" ]

With an argument of `0`, **indent** creates a new indentation method with the integer handle *ind-no*. The indentation method is assigned to a buffer by setting **$buffer-indent**(5) to *ind-no*. *ind-no* cannot be 0 as setting **$buffer-indent** to zero disables indentation. If the indentation method with the same *ind-no* already exists, then the existing method is deleted and a new method may be created.

*flags* Sets the indent bit flags where:

`0x01` Indent method is case insensitive. Note that **indent** tokens must be specified in lower case.

`0x02` Defines a **fast look-back** indentation scheme to be used in conjunction with the current indentation scheme. The fast look-back scheme is used to identify a change in the current buffer's indent scheme, for example the starting of embedded script in html. The *indent-scheme* argument is required when this flag bit is set and identifies the handle of the fast look-back indentation scheme to be used.

This feature is only used when indent is required to support multiple complex indentation schemes within a single file, for example indentation of scripts within html, asp or php pages etc. When defined the given *indent-scheme* must also be defined using **indent**, the scheme's *look-back* argument is typically set to look back a long way to ensure correctness, i.e. 500 lines, but it should therefore contain few tokens to

maintain performance.

`0x04` Defines C mode of indentation. C styling is built into the editor and is enabled through the **indent** command. The C form is discussed in more detail in the next section.

`0x08` Ring the bell if **goto-matching-fence**(2) fails.

`0x10` Indent top level code, i.e. code outside braces; this is only valid when C mode is defined. Code can only exist in functions for most pre-compiled languages such as C, so a naked `if` statement is misplaced and indented to 0, whereas other languages such as Javascript can have code anywhere in the file, set this flag for these languages.

*look-back* specifies the maximum number of lines, prior to the current line, considered when calculating the indentation of a line, i.e. if there are *look-back* number of lines between the line to be indented and the previous non-blank line then the current indentation is lost. This is only valid when C mode is not defined.

If *look-back* is set to 0 then the indentation is effectively disabled as the current indentation can never be found. The maximum value has been increase as of January 2004 from 255 to an integer (large number), a value of 10 is typically sufficient for a normal indent scheme and 500 for a fast look-back scheme.

### C Indentation

*0* **indent** "*ind-no*" "*flags*"
**indent** "*ind-no*" `s` "*statement*"
**indent** "*ind-no*" `c` "*continue*"
**indent** "*ind-no*" `x` "*max*"
**indent** "*ind-no*" `b` "*brace*"
**indent** "*ind-no*" `e` "*braceStatement*"
**indent** "*ind-no*" `w` "*switch*"
**indent** "*ind-no*" `a` "*case*"
**indent** "*ind-no*" `l` "*label*"
**indent** "*ind-no*" `m` "*margin*"
**indent** "*ind-no*" `u` "*comcont*"

C mode is built into the editor by default and is enabled by setting the indent intialization flag bit `0x04`. With this bit enabled then the indentation for 'C' mode is defined through the additional arguments. *ind-no* and *flags* have the same meaning as defined in the previous section. C mode is typically initialized with the value 12. The remaining C arguments are defined as follows:

**indent** "*ind-no*" "`s`" "*statement*" (=t)
The indent of the current line is derived from *statement* value plus the indent of the last c token (*if*, *else*, *while* etc.) or the last '{' (which ever was found first). i.e. if the last '{' was found at column 16 then the current line will be indented to 20:

```
{
    xxxxxxxxxx
    xxxxxxxxxx
```

or

```
if(xxxxx)
    xxxxxxxxxx
```

C tokens are only used to indent the next line, whereas '{' are used in indenting every line to it's partnering '}'.

**indent** "*ind-no*" "c" "*continue*" (=3/2t)

*continue* sets the indent to be added to a split line, i.e. for an indent of 20, a continued statement would be indented to 30. A continued statement is a single c statement which is spread over 2 or more lines, the 2nd and any following lines would be indented to 30. For example

```
thisIsAVeryLongVariableWhichMeansAssignmentsAreSplit =
          ThisIsTheFirstContinuedStatementLine +
          ThisIsTheSecondContinuedStatementLine + etc ;
```

The indent is changed if there is an open bracket, continued statements are indented to the depth of the open bracket plus one, e.g.

```
func(firstFuncArg,
     secondFuncArg,
     anotherBracketForFun(firstAnotherBracketForFunArg,
                          secondAnotherBracketForFunArg),
     thirdFuncArg) ;
```

**indent** "*ind-no*" "x" "*max*"

*max* sets an upper limit of the indentation where an open bracket is encountered, in the case where the leading indent of the function name and open bracket exceeds *max*, then the continuation is reduced to the continuation indent.

The effect of *max* is described as follows; if *max* is set to a large value then the default open brace offset appearence is:

```
longVariable = LongFunctionNameWhichMeans(isSoFar,
                                          OverAndYouRunOutOfRoom) ;
```

Setting *max* to 16 gives:

```
longVariable = LongFunctionNameWhichMeans(isSoFar,
                overAndYouRunOutOfRoom) ;
```

Where by the second argument indent has been artificially reduced because of it's length.

**indent** "*ind-no*" "b" "*brace*" (=0)

Sets the indent of a '{' and a '}' on a new line from the current statement indent. For example, a value of 0 results in the following

```
if(xxxxxx)
{
    xxxxxxxxxx
    xxxxxxxxxx
}
```

With a setting of 2, this would become:

```
if(xxxxxx)
  {
    xxxxxxxxxx
    xxxxxxxxxx
  }
```

This works in conjunction with *statement* and *braceStatement* a change to *statement* will change the position of '{'s. In summary:

```
if (condition)      if (condition)     if (condition)
{                       {                 {
    statement;          statement;          statement;
}                       }                 }

brace = 0             brace = 4         brace = 2
```

**indent** "*ind-no*" "e" "*braceStatement*" (t)
The *braceStatement* defines the indentation following a curly brace '{' which follows a statement.

**indent** "*ind-no*" "w" "*switch*" (=0)
The additional indent to be applied after a **switch** statement, the *statement* indent is applied in addition to the *switch* indent. See the *case* argument.

**indent** "*ind-no*" "a" "*case*" (=-t)
*switch* value sets the offset of a "case" entry statement from the opening brace left margin position. The default value is zero. e.g.

```
switch(xxxxxxxxx)
{
case 1:
    xxxxxxxxxx
    xxxxxxxxxx
case 2:
    xxxxxxxxxx
}
```

Setting the value to 4, increases the leading space on the "case" statement, e.g.

```
switch(xxxxxxxxx)
{
    case 1:
        xxxxxxxxxx
        xxxxxxxxxx
    case 2:
        xxxxxxxxxx
}
```

*case* sets the offset of the lines following a "case" statement, from the current indent. For example, using the default settings, if the current indent was 20 then a line starting with a "case" would be indented to 16, i.e.

```
    xxxxxxxxxx
case xxxxxxxxxx
    xxxxxxxxxx
```

This is used inside "switch" statements, the default setting give the following layout:

```
switch(xxxxxxxxxx)
{
case 1:
    xxxxxxxxxx
    xxxxxxxxxx
case 2:
```

This works in conjunction with the *statement* variable, a change to *statement* will change the position of '{'s.

In summary:

```
switch (variable)    switch (variable)
{                    {
case 1:                  case 1:
    break;                   break;
default:                 default:
    break;                   break;
}                    }

  switch = 0           switch = t
  case = -t            case = -t
```

**indent** "*ind-no*" "l" "*label*" (=-63)
The additional indent to be applied to a line that starts with a **label** statement, a value of -63 (most negative number) is interpreted as an infinitely large negative number to ensure the indent ends up being 0. In C labels are typically left aligned so that they standout so the *label* indent is set to -63, giving the following:

```
        norm-code;
label:
        norm-code;
```

However, in javascript labels can be used more diversely so tend not to be indented, i.e. *indent* set to 0:

```
        norm-code;
        label:
        norm-code;
```

**indent** "*ind-no*" "m" "*margin*" (=-1)
If inserting a comment at the end of a C line, it is tedious typing *x* number of spaces to the comment column (by default tab does not insert a tab when **indent**(2) is enabled, it reformats the indentation of the line regardless of the cursor position). This variable sets the indent column of these comments. So with the default settings and the following line,

```
        xxxxxx ;/
```

when a '*' is entered the line becomes

```
        xxxxxx ;                /*
```

The indenting of the "/*" occurs only if there is text on the line before it, and none after it. If the current column is already past *margin* then it is indented to the next tab stop.

A value of -1 disables this feature.

In summary:

```
    statement;        /* Comment */      statement; /* Comment */

    margin = 40                          margin = -1
```

**indent** "*ind-no*" "u" "*comcont*" (" * ")

This defines the string which is inserted when a new line is started while in a comment. The string is only inserted if the cursor is at the end of the line when the **newline**(2) command is given. For example, for the default settings, if a **newline** was entered at the end of the first line, the second line would initialize to:

```
/* xxxxxxxxxx
   @
```

where '@' is the current cursor position. With a setting of " * ", then:

```
/* xxxxxxxxxx
 * @
```

In summary:

```
/*                  /*                /* Comment
 * Comment           ** Comment           Comment */
 * Comment           ** Comment
 */                 */
 comcont = " * "     comcont = "** "    comcont = "    "
```

The standard indent rule for setting up C and Java is defined as:

```
0 indent .hilight.c 12
indent .hilight.c "s" t     ; Default may be omitted
indent .hilight.c "e" t     ; Default may be omitted
indent .hilight.c "c" 3/2t  ; Default may be omitted
indent .hilight.c "x" 0     ; Default may be omitted
indent .hilight.c "b" 0     ; Default may be omitted
indent .hilight.c "w" 0     ; Default may be omitted
indent .hilight.c "a" -t    ; Default may be omitted
indent .hilight.c "m" -1    ; Default may be omitted
indent .hilight.c "u" " * "
```

This is typically written with the defaults omitted.

```
0 indent .hilight.c 12
indent .hilight.c "u" " * "
```

If GNU Emacs indentation is required then the following definition may be used:

```
0 indent .hilight.c 12            ; if (a == b)
indent .hilight.c "e" t           ;   {
indent .hilight.c "b" t           ;     c = d;
indent .hilight.c "s" t           ;   }
indent .hilight.c "u" " * "
```

The default configuration scheme is defined as:

```
0 indent .hilight.c 12            ; if (a == b)
indent .hilight.c "e" t           ; {
indent .hilight.c "b" 0           ;   c = d;
indent .hilight.c "s" t           ; }
indent .hilight.c "u" " * "
```

Where comments are to be aligned at the end of a statement then the margin parameter may be defined in <user>.emf.

```
indent .hilight.c "m" 40
```

### Indentation Rule Creation

**indent** "*ind-no*" "*type*" "*token*" [ "*close*" [ "*ignore*" ]] [ "*indent*" ] [ "*indent2*" ] [ "*indent-scheme*" ]

With the default argument of `1`, **indent** creates a new rule for the indentation method *ind-no* which must have previously been defined and initialized. An argument of `-1` will remove the given token from indentation method *ind-no*.

The indentation of a line in a buffer, which is using an indentation method, is affected by the token types matched on the line (*type* `f`, `o`, `s`) and the current indentation (if line is not of type `f`).

The current indentation is determined by searching the previous lines (look-back) for the indentation of the last indented line. This may not simply be the indentation of the last non-blank line, the exact indentation is determined by searching for tokens in the line and assessing their effect on the indentation of the current line.

The format of the regex valid in the "*token*" and "*close*" arguments are the same as at used by hilight token creation, see **hilight**(2) for more information.

The *indent* argument specifies the indent to apply in characters. The indent may be specified as a literal integer value, a positive value increases the indent a negative value reduces the indent. The *indent* may be specified in terms of buffer specific generic value **$buffer-indent-width**(5) using the value **t**, indicating a tab width. The value of **t** may be prefixed with a signed integer or vulgar fraction which adjusts the size of the indentation width by the appropriate amount. Modifying the value of $buffer-indent-width modifies the indent width that is applied allowing the indentation method to be specified independently of the indentation width.

The indent tokens may be assigned one of the following types, using the *type* argument. If the type is specified in upper case then the token must be surrounded by non-alpha-numeric characters:

**Fixed** (*type* = '`f`' or '`F`')

A line containing a fixed indent token will be indented to the given *indent* column from the left-hand edge. *indent* is the only argument specified. e.g. MicroEmacs macro `!goto` labels:

```
indent .hilight.emf f "*" 0
```

producing

```
    .....
*label
    .....
```

The fixed token must be the first non-white character on the line, the rest of the line is ignored. The indentation of the previous line has no effect.

**Indent-from-next-line-onward** (*type* = '`n`' or '`N`')

The indentation changes by *indent* from the next line onwards from the current line. *indent* is the only argument specified. e.g. MicroEmacs macro `!if`:

```
indent .hilight.emf n "!if" t
```

Keeps the indentation of the `!if` line the same as the previous indentation, change the indentation on the following lines by an extra $buffer-indent-width characters, to produce:

```
....
!if
    ....
```

**Indent-from-current-line-onward** (*type* = 'o' or 'O')

Increment the current and following lines indentation by *indent*. *indent* is the only argument specified. e.g. MicroEmacs macro `!endif`

```
indent .hilight.emf o "!endif" -t
```

decrement the indent of the `!endif` line and following lines by $buffer-indent-width spaces producing:

```
    ....
!endif
....
```

**Universal-indent** (*type* = 'u' or 'U')

Independent control of the current line and next line indent. Increment the current line indentation by *indent* and the following line indentation by *indent2*. Where *indent* represents indent with the `o` option and *indent2* represents indent with the `n` option. This command is useful where the indentation of both the current and next line need to be controlled independently given a single token.

```
indent .hilight.emf u "<cfelse" -t 3/2t
```

This has been used in the *ColdFusion* indentation to control the `<cfelse>` condition which requires control of both the current line and next line with different values from the same token. Note that a **Indent-single** could be implemented with this command using an argument of `-t t`.

**Indent-single** (*type* = 's' or 'S')

Changes the indentation of the current line ONLY by *indent*. *indent* is the only argument specified. e.g. MicroEmacs macro `!elif`:

```
indent .hilight.emf o "!elif" -t
```

decrements the indentation of the `!elif` line by $buffer-indent-width characters, but restores the previous indentation after the current line, producing:

```
    ....
!elif
    ....
```

**Bracket** (*type* = 'b' or 'B')

A bracket should be used when a starting token pairs with a closing token which may span multiple lines. i.e. the opening and closing braces of a programming language. Note that the opening and closing tokens must be different otherwise they cannot be differentiated. A bracket has two main effects:

When the previous line has an unmatched open bracket

In this situation the current line is indented to the right of the mismatched bracket.

When the previous line has an unmatched close bracket

In this situation the matching open bracket is hunted for in previous lines until either the *look-back* limit (See **Indentation Method Creation**) is exhausted or the bracket is matched, in which case the indent of that line is used.

For a bracket the only other argument given is the *close*. e.g. tcl's '`(`' and '`)`' brackets

```
indent .hilight.tcl b "(" ")"
```

Which produces:

```
....
.... (....
      ....
      ....)
....
```

**Continue** (*type* = 'c' or 'C')
Indicates that when *token* is found on the current line, the next line is a continuation of the current line. The indentation of the next line is the indentation of the first continuation line plus the given *indent*. *indent* is the only argument specified. e.g. tcl's '\'

```
indent .hilight.tcl c "\\" 3/2t
```

A simple example is

```
....
12345678901234567890      \
         ....
....
```

When used in conjunction with brackets, the following effect is observed:

```
....
12345678901234567890      \
          ....(....       \
               ....)      \
          ....            \
          ....
....
```

This shows why the first continuation line (the `123456...` line) must be located and used as the base line from which the indentation is derived; again the *look-back* limits the search for this line.

**Exclusion** (*type* = 'e' or 'E')
Used to exclude text between start *token* and *close* token from the indentation calculation, typically used for quotes. The *ignore* argument is also specified (see **hilight**(2) `type 0x004` type bracket) e.g. MicroEmacs macro quotes:

```
indent .hilight.emf e "\"" "\"" "\\"
```

e.g. tcl's quotes

```
indent .hilight.tcl e "\"" "\"" "\\"
```

producing:

```
....
".... ignore { ... \" ... ignore another { token ... "
....
```

**Ignore** (*type* = 'i' or 'I')
Text to the right of a line containing *token* is to be ignored; typically used for comments. e.g. MicroEmacs macro '**;**' comment:

```
indent .hilight.emf i ";"
```

Or tcl's '#' comment

```
indent .hilight.tcl i "#"
```

producing

```
....
# ... ignore this { indent token
....
```

**Type** (*type* = 't' or 'T')

Used in a **fast look-back** indentation scheme (see bit 0x02 of **0 indent** above) to define the scheme change tokens. The fast look-back token typically shadows the x and w types and identifies the indentation scheme. The token itself does not change the indentation, it simply identifies the indentation scheme. The 'x' or 'w' types are used to perform the actual indentation of the line if required.

If the *indent-scheme* is defined as zero (0) then the indentation scheme is set to the same value as the **$buffer-indent**(5) variable.

The following example is taken from the XML language definition . A fast-look back scheme is used (hilight.xmlT) to search for the alternative indentation schemes for *javascript* and *vbscript*. The t indent type defines the start of script tokens and to switch the indentation to the new indent type.

```
; Fast look-back indentation scheme
0 indent .hilight.xmlT 1 500
; Indentation scheme
0 indent .hilight.xml 3 20 .hilight.xmlT

!if &exist .indent.javascript
    indent .hilight.xmlT t "<script ... javascript>"   .indent.javascript
    indent .hilight.xml  x "<script ... javascript>" 2 .indent.javascript
    indent .indent.javascript w "</script>" -2 0
!endif
!if &exist .indent.vbscript
    indent .hilight.xmlT t "<script ... vbscript>"   .indent.vbscript
    indent .hilight.xml  x "<script ... vbscript>" 2 .indent.vbscript
    indent .indent.vbscript w "</script>" -2 0
!endif
indent .hilight.xmlT t "</script>" 0
indent .hilight.xml  o "</script>" -2
; Define the indentation tokens. XML is well defined so there are
; always opening and closing brackets.
indent .hilight.xml e "\"" "\"" "\\"
indent .hilight.xml b "<!--" "-->"
indent .hilight.xml b "<\\?" "\\?>"
indent .hilight.xml n "<[^/!]" 2
indent .hilight.xml o "</"    -2
indent .hilight.xml o "/>"    -2
```

**Change** (*type* = [x—X—w—W—v—V])

Change the indentation scheme from the current scheme to the new *indent-scheme*. If the *indent-scheme* is specified as zero (0) then the indentation scheme is set to the same handle value as the **$buffer-indent**(5) variable. *indent* specifies the indent value, when the *type* is w then the current line onwards is indented (similar to o), a *type* of x indents the next line onwards (similar to n), a *type* of v indents both the current line and next time onwards (similar to u).

Refer the the *type*=t for an example of indentation switching.

### Get Current Indentation Scheme

*2* **indent**

With an argument of 2, **indent** sets the **$result**(5) variable to the indent scheme used for the current location, this can be used for files that support multiple formats in the same file, such as HTML which can have section of Javascript embedded within it, to identify the major mode of the current cursor position.

### Set Indent Change Callback Macro

*3* **indent** "*ind-no*" "*change-macro*"

With an argument of 3, **indent** sets the indent change callback macro for the given indent scheme. This macro is called whenever the MicroEmacs indenter is about to change the indentation of a line. The macro is called with the current cursor set to the line whose indent is to change with a numeric argument of the new indent level. If the macro sets its command variable **.change-macro.indent** to a valid indent level and aborts rather that return, the line will be indented to the value of **.indent** instead.

## EXAMPLE

Examples of indentation method creations can be found in macro files `hkemf.emf`, `hktcl.emf` and `hkvrml.emf`. The following example is taken from `hkemf.emf`:

```
!if &not &exist .hilight.emf
    set-variable .hilight.emf &pinc .hilight.next 1
!endif

...

0 indent  .hilight.emf 0 10
indent .hilight.emf N "define-macro" t
indent .hilight.emf n "!if" t
indent .hilight.emf s "!eli" -t
indent .hilight.emf s "!els" -t
indent .hilight.emf o "!end" -t
indent .hilight.emf n "!whi" t
indent .hilight.emf o "!don" -t
indent .hilight.emf n "!rep" t
indent .hilight.emf o "!until" -t
indent .hilight.emf o "!ema" -t
indent .hilight.emf e "\"" "\"" "\\"
indent .hilight.emf i ";"
indent .hilight.emf f "*" 0
```

Note that a **.hilight** command variable is typically used as a buffer with indentation rules will almost certainly have hilighting and the same variable is used to define the hilighting scheme.

## NOTES

The variables **$indent-width**(5), **$buffer-indent-width**(5) and the **t** indent notation were introduced in the November 2004 version of MicroEmacs.

The electric C mode **cmode**(2m) was removed in December 2004, as were the variables **$c-brace**(5), **$c-case**(5), **$c-contcomm**(5), **$c-continue**(5), **$c-margin**(5) and **$c-statement**(5). All replaced by the C Indent initialisation method.

The Universal indent (`u`) was introduced in the 2006 release.

## SEE ALSO

File Language Templates, **$buffer-indent**(5), **$buffer-indent-width**(5), **$indent-width**(5), **major-**

**mode**(2), **hilight**(2).

Ref: indent(2) File: m2cmd033.2 Date: 2026/01/17

## NAME

insert-file - Insert file into current buffer

## SYNOPSIS

*n* **insert-file** "*file-name*" [ *upper-offset lower-offset byte-length* ] (**C-x C-i**)

## DESCRIPTION

**insert-file** inserts the named file *file-name* into the current buffer at the beginning of the current line. The buffer mark is set to the start of the insertion and the cursor is moved to the end. If the *file-name* is a directory then a directory listing is inserted into the buffer.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x02** If this bit is set the current buffer's file permissions and modification time are set to those of the file being inserted.

**0x04** If this bit is set then a partial insert is performed based on skipping the first (**upper-offset** << 32) + **lower-offset** bytes and inserting the next **byte-length** bytes. If **byte-length** is negative then the same is done except moving from the end of the file. Partial insert is only supported for regular files on a local disk.

## SEE ALSO

**find-bfile**(3), **find-file**(2), **insert-file-name**(2), **page-bfile**(3), **page-file**(3), **set-mark**(2), **view-file**(2).

Ref: insert-file(2) File: m2cmd073.2 Date: 2023/02/21

## NAME

insert-file-name - Insert filename into current buffer

## SYNOPSIS

*n* **insert-file-name** (**C-x C-y**)

## DESCRIPTION

**insert-file-name** inserts the current buffer's file name into the current buffer or, if entering text on the message line then enters the file name into the message line buffer.

If the numeric argument *n* is `-1` then the buffer name is inserted into the buffer.

## SEE ALSO

**insert-file**(2), **yank**(2).

Ref: insert-file-name(2) File: m2cmd050.2 Date: 2022/06/12

## NAME

insert-macro - Insert keyboard macro into buffer

## SYNOPSIS

**insert-macro** "*command*"

## DESCRIPTION

**insert-macro** inserts the named *command* into the current buffer in the MicroEmacs macro language, thus enables it to be saved, re-load and therefore re-used at a later date. This is often used in conjunction with **start-kbd-macro**(2), **end-kbd-macro**(2) and **name-kbd-macro**(2). The given *command* must have been defined either by a keyboard macro or in MicroEmacs macro code.

## NOTES

The **insert-macro** provides a good method of identifying unknown low level key codes. Simply record the unknown key as a macro and insert the macro into the scratch buffer. The low level key code appears within the string.

## SEE ALSO

**start-kbd-macro**(2), **name-kbd-macro**(2), **define-macro**(2), **execute-file**(2).

Ref: insert-macro(2) File: m2cmd074.2 Date: 2022/06/12

## NAME

insert-newline - Insert new line at cursor position

## SYNOPSIS

*n* **insert-newline** (**C-o**)

## DESCRIPTION

**insert-newline** inserts *n* new lines at the current cursor position, but does not move the cursor. Any text following the cursor is moved to the newly created line.

**insert-newline** is the built in key binding for (**C-o**) the key binding is typically replaced with **open-line**(3) which behaves correctly by maintaining the cursor position.

## SEE ALSO

**newline**(2), **open-line**(3).

Ref: insert-newline(2) File: m2cmd097.2 Date: 2022/06/12

## NAME

insert-space - Insert space(s) into current buffer

## SYNOPSIS

*n* **insert-space**

## DESCRIPTION

**insert-space** inserts *n* spaces at the current cursor position, moving the cursor position.

## SEE ALSO

**insert-string**(2), **insert-tab**(2), **insert-newline**(2), **space-rectangle**(3).

Ref: insert-space(2) File: m2cmd075.2 Date: 2022/06/12

## NAME

insert-string - Insert character string into current buffer

## SYNOPSIS

*n* **insert-string** "*string*"

## DESCRIPTION

**insert-string** inserts a string *n* times into the current buffer, moving the cursor position.

**insert-string** allows text to be built in a buffer without reading it from a file. Some special escape characters are interpreted in the *string*, as follows:

\n - Enters a new line
\t - A tab character
\b - Backspace
\f - Form-feed
\\ - Literal backslash character '\'
\xXX - Hexadecimal value of character ASCII value

## NOTES

If a negative argument *n* is given to **insert-string** the given string is inserted 0-*n* times into the buffer. The behavior is the same as if a positive argument is given except that if the string contains a new-line character (ascii char 0x0a), rather than splitting the string over multiple lines the character is inserted into a single line of the buffer.

This is highly irregular and can break MicroEmacs (for example **undo**(2) will fail and could lead to a crash); the feature has been added to support bit `0x100` of the **shell-command**(2) process spawning group of commands. It's use outside of this context is to be avoided.

## SEE ALSO

**insert-file**(2), **insert-newline**(2), **insert-space**(2), **insert-tab**(2), **newline**(2).

Ref: insert-string(2) File: m2cmd076.2 Date: 2022/06/12

## NAME

insert-tab - Insert tab(s) into current buffer

## SYNOPSIS

*n* **insert-tab** (**C-i**)

## DESCRIPTION

**insert-tab** inserts *n* tab characters at the current cursor position, moving the cursor. The command is not affected by the **tab**(2m) mode as it always inserts literal tab characters.

## SEE ALSO

**insert-space**(2), **insert-string**(2), **insert-newline**(2), **tab**(2), **normal-tab**(3), **tab**(2m).

Ref: insert-tab(2) File: m2cmd156.2 Date: 2022/06/12

## NAME

ipipe-shell-command - Incremental pipe (non-suspending system call)
ipipe-kill - Kill a incremental pipe
ipipe-write - Write a string to an incremental pipe

## SYNOPSIS

*n* **ipipe-shell-command** "*command*" ["*buffer-name*"] ["*buffer-ipipe-command*"] (**esc back-slash**)
*n* **ipipe-write** "*string*"
*n* **ipipe-kill**

## PLATFORM

UNIX - *linux*, *macos*, *freebsd*, *hpux*, *sunos*.
Windows NT - *windows*.

## DESCRIPTION

**ipipe-shell-command** executes the given system command *command*, opening up a ***icommand*** buffer into which the results of the command execution are displayed. Unlike the **pipe-shell-command**(2), the user may continue editing during command execution. The command may be terminated by deleting the buffer or issuing a **ipipe-kill** command.

The argument *n* can be used to change the default behavior of pipe-shell-command described above, *n* is a bit based flag where:

**0x01** Enables the use of the default buffer name ***icommand*** (default). If this bit is clear the user must supply a buffer name *buffer-name*. The buffer naming allows another command to be started without effecting any other existing command buffer.

**0x02** Hides the output buffer, the default action pops up a window and displays the output buffer in a new window.

**0x04** Disable the use of the command-line processor to launch the program (MS Win32 versions only). By default the "*command*" is launched by executing the command:

```
%COMSPEC% /c command
```

Where `%COMSPEC%` is typically `command.com`. If this bit is set "*command*" is launched directly.

**0x08** Force the use of a pseudo terminal, i.e. a UNIX PTY or a Windows ConPTY, ignored when bit **0x2000** is set. When bits **0x08** and **0x2000** are both clear, pseudo terminals are used on UNIX systems, but not on Windows. A pseudo terminal cannot always be created, whether one was used can be determined from bit **0x02** of **$buffer-ipipe-flags**(5).

**0x10** Disable the command name mangling (MS Win32 versions only). By default any '/' characters found in the command name (the first argument only) are converted to '\' characters to make it Windows compliant.

**0x20** Displays the new process window (windows versions only), by default the window is hidden.

**0x40** Raw ipipe, disable text annotation at the start and end of the pipe output and force **wrap**(2m) mode off.

**0x80** Supply a function *buffer-ipipe-command* to be invoked when the pipe receives data and terminates. This is equivalent to setting the buffer variable **$buffer-ipipe**(5) but allows the pipe to be set when the ipipe is created. Setting the *$buffer-ipipe* on invocation allows the pipe exit status to be trapped properly.

**0x100** When set the command-line to be run is the first line of buffer *string*, this feature can be used to overcome the default 1024 character command-line limit. When a multiple-line command-line is required, a negative argument can be given to **insert-string**(2).

**0x200** When set forces **wrap**(2m) mode off minimising line wrapping.

**0x800** When the terminal is operating in UTF-8 mode (i.e. **$system**(5) bit `0x20000000` is set, auto-detected at startup from the locale), pipe output is automatically decoded from UTF-8: multi-byte sequences are converted to the nearest character in the current codepage, box-drawing characters are mapped to the MicroEmacs box-drawing set, and common typographic characters (curly quotes, em-dash, bullet etc.) are approximated with ASCII. Setting bit **0x800** suppresses this automatic decoding so that raw UTF-8 bytes are stored in the buffer instead. Bit **0x40** (raw mode) also suppresses UTF-8 decoding.

**0x1000** Enables support for ANSI color in the terminal processing, semantically converting foreground, background and style combinations to meth based scheme codes. The buffer's **$buffer-hilight**(5) variable must be set appropriately for the high-lighting to be recognisable.

**0x2000** Stop the use of a pseudo terminal, forcing the use of a plain anonymous pipe, this overrides bit **0x08**. Note that many interactive programs, such as a shell, do not work well without a pseudo terminal.

As some of these settings may be changed by MicroEmacs, e.g. when a pseudo terminal cannot be created, the way in which the pipe was actually created is recorded in the buffer variable **$buffer-ipipe-flags**(5), this may be used by a macro to determine how the process was launched and how its output was processed.

Many other macro commands (see **compile**(3), **grep**(3), etc.) use this command.

**ipipe-write** writes a string *string* to an open ipipe, *n* times.

**ipipe-kill** terminates an open ipipe and is automatically invoked when the ipipe buffer is deleted using **delete-buffer**(2) or when MicroEmacs is exited. The numeric argument *n* may be used to change the way in which the process is terminated, if a 0 argument is given an interrupt signal (kill request or nice-kill) is sent, writing a *Ctrl-C* to the *<stdin>* pipe on UNIX or sending *Ctrl-C* key events on Windows. This is likely to fail, particularly on Windows which does not have a robust signal interrupt system, however it is still typically bound to `C-c C-c` as Windows does not offer an alternative. On UNIX based systems `C-c C-c` is typically bound to ipipe-kill with a numeric argument of 2 which sends a *SIGINT*, see **signal**(7).

On UNIX platforms, if a +ve argument is given then only the top level process is sent a kill signal of the given value, e.g. the process would be terminated with a core dump

if a value of 3 was given. If a -ve argument is given the process and all its children, in the same process group, will be sent a 0-n kill signal. When no argument is given the default used is either -9, -SIGKILL, or -15, -SIGTERM, depending on the setting of bit 0x2000000 of **$system**(5).

On Windows, if no argument or a -ve argument is given **ipipe-kill** terminates the process and all non-detached sub-processes, whereas a +ve only terminates the top level process.

**ipipe-kill** with no argument is typically bound to C-c C-k.

## NOTES

On UNIX platforms the TERM environment variable of the new process is automatically set by MicroEmacs to a suitable value. When color is not being used, see bit **0x1000**, MicroEmacs uses the **tput**(1) system utility to determine whether vt100-nam is available and if so sets TERM to it, otherwise it sets TERM to vt100. When color is being used MicroEmacs determines whether terminal ansi is available, if so it set TERM to "ansi", otherwise it sets TERM to the default for non-color. This setting may be overridden by setting the user variable **%ipipe-term** or **%ipipe-color-term**, respectively, to the required value, e.g.:

```
set-variable %ipipe-term "TERM=vt100-nam"
```

Ipipe shells support a large sub-set of vt100 terminal commands, notable exceptions are full color (semantic suggestion only), and font support. Using the terminal type "vt100-nam" disables the use of auto-margins, which generally provides better support.

On platforms which do not support **ipipe-shell-command**, such as MS-DOS, executing **ipipe-shell-command** automatically invokes pipe-shell-command. Therefore macros may safely use ipipes without explicitly checking the platform type. **ipipe-shell-command** does not run reliably on **Windows 3.11** and **Windows 95**; **Windows NT** does support ipipes.

While the pipe command is running, mode **pipe**(2m) is enabled. Modes **lock**(2m) and **wrap**(2m) effect the output behavior of an **ipipe-shell-command**. When **wrap** mode is enabled text output is automatically wrapped at the width of the current window, when disabled (see arg **n** bits 0x40 and 0x200) lines will only wrap at (maximum length of a MicroEmacs variable - 2), currently 2046, so that the ipipe command can safely process the output. This length can be controlled by the **%ipipe-width** user variable.

## EXAMPLE

The following example is the **xgrep**(3) command macro which utilizes the **ipipe-shell-command**, diverting the output to a buffer called ***grep***.

```
define-macro xgrep
    !force set-variable #l0 @1
    !if &not $status
        set-variable #l0 @ml00 %xgrep-com
    !endif
    !if @?
        1 pipe-shell-command &cat %xgrep-com #l0 "*grep*" @mna
    !else
        1 ipipe-shell-command &cat %xgrep-com #l0 "*grep*" @mna
    !endif
!emacro
```

Note that if an argument is passed to **grep** then it uses pipe-shell-command instead. This is useful if another command is using **grep** which must finish before the calling command can continue, see **replace-all-string**(3) for an example.

## BUGS

On MicroSoft Windows platforms, **ipipe-shell-command** spawns the shell (e.g. `command.com`) with the appropriate command line to make it execute the given command. If the command to be run detaches from the shell and creates its own window, for example `me.exe`, **ipipe-kill** will only kill the shell, it will not kill the actual process, i.e. the `me.exe`.

On MicroSoft Windows platforms **ipipe-shell-command** does not work on Novell's Intranet Client v2.2 networked drives, version 2.5 does appear to work.

## SEE ALSO

**$buffer-ipipe**(5), **$buffer-ipipe-flags**(5), **$buffer-input**(5), **compile**(3), **grep**(3), **pipe-shell-command**(2), **replace-all-string**(3), **shell-command**(2), **pipe**(2m), **lock**(2m), **wrap**(2m).

Ref: ipipe-shell-command(2) File: m2cmd077.2 Date: 2026/07/16

## NAME

isearch-forward - Search forward incrementally (interactive)
isearch-backward - Search backwards incrementally (interactive)

## SYNOPSIS

*n* **isearch-forward** [ "*arg*" ] (**C-s**)
*n* **isearch-backward** [ "*arg*" ] (**C-r**)

## DESCRIPTION

**isearch-forward** provides an interactive search in the forward direction. This command is similar to **search-forward**(2), but it processes the search as each character of the input string is typed in. This allows the user to only use as many key-strokes as are needed to uniquely specify the string being searched. The string argument *arg* is not supplied when invoked interactively.

The follow keys can be used at the start of an incremental search only:

`C-s` - Search for last string.
`C-m` - Perform a search-forward instead.
`esc p`,
`esc n` - Scroll through history list etc (See **ml-bind-key**(2)).

Several control characters are active while isearching:

**C-s** or **C-x**
: Skip to the next occurrence of the current string

**C-r**
: Skip to the last occurrence of the current string

**C-h**
: Back up to the last match (possibly deleting the last character on the search string)

**C-w**
: Insert the next word into the search string.

**C-g**
: Abort the search, return to start.

**esc** or **C-m**
: End the search, stay here

**isearch-backward** is the same as **isearch-forward**, but it searches in the reverse direction.

For both commands when the end of the buffer is reached, an alarm is raised (bell etc.) a further search request (`C-s`) causes the search to commence from the start of the buffer.

The numeric argument *n* controls the behavior of **isearch-*** at the end of the buffer. When specified as `0` then the search exits when the start/end of the buffer is reached and the search is exhausted. When omitted (or specified as `1`) then the search recommences again at the start/end of the buffer, this is the default behavior.

When **isearch-*** is invoked from a macro then the string argument *arg* specifies the search string. **isearch** is primarily designed for for interactive use and is not typically used from within a macro. When embedded in a macro then it is typically invoked with the `0` argument so that the calling macro may handle the start/end of buffer condition. The macro **page-file**(3) uses **isearch-*** and performs exactly this operation, for a comprehensive example refer to macro file `pagefile.emf`.

## NOTES

The **ml-bind-key**(2) bindings are used.

The incremental search supports buffer modes **exact**(2m) and **magic**(2m).

## HISTORY

The numeric argument to **isearch-*** was first introduced into MicroEmacs '04.

## BUGS

Due to the dynamic nature of active **ipipe-shell-command**(2) buffers the search history cannot be stored in the same way (list of fixed locations). As a result the search history is stored as a list of searches which are not guaranteed to be consistent.

## SEE ALSO

**exact**(2m), **hunt-forward**(2), **magic**(2m), **ml-bind-key**(2), **narrow-search-forward**(3), **page-file**(3), **search-forward**(2).
Regular Expressions

Ref: isearch-forward(2) File: m2cmd078.2 Date: 2022/06/12

## NAME

kbd-macro-query - Query termination of keyboard macro

## SYNOPSIS

*[Definition]*
**kbd-macro-query** (**C-x q**)
*[Execution]*
**kbd-macro-query** "**y**"—"**n**"—"**C-g**"

## DESCRIPTION

**kbd-macro-query** queries the termination state of keyboard macro recording. If the command is executed during a keyboard macro definition, at that point during its execution the user is prompted as to whether to continue the macro execution. A reply of "**y**" continues the execution as normal, "**n**" stops execution at that point once, if executing the macro *n* times the macro will still executed a further *n-1* times. If the "`C-g`" abort command is entered then all keyboard macro execution is aborted, regardless of the number of repetitions.

## SEE ALSO

**start-kbd-macro**(2), **execute-kbd-macro**(2).

Ref: kbd-macro-query(2) File: m2cmd079.2 Date: 2022/06/12

## KEY BINDING NAMES

Every key which can be generated in MicroEmacs has a character string or name representation which can be used to bind and unbind the key to a command. The name of simple keys like "a" or "$" is simply the character, i.e. "a" and "$". Following is a list of other parts to a key name.

### Modify Keys

There are 3 modifying keys, Shift, Control and Alt, these are represented as "S-", "C-", "A-" respectively. For example the key "A-C-S-up" is generated when the up cursor key is pressed when Shift, Control and Alt keys where also pressed.

The Control and Alt modifiers are case insensitive so C-a is the same as C-A and C-S-a.

### Prefix Keys

Many binding are single stroke key sequences (e.g. "C-a" => beginning-of-line). However MicroEmacs has a **prefix**(2) command which can be used to bind up to 8 single stroke keys, turning them into two stroke keys; this greatly increasing the number of available bindable key sequences. For example **prefix 1** is bound to the escape character (esc), this allows key sequences like "esc a" to be used. Following is a list of prefixes and their default bindings

prefix 1 => esc
prefix 2 => C-x
prefix 3 => C-h
prefix 4 => C-c

### Special Keys

Following is a complete list of recognized keyboard key names, not all are able to be generated on every platform:

backspace, delete, down, end, esc, f1, f2, f3, f4, f5, f6, f7, f8, f9, f10, f11, f12, f13, f14, f15, f16, f17, f18, f19, f20, home, insert, kp-0, kp-1, kp-2, kp-3, kp-4, kp-5, kp-6, kp-7, kp-8, kp-9, kp-add, kp-begin, kp-decimal, kp-delete, kp-divide, kp-down, kp-end, kp-enter, kp-home, kp-insert, kp-left, kp-multiply, kp-page-down, kp-page-up, kp-right, kp-subtract, kp-up, left, page-down, page-up, return, right, space, tab, up

The name of any key can be obtained by using **describe-key**(2).

### Mouse Keys

Following is a list of mouse related keys:

**mouse-pick-1**, **mouse-pick-2**, **mouse-pick-3**, **mouse-pick-4**, **mouse-pick-5**
: These keys are generated when the user presses a mouse button, these key events are always created. On most systems button 1 is the left, 2 the middle and 3 the right mouse button. If the system only has a 2 button mouse then a mouse-pick-2 cannot be generated. The order of the buttons can be revered (i.e. 1 becomes right) and the number of buttons can be set using the **$system**(5) variable. Note that X-servers support up to 5 buttons and with the growing popularity of pilot 'wheel' mice,

the 4th and 5th button are often used to report wheel spin up and down events. The **translate-key**(2) command can be used to translate these buttons to the mouse wheel keys.

**mouse-drop-1**, **mouse-drop-2**, **mouse-drop-3**, **mouse-drop-4**, **mouse-drop-5**
: These keys are generated when the user release a mouse button, these key events are always created.

**mouse-move-1**, **mouse-move-2**, **mouse-move-3**, **mouse-move-4**, **mouse-move-5**, **mouse-move**
: These key events are generated when the user moves the mouse and are only if they are bound to a command. The key generated depends on whether a button is being held down by the user, if the user is pressing button 1 then a `mouse-move-1` key is created etc.

**mouse-time-1**, **mouse-time-2**, **mouse-time-3**, **mouse-time-5**, **mouse-time-5**, **mouse-time**
: These key events are generated only when they are bound to a command. They are pseudo keys created when the user hold the mouse buttons done for a period of time, see **Pseudo Keys** below for more information.

**mouse-wheel-up**, **mouse-wheel-down**
: Pilot mouse wheel events, generated when the wheel is spun up or down respectively.

### Modifier Keys

The Shift, Control and Alt modifier keys will also generate key input whenever pressed or released. The keys are however only generated if they are bound to a command. The key names are as follows:

**S-pick**, **S-drop**
: Shift modifier.

**C-pick**, **C-drop**
: Control modifier.

**A-pick**, **A-drop**
: Alt modifier.

  Note that the keys are only generated when another key is pressed, i.e. if the user presses and holds only the shift key, no "`S-pick`" key will be generated until another key, such as `down`, is also pressed. If the shift key is released before another key is pressed the event will not be reported.

### Pseudo Keys

Pseudo keys events cannot be directly created by the user, they are created internally by MicroEmacs. They are treated like normal keys to allow the user to handle the events properly themselves. Following is a complete list of the system generated pseudo keys:

**bell**
: The pseudo key is generated when the bell is rung.

**bp-start**, **bp-end**
: These keys are only define and used by *ncurses* enabled UNIX console versions. When pasting into a UNIX console it is advantageous for the console program to know the text entered is part of a block-text paste as it allows the editor to format the text correctly (for more information see **Wikipedia** page on *Bracketed paste*). **bp-start** is bound to **void**(2), however the key is use to trigger the start of a paste, **bp-end**

is use by the system to indicated the end of the paste. MicroEmacs converts the text in between into a kill buffer and as bp-end is bound to **yank**(2) it also yanks the text into the current buffer.

**callback**
The pseudo key when a **create-callback**(2) macro is executed, this allows the executed macro to know it was executed via a create-callback as **@cck**(4) will be set to this.

**find-file** The command bound to this pseudo key is execute whenever MicroEmacs cannot locate a required file, such as the main help file `me.ehf` or an **OpenSSL**(1) dynamic libraries. The command can get the name of the missing file from the **$result**(5) variable and either **!abort**(4) if the file cannot be located or return the full path to the file in **$result**. Care must be taken to avoid a loop, for instance if to attempt to locate a file the macro triggers an similar request to locate the same file MicroEmacs will spin.

**idle-pick**, **idle-drop**, **idle-time**
The commands bound to these keys are executed when the system becomes idle for a period of time. See help on **$idle-time**(5) for more information.

**mouse-time-1**, **mouse-time-2**, **mouse-time-3**, **mouse-time-4**, **mouse-time-5**, **mouse-time**
The command bound to these keys are executed when mouse button 1, 2, 3, 4, 5 or a combination are held bound for a period of time. See help on **$delay-time**(5) for more information.

**redraw** The command bound to this pseudo key is executed whenever the screen needs redrawing, by default it is bound to **screen-update**(2). If the user unbinds this key then **screen-update** is still called, but if the user binds it to a function which does not redraw the screen, such as **void**(2), then the screen will not be up-dated.

The command executed is always given an argument, a non-zero argument indicates a forced complete redraw, an argument of zero indicates that just an up-date is required.

### Alt Key

The **Alt Key** has special binding priorities defined as follows:

- Direct key binding (e.g. **A-b** executes file-browser)
- Main menu hot key (e.g. **A-f** opens the File menu)
- Meta key binding (e.g. **A-space** -> **esc space** -> set-mark)

If the `ALT` key is to be used strictly as the Emacs Meta key then the bindings for the menu should be over-ridden by *Direct Key Bindings* from the user configuration file i.e. to re-map the default MicroEmacs Alt key to equivalent `esc` keys then the following keys should be re-bound.

```
global-bind-key forward-word "A-f"      ; Over-ride File menu binding
:                                       ; For all of the other menu items.
:
global-bind-key backward-word "A-b"     ; Over-ride the file browser.
global-bind-key replace-string "A-r"    ; Over-ride tools binding.
```

This creates a higher priority binding which overrides the underlying default. The commands that are displaced would have to be re-bound to different keys if required.

## KEYBOARD MACROS

Keyboard macros do not store the name of keys, instead a more machine oriented format is used (usually in the form "\s??") these will work across platforms (assuming the key bindings are the same) but they may not work across different releases.

As a result it is advised that any long term macro should avoid named keys like `up` in favor of using a standard key binding such as `C-p`. See help on **execute-string**(2) for more information.

Ref: keyNames(2) File: m2cmd203.2 Date: 2025/09/12

## NAME

kill-line - Delete all characters to the end of the line

## SYNOPSIS

*n* **kill-line** (**C-k**)

## DESCRIPTION

**kill-line**, when used with no argument *n*, deletes all text from the cursor to the end of a line, the end of line character is NOT deleted. The deleted text is placed in the kill buffer, see **yank**(2) for more information on the kill buffer. If the line is blank then the whole line is deleted, including the end of line character.

If a positive argument *n* is supplied the specified number of lines are deleted. If a negative argument is specified the previous -*n* lines are deleted. If *n* is 0 the command has no effect.

## NOTES

If a line is accidentally removed then use yank to restore the text or use **undo**(2).
The Emacs Key binding for kill line ((**C-k**) ) may be re-bound to affect the behavior of the command.

```
; MicroEmacs binding. Bind to kill the end of line character
1 global-bind-key kill-line "C-k"

; GNU Emacs binding. Bind to leave the end of line character
  global-bind-key kill-line "C-k"
```

## HISTORY

The behavior of **kill-line** was modified in November 2004. **kill-line** now interprets the numeric argument *n* in both a positive and negative direction. The existing implementation did not operated in a negative direction, the negative argument controlled the kill buffer behavior of the line. The kill behavior is no longer an option to **kill-line** and the line(s) are always added to the kill buffer, control of the kill buffer is required then deletion should be performed with **kill-region**(2).

The behavior of the end of line kill used to be controlled by a **line**(2m) mode, this behavior may be replicated by modification of the binding as described in the previous section.

## SEE ALSO

**forward-kill-word**(2), **global-bind-key**(2), **kill-paragraph**(2), **kill-region**(2), **undo**(2), **yank**(2).

Ref: kill-line(2) File: m2cmd082.2 Date: 2022/06/12

## NAME

kill-paragraph - Delete a paragraph

## SYNOPSIS

*n* **kill-paragraph**

## DESCRIPTION

**kill-paragraph** deletes the next *n* paragraphs from the current cursor position, the paragraphs that are killed are added to the kill buffer. If the cursor is in the middle of a paragraph then the text from the current cursor position to the end (or start, depending on direction) of the paragraph is deleted.

If *n* is positive then the kill direction is forwards, if *n* is negative then the kill direction is backwards. If *n* is `0` (zero) the command simply returns. The default value for *n* is `+1`.

## DIAGNOSTICS

The following errors can be generated, in each case the command returns a FALSE status:

**[end of buffer]**
: The positive argument *n* is greater than the number of remaining paragraphs, all of the remaining paragraphs to the end of the buffer are still removed.

**[top of buffer]**
: The negative argument *n* is greater than the number of previous paragraphs, all of the previous paragraphs to the top of the buffer are still removed.

## NOTES

A paragraph is terminated by a blank line. All text residing between two blank lines is considered to be a paragraph - regardless of the text layout.

## HISTORY

The behavior of **kill-paragraph** was modified in November 2004 after which the numeric argument *n* was interpreted to control the number and direction of paragraphs to delete. Previous releases interpreted a negative value as a control to disable the deleted text from entering the kill buffer. If the existing behavior is required then **kill-region**(2) with a negative argument should be used to delete a region of marked text, **forward-paragraph**(2) and **backward-paragraph**(2) may be used to move to the appropriate kill position.

## SEE ALSO

**backward-paragraph**(2), **forward-paragraph**(2), **kill-region**(2), **undo**(2), **yank**(2).

Ref: kill-paragraph(2) File: m2cmd080.2 Date: 2026/02/28

## NAME

kill-rectangle - Delete a column of text
yank-rectangle - Insert a column of text

## SYNOPSIS

**kill-rectangle** (**esc C-w**)
*n* **yank-rectangle** (**esc C-y**)

## DESCRIPTION

**kill-rectangle** deletes a rectangle (or column) of text defined between the cursor and the set-mark position. The text between the mark column and the cursor column is removed from every line between the mark line and the cursor line inclusive and copied to the kill buffer. The delete text may then be extracted from the kill buffer using **yank**(2), **yank-rectangle** or **yank-rectangle-overwrite**(3).

The mark position may be ahead or behind the current cursor position. If the rectangle column boundary divides a tab character which spans multiple columns, the tab character is replaced with the equivalent number of spaces. Similarly if the boundary divides an unprintable character which is displayed using multiple characters (e.g. '^A' for character 0x01) then spaces are inserted before the character to move it to the right of the boundary.

**yank-rectangle** inserts the current kill buffer (which may or may not have been generated using **kill-rectangle**) into the current buffer in a column fashion. That is to say that the first line of text in the kill buffer is inserted into the current line of text in the current buffer from the current cursor column, the cursor is then moved the the next line and placed at the same column. The process is then repeated for the second line of text in the kill buffer etc.

## SEE ALSO

**copy-rectangle**(3), **copy-region**(2), **kill-region**(2), **reyank**(2), **set-mark**(2), **string-rectangle**(3), **undo**(2), **yank**(2), **yank-rectangle-overwrite**(3).

Ref: kill-rectangle(2) File: m2cmd159.2 Date: 2022/06/12

## NAME

kill-region - Delete all characters in the marked region

## SYNOPSIS

*n* **kill-region** (**C-w**)

## DESCRIPTION

**kill-region** deletes all characters from the cursor to the mark set with the **set-mark**(2) command. The characters removed are copied into the kill buffer and may be extracted using **yank**(2). If a numeric argument of 0 is given the command has no effect. If a negative argument is given the characters are not placed in the kill buffer, therefore the text is effectively lost (this does not effect the **undo**(2) operation).

The mark position may be ahead or behind the current cursor position.

Where the region contains a narrowed region, defined by **narrow-buffer**(2), then **kill-region** deletes all text contained within the narrow, this text is added to the kill buffer.

## USAGE

To move text from one place to another:

- Move to the beginning of the text you want to move.
- Set the mark there with the set-mark (**esc space**) command.
- Move the point (cursor) to the end of the text.
- Use the **kill-region** command to delete the region you just defined. The text will be saved in the kill buffer.
- Move the point to the place you want the text to appear.
- Use the yank (**C-y**) command to copy the text from the kill buffer to the current point.

Repeat the last two steps to insert further copies of the same text.

## NOTES

If a region is accidentally removed then yank the text back immediately or use **undo**(2).

Windowing systems such as X-Windows and Microsoft Windows utilize a global windowing kill buffer allowing data to be moved between windowing applications (*cut buffer* and *clipboard*, respectively). Within these environments MicroEmacs automatically interacts with the windowing systems kill buffer, the last MicroEmacs **kill-region** entry is immediately available for a paste operation into another windowing application.

## SEE ALSO

**backward-kill-word**(2), **copy-region**(2), **forward-kill-word**(2), **kill-line**(2), **kill-paragraph**(2), **kill-rectangle**(2), **narrow-buffer**(2), **reyank**(2), **set-mark**(2), **undo**(2), **yank**(2).

Ref: kill-region(2) File: m2cmd081.2 Date: 2022/06/12

## NAME

list-buffers - List all buffers and show their status

## SYNOPSIS

*n* **list-buffers** (**C-x C-b**)

## DESCRIPTION

**list-buffers** splits the current window and in one half brings up a list of all the buffers currently existing in the editor. The active modes, change flag, and active flag for each buffer is displayed. (The change flag is a * character if the buffer has been changed and not written out. The active flag is not an @ if the file had been specified on the command line, but has not been read in yet since nothing has switched to that buffer.)

The numerical argument *n* is a bitwise flag which determines the information that is displayed by the command, it is interpreted follows:

**0x01** The default, when the bit is set displays visible buffers only. When clear displays both visible and hidden buffers.

**0x02** When set shows buffer size information, including file size, memory size and undo size in bytes.

The buffer list has some special command keys associated with it which allow the state of the buffers to be edited from the buffer list, the editing allows buffers to be killed and saved to disk. The key codes are defined as follows:

**return** or **1** - Switch to buffer
Switch to buffer and make it the only buffer.

**space** or **2** - Move to buffer
Switch the buffer list window to that buffer.

**D** - toggle delete buffer
Flag buffer for deletion. A buffer scheduled for deletion is marked with a '**D**' in first column. This is not performed immediately, for the delete to be enacted the '**X**', execute command, must be given. The flag may be removed with another '**D**' or the '**U**' command.

**E** - toggle buffer edit mode
Toggle the buffer's **edit**(2m) mode, this command is executed immediately.

**S** - toggle save buffer
Flag buffer for saving. A buffer scheduled from saving is marked with a '**S**' in the second column. This is not performed immediately, for the buffer to be actually saved the **X**, execute command, must also be given. Note that a buffer may be marked for saving and deletion, the save operation is performed before the delete.

**U** - unmark buffer
Unmark the '**D**' and '**S**' flags on current line.

**V** - toggle buffer view mode
Toggle the buffer's **view**(2m) mode, this command is executed immediately.

**X** - execute

Execute all the '**D**' and '**S**' flags currently set. The **S**ave is enacted first.

For all but '**X**', the buffer selected is the buffer noted on the current cursor line. These keys are not remappable.

### NOTES

The '**D**' and '**S**' flags are stored in the **:buffer-info** buffer variable.

### SEE ALSO

**list-variables**(2), **list-commands**(2), **split-window-horizontally**(2), **edit**(2m), **view**(2m).

Ref: list-buffers(2) File: m2cmd083.2 Date: 2024/05/21

## NAME

list-commands - List available commands

## SYNOPSIS

*n* **list-commands** (**C-h c**)

## DESCRIPTION

**list-commands** constructs a list of all known built in commands and macros that are currently defined by MicroEmacs and presents a list of those commands in the buffer "***commands***". Each entry is formatted as:

**command ..... keyCode**

Where multiple keys are bound to the same command, then each of the *keyCode*'s is shown.

**list-commands** only displays visible commands, if a numeric argument *n* of 0 is given then all commands are displayed, irrespective of their hidden status.

**list-commands** is similar to **describe-bindings**(2) except that the commands are presented in alphabetical order (as opposed to key binding order).

## EXAMPLE

The following is an example of the output of **list-commands**:

```
backward-char .................. "C-b"
                                 "left"
backward-delete-char ........... "backspace"
                                 "S-backspace"
backward-delete-tab ............ "S-tab"
backward-kill-word ............. "esc backspace"
backward-line .................. "C-p"
                                 "up"
                                 "C-up"
backward-paragraph ............. "esc ["
                                 "esc p"
backward-word .................. "esc b"
                                 "C-left"
beginning-of-buffer ............ "esc <"
                                 "home"
beginning-of-line .............. "C-a"
buffer-bind-key
buffer-info .................... "C-x ="
buffer-mode .................... "esc ~"
                                 "C-x m"
                                 "insert"
buffer-unbind-key
:
:
```

## SEE ALSO

**command-apropos**(2), **describe-bindings**(2), **list-variables**(2).

## NAME

list-registry - Display the registry in a buffer

## SYNOPSIS

*n* **list-registry** ["*path*"] ["*buffer*"]

## DESCRIPTION

**list-registry** lists the contents of the registry in the a buffer in a hierarchical format. The key name and any associated string is optionally shown as a hierarchical tree, the resultant information that is displayed is controlled by the numeric argument *n* and mode of the registry node.

The registry listing is generated in the buffer "`*registry*`" by default, the buffer may be modified with the *buffer* argument.

The numerical argument *n* is interpreted as a bit-mask as follows:

*1* If bit `0x1` is clear then the listing buffer is specified by the argument *buffer*. The default (when set) used buffer `*registry*`.

*2* If bit `0x2` is set the path of the root of the registry is defined by argument *path*. The default when clear is to list the whole registry.

The visibility of the node is controlled by the attributes assigned by **mark-registry**(2). The node value may be hidden (modes `!`, `i` and `h`) and are not displayed in the resultant listing.

## HISTORY

The numeric argument to **list-registry** with associated parameters and extended hidden modes were added in January 2005.

## SEE ALSO

**copy-registry**(3), **delete-registry**(2), **mark-registry**(2), **read-registry**(2), **save-session**(3), **save-registry**(2), **set-registry**(2), **&reg**(4), **erf**(8).

Ref: list-registry(2) File: m2cmd025.2 Date: 2022/06/12

## NAME

list-variables - List defined variables

## SYNOPSIS

*n* **list-variables** (**C-h v**)

## DESCRIPTION

**list-variables** pops up a window with a list of all register, buffer, user and global variables with their current setting. The variables are shown for the current buffer from which the command was invoked

The numerical argument *n* is a bitwise flag which determines the information that is displayed by the command, it is interpreted follows:

**0x01** The default, when the bit is set displays the variables of only the current buffer and no command variables. When clear it displays the variables of all buffers and commands/macros (including hidden).

**list-variables** provides a good alternative to **describe-variable**(2) where the value of multiple variables is to be interrogated.

The default output is displayed in four sections:

**Register variables**
The current settings of the global register variables ('**#**' prefix).

**Buffer Variables**
The current setting of the buffer variables ('**:**' prefix). This variables relate to the current buffer from which the command was invoked.

**System Variables**
The current settings of the system variables ('**$**' prefix).

**Global Variables**
The current setting of the global variables ('**%**' prefix).

## EXAMPLE

An example output from **list-variables** is shown below:

```
Register variables:

    #g0 ........................... "29"
    #g1 ........................... ""
    #g2 ........................... "ERROR"
    :
    :
    #g8 ........................... "ERROR"
    #g9 ........................... "ERROR"

Buffer [m2cmd086.2] variables:


System variables:

    $auto-time .................... "300"
    $buffer-bhook ................. "bhook-nroff"
    $buffer-bname ................. "m2cmd086.2"
```

```
    $buffer-ehook ................. "ehook-nroff"
    $buffer-fhook ................. "fhook-nroff"
    $buffer-fmod .................. "040"
    $buffer-fname ................. "d:/emacs/doc/m2cmd086.2"
    $buffer-hilight ............... "3"
    :
    :
    $window-width ................. "80"
    $window-x-scroll .............. "0"
    $window-xcl-scroll ............ "0"
    $window-y-scroll .............. "52"

Global variables:

    %company-name ................. "JASSPA"
    %compile-com .................. "build "
    %diff-com ..................... "diff -c -w"
    :
    :
    %tag-file ..................... "tags"
    %tag-option ................... "rcemsv"
    %xgrep-com .................... "grep -n "
```

## SEE ALSO

**describe-variable**(2), **list-commands**(2).

Ref: list-variables(2) File: m2cmd086.2 Date: 2022/06/12

## NAME

major-mode - Declare or set a buffer major mode

## SYNOPSIS

**major-mode** *id*
*-1* **major-mode**
*0* **major-mode** *id*
*n* **major-mode** *id* [”*cid-label*”] [”*extension-list*”] [”*full-name*”] [*cid-line-count* ”*cid-regex*”] [”*macro1-name*” ”*macro2-name*” ...]
*16* **major-mode** *id* [*cid-line-count* ”*cid-regex*”]

## DESCRIPTION

The **major-mode** command is used to define the main content *type* of a buffer, such as `C` or `HTML` etc. The command changes the *major-mode* assignment of the current buffer, defines or modifies the *major-mode* properties i.e. the file extension list etc.

When the *major-mode* of a buffer is assigned, the buffer variables **$buffer-major-mode**(5) and **$buffer-fhook**(5) are updated and the **$buffer-fhook** macro is executed. The **$buffer-fhook** macro (and hook file that it is defined in) configure the major mode features such as hilighting, indentation rules, etc.

**major-mode**, with no argument *n*, prompts for a new major mode *id* and is immediately applied. If *id* matches the buffers current *major-mode* identity then the command prints the full name of the current *major-mode*, the content label (if enabled) and name of the hook file.

When a argument of *-1* is given, **major-mode** reevaluates and assigned the best major mode for the current buffer. This will be based on the buffer's content, file name or buffer name if it has no file name. Note that when the **$buffer-fhook** is executed for a buffer without an associated file, it is only given the *new file* flag if the buffer is empty, this avoids inserting new file templates into buffers with content.

**major-mode**, with a positive numeric argument *n* modifies the major mode settings, the *id* argument must be supplied and the effect of the numeric argument *n* bits are as follows:

**0x01** When set, the *major-mode* is created if it does not already exist. An error is returned if the bit is clear and the major-mode *id* does not exist.

**0x02** When set, the *content-ID label* of the *major-mode* is assigned the argument value *cid-label*. If *cid-label* is an empty string (i.e. ””) the feature is disabled.

When clear, the *content-ID label* remains unchanged unless the call is creating the *major-mode*, in this case, the *cid-label* is set to the same value as the *id*.

Refer to the **Notes** section below for more information on the use of the *content-ID label*.

**0x04** When set, the file extensions associated with the major-mode are set to *extension-list*. *extension-list* is a space separated list of *file endings* (as opposed to true file extensions) and are usually specified with the extension separator. For example, the extension list ”`.c .h`” maybe used for C major-mode to associate files ending in `.c` or `.h`.

A forward slash ('/') character at the start of an extension forces the match of the whole file name, for example, the Unix **info** system often has directories with a 'dir' file, an extension-list value of ".info /dir" matches any file ending in .info and a file with the name dir.

A caret ('^') character at the start of an extension matches the start of the filename with the extension.

If *extension-list* is the empty string then the feature is disable.

When bit **0x04** is clear, the value remains unchanged unless this is the initial creation call, in which case the value is set to "*.id*".

Extension-list values are not concatenated when the *extension-list* is redefined, the full list of extensions must always be provided. The extension comparison is always case insensitive to ensure consistent behaviour across all platforms.

**0x08** When set, the full name of the major-mode is set to *full-name*. When this bit is clear the value remains unchanged unless the call is also creating the major-mode, when the name is set to the capitalised value of **cid-label**, if set, or **id**. The name is forced to uppercase if the value does not contain a vowel e.g. HTML.

**0x10** When set, the *cid-line-count* and *cid-regex* parameters are used to define a content based identifier which recognises the file type from the contents of the file. A file to be identified must have content within the first *cid-line-count* non-blank lines matching the *cid-regex*. If the line count *cid-line-count* is positive the regular expression search is case sensitive, if the line count is negative, the count is negated and the search is case insensitive.

When bit **0x10** is used on its own, a special 'regex replace' form of the *id* can be used, see the **Notes** section below for more information.

The order the major-modes are defined is used when there is a conflict in assignment, for example, both C and C++ may use the **.h** extension. Priority is given to the **last** defined major-mode, i.e. the first defined major-mode has the lowest priority, ensuring that any user created major-modes have the highest priority. The order may be manipulated by simply redefining the major-mode with an argument of 1. For example, adding the following line to the end a user macro file would make the shell script major-mode the highest priority:

```
1 major-mode "sh"
```

If a numerical argument of 0 is given to **major-mode** the command uses the variable **$result**(5) to return information about the given *id* major-mode in the following MicroEmacs list format:

```
\b<id>\b<flags>\b<cid-label>\b<extension-list>\b<full-name>\b
```

Where '\b' is the list value separator character. The <*flags*> member is an integer interpreted as a bit flag with the following values:

**0x01** If set, initialisation of the major-mode has been attempted, if bit **0x02** is also set then the attempt was successful and the major-mode would have been applied to a buffer, if bit **0x02** is clear then MicroEmacs failed to locate the major-mode fhook-<*id*> macro and aborted.

**0x02** If set, the major-mode has been successfully initialised and the `fhook-<id>` macro was located.

**0x04** If set, the major-mode has an `ftest-<id>` macro, this macro will be invoked to resolve major-mode assignment conflicts based solely on the file extension. This bit is always clear until the hook has successfully initialised.

**0x08** If set, the major-mode has a `bhook-<id>` macro, see **$buffer-bhook**(5). This bit is always clear until the hook has successfully initialised.

**0x10** If set, the major-mode has an `ehook-<id>` macro, see **$buffer-ehook**(5). This bit is always clear until the hook has successfully initialised.

**0x20** If set, the major-mode has a `dhook-<id>` macro, see **$buffer-dhook**(5). This bit is always clear until the hook has successfully initialised.

When **major-mode** is called with a numeric argument of 16, a special 'regex replace' style of *id* may be used, for example:

```
16 major-mode "\\1" -2 "-[*!]- *\\(\\m+\\).*?-[*!]-"
```

In this form the *id* string must be a '\' character followed by a digit between `0` and `7` which must refer to a group within the *cid-regex*. When a match is found, the value of the group is compared with all major-mode content-id labels (*cid-label*) and if a match is found that mode is applied. This has the major advantage of being able to support all major-modes in a single regex search, greatly reducing the performance impact of content based identification.

## NOTES

The major-mode has a very specific order of event execution which is defined in the following paragraphs:

The common major-modes must be defined first and this occurs in the main **me.emf** macro file, user and company created major-modes are typically defined in the user and company macro files. The definitions at this stage are minimal to enable MicroEmacs to start-up as quickly as possible, a single call to **major-mode** will usually suffice.

When the user loads a file MicroEmacs loops through all of the major-modes, in order, checking every defined *content-id* from the start of the first non-blank line of the loading file. If a match is found in the buffer then the major-mode is initialised (if not already initialised - see below) and assigned to the buffer.

If no content-id based match was found, MicroEmacs loops through all major-modes, in order, checking every defined extension list. If a match is found the major-mode is initialised, if required, and checked for the existence of a `ftest-<id>` macro. If the macro exists it is executed and the return status assessed, if there was no match then the buffer is assigned the major mode of the file extension. If the buffer content matches the `ftest-<id>` type the major-mode is immediately assigned and the search halted, otherwise the probability of the buffer being of that type is stored and the search is continued. If, by the end of the search, no definitive match has been found, the most likely major-mode is assigned, or the `default` major-mode where no match was found.

If a major-mode has been assigned, the major-modes **$buffer-fhook**(5) function (named `fhook-<id>`) is executed within the current buffer. The hook macros configure settings such as the hilighting, indentation schemes, buffer based binding and modes etc.

A major-mode is deemed initialised if the main `fhook-<id>` macro is defined, if it is not, MicroEmacs attempts to locate and execute the associated hook file, `hk<id>.emf`. If the file is not found or, having executed the file, the main `fhook-<id>` macro is still not defined the initialisation is considered failed and the major-mode becomes unusable.

### Migration

In January 2026 the existing command **add-file-hook** was replaced with **major-mode** with the aim to consolidating the major-mode concept and improving compatibility with Emacs. As a result, all existing calls to **add-file-hook** need to be migrated to **major-mode**.

Calls to **add-file-hook** take one of two different forms:

File Extension Recognition

Calls without a numeric argument were used to bind a list of extensions to a file-hook macro. To migrate these definitions, first determine if the hook is a standard system hook by locating the `hk<id>.emf` file and if so whether the extensions are now supported by checking the relevant call to **major-mode** in the current `me.emf`. If the hook is a system one and all extensions are listed then the existing call to add-file-hook maybe removed.

If the hook is a system one but not all extensions are listed, replace the add-file-hook line with one like the following:

```
4 major-mode "<id>" "<combined-full-extension-list>"
```

Where <id> is taken from the hook macro name which should have the form `fhook-<id>`. If it is not a system hook, replace the line with one like the following:

```
5 major-mode "<id>" "<extension-list>"
```

If the hook file contains a user friendly name for the hook, e.g. contains a line like the following:

```
set-variable .fhook-<id>.name "<File-type-name>"
```

Then remove this line from the hook file and change the call to major-mode to:

```
13 major-mode "<id>" "<extension-list>" "<File-type-name>"
```

File Content Recognition

Calls with a numeric argument to add-file-hook were used to define a content based identifier, given in the form of a regex search pattern. Many of these *content-IDs* take the form of "`-*- <id> -*-`" or "`-!- <id> -!-`", these are generally no longer required as the new major-mode *cid-label* feature automatically performs the same function. If the label used in the content-ID is something other than <*id*> then the call to add-file-line can usually be replaced with:

```
3 major-mode "<id>" "<cid-label>"
```

If this is not a system hook then this can be combined with an extension-based major-mode definition, i.e. a single call to major-mode like one of the following:

```
7  major-mode "<id>" "<cid-label>" "<extension-list>"
15 major-mode "<id>" "<cid-label>" "<extension-list>" "<File-type-name>"
```

However, if there is more than one label or the content-ID does not take this form then replace the call to add-file-hook with a line of the following form:

```
17 major-mode "<id>" n "<regex>"
```

Where <*id*> is taken from the hook macro name which should have the form `fhook-`<*id*>, *n* is the numeric argument given and <*regex*> is the first argument given. Again, if this is not a system hook this line can be combined with the major-mode definition call unless there are more than one.

## SEE ALSO

File Hooks, Language Templates, **$buffer-major-mode**(5), **$buffer-fhook**(5), **define-macro-file**(2).

Ref: major-mode(2) File: m2cmd164.2 Date: 2026/08/01

## NAME

mark-registry - Modify the operating mode of a registry node

## SYNOPSIS

*n* **mark-registry** "*root*" "*mode*"

## DESCRIPTION

**mark-registry** modifies the *mode* of a registry node *root*. If an argument *n* is supplied then the *n*th register node down from **root** (as viewed from **list-registry**(2) output) is modified instead. The *mode* is string specifying the modes, each mode is represented by a character. Lower case characters add a mode, upper case characters delete a mode. The modes are defined as:

**?** - Query Name
: Returns the full name, including path, of the given registry node in the variable **$result**(5). This does not alter the registry.

**!** - Hide Value
: Hides the value of the given registry node, i.e. the node is displayed but its value will not be displayed in the output of **list-registry**(2). Once set, this mode cannot be removed.

**a** - Autosave
: Automatically saves the registry when it is deleted or unloaded from the registry. The user is not prompted for a save.

**b** - Backup
: Automatically performs a backup of the registry file whenever a save operation is performed.

**c** - Create
: If the registry file cannot be loaded then the *root* node is created and the invocation succeeds. If this mode is omitted then the call fails if the *file* cannot be found.

**d** - Discard
: Marks the registry as discardable. This is typically used for registries that are not saved.

**f** - File
: The registry node is marked as a file root, the value must be set to the registry file name.

**g** - Get Modes
: Returns the list of modes currently set on the given registry node in the variable **$result**(5). This does not alter the registry.

**h** - Hierarchy Hidden
: The registry node is marked as *Hidden*, i.e. its children will not be shown in **list-registry**(2) output. A node is marked has hidden when it is folded up.

**i** - Invisible
: The registry node is not displayed in the **list-registry**(2) output.

**m** - Merge
: Marks the registry to enable merging, see **read-registry**(2).

**p** - Password Encryption

Marks the registry node for encryption using a manually entered password, node must be a file root (see **f** above). The registry file is encrypted when saved with **save-registry**(2) using a user entered password. This can be used in conjunction with flag **y** for increased security. The password must be entered on all read and write operations.

**r** - Reload

Marks the registry to enable reload, see **read-registry**(2).

**u** - Updated

Marks the registry as modified. The modified bit is removed when the registry file is saved. If the modified bit is applied to a registry node the user will be prompted to save the registry when it is deleted (or it will be automatically saved when the *Autosave* mode is used).

**y** - Auto-Encryption

Marks the registry node for encryption using an auto-generated encryption key, node must be a file root (see **f** above). The registry file is encrypted when saved with **save-registry**(2) using an auto-generated password. This can be used in conjunction with flag **p** for increased security. Do not rename the physical file as the file name is used as part of the encryption key.

Multiple modes may be applied.

## EXAMPLE

A history registry can be hidden with the following invocation:

```
mark-registry "/history" "h"
```

It could then be made visible again using:

```
mark-registry "/history" "H"
```

## BUGS

At exit only registry nodes attached to the root are saved.

## DIAGNOSTICS

**mark-registry** fails if *root* does not exist.

## SEE ALSO

**copy-registry**(3), **delete-registry**(2), **find-registry**(2), **list-registry**(2), **read-registry**(2), **save-registry**(2), **set-registry**(2), **&reg**(4), **erf**(8).

Ref: mark-registry(2) File: m2cmd154.2 Date: 2025/02/10

## NAME

ml-bind-key - Create key binding for message line
ml-unbind-key - Remove key binding from message line

## SYNOPSIS

*n* **ml-bind-key** "*command*" "*key*"
*n* **ml-unbind-key** "*key*"

## DESCRIPTION

**ml-bind-key** creates a key binding local to the message line input buffer. There are several commands that can be used in message line input, each command is associated with a main buffer editing command and inherits all that commands global bindings, i.e. moving forward 1 character is associated with the command **forward-char**(2) and thus inherits the binding `C-f` (as well as any other like the right cursor key). The following is a list of available commands, what they do and their associated commands

### Cursor Movement

- move backwards 1 character, command: backward-char (**C-b**, **left**)
- move forwards 1 character, command: forward-char (**C-f**, **right**)
- move backwards 1 word, command: backward-word (**esc b**)
- move forwards 1 word, command: forward-word (**esc f**)
- move to beginning of buffer, command: beginning-of-line (**C-a**, **home**)
- move to the end of buffer, command: end-of-line (**C-e**, **end**)
- move to beginning of buffer, command: beginning-of-buffer (**esc <**)
- move to the end of buffer, command: end-of-buffer (**esc >**)

### Input

- Quote a character, command: quote-char (**C-q**)
- Yank kill buffer into message line, command: yank (**C-y**)
- insert current buffers current line into the buffer, command: insert-newline (**C-o**)
- insert current buffers file name into the buffer, command: insert-file-name (**C-x C-y**)
- insert current buffer name into the buffer, command: -1 insert-file-name (**C-x y**)
- get the previous item in the yank buffer, command: reyank (**esc y**)

### Deletion

- copy marked region, command: copy-region (**esc w**)
- delete marked region, command: kill-region ( **cw**)

- delete backward 1 character, command: backward-delete-char (**C-h**, **backspace**)
- delete forward 1 character, command: forward-delete-char (**C-d**, **delete**)
- delete forward word, command: forward-kill-word (**esc d**)
- delete backward word, command: backward-kill-word (**esc backspace**)
- kill text from current position to end of line, the text is placed in the kill buffer. Command: kill-line (**C-k**).
- erase whole line, the text is placed in the kill buffer, command kill-region (**C-w**). Note that in incremental searches this is used to add the current word to the search string.

### History

MicroEmacs stores the last 20 entries of each kind (command, buffer, file, search and general which is also saved in the history file so the state of the history is retained when next loaded. The following commands can be used to manipulate the history.

- next history list entry (loop through history), command: forward-paragraph (**esc n**, **C-n**, **down**)
- previous history list entry, command: backward-paragraph (**esc p**, **C-p**, **up**)

### Completion

When entering a command, file, buffer or a mode name MicroEmacs creates a list of possible completions the following operations can be performed on this list.

- expand. This completes the given input until the first ambiguous character, command: a space (' ') or tab (**C-i**).
- expand to the previous completion (loops through the completion list), command: scroll-next-window-up (**esc C-z**)
- expand to the next completion (loops through the completion list), command: scroll-next-window-down (**esc C-v**)
- create a listing of all completions, command: a double expansion, i.e. 2 spaces or tabs. The first expands and the second creates the list.
- page up the completion list buffer, scroll-up (**C-z**, **page-up**)
- page down the completion list buffer, scroll-down (**C-v**, **page-down**)

### Miscellaneous

- abort input, returning failure to the input, abort-command (**C-g**)
- re-fresh the message line, command: recenter (**C-l**)
- finish input, command newline (**C-m**, **return**)
- set mark, command: set-mark (**C-space**)
- exchange point and mark, command: exchange-point-and-mark

- transpose previous character with current character, command: transpose-chars (**C-t**)
- capitalize the next word, command: capitalize-word (**esc c**)
- Turn the whole of the next word to upper case letters, command: upper-case-word (**esc u**)
- Turn the whole of the next word to lower case letters, command: lower-case-word (**esc l**)
- Cursor to mouse position, command: set-cursor-to-mouse (mouse select in window)
- Empty function, command: void

**ml-unbind-key** unbinds a user created message line key binding, this command effects only the message line key bindings. If a negative argument is given to **ml-unbind-key** then all message line bindings are removed.

## EXAMPLE

If expansion was required on the **esc esc** key binding then use the following:

```
ml-bind-key tab "esc esc"
```

## NOTES

The prefix commands cannot be rebound with this command.

Command key response time will linearly increase with each local binding.

## SEE ALSO

**global-bind-key**(2), **buffer-bind-key**(2), **describe-bindings**(2), **osd-bind-key**(2), **global-unbind-key**(2).

Ref: ml-bind-key(2) File: m2cmd085.2 Date: 2026/06/19

## NAME

ml-write - Write message on message line

## SYNOPSIS

*0* **ml-write**
*n* **ml-write** "*message*"

## DESCRIPTION

**ml-write** writes the argument "*message*" to the message line. If a positive numeric argument *n* is specified then the message is displayed with a *n* milliseconds uninterruptible delay allowing the user time to see the message.

The arguments to **ml-write** are defined as follows:

*No argument*
: The default form, writes a *message* to the command line with no delay.

*+ve*
: Displays the *message* on the message line and waits *n* milliseconds before proceeding with the next command.

`0`
: the *message* argument is not required and the command clears the message line. This may be used to clear any previous message from the last command(s) in a script and is equivalent to the old MicroEmacs command **ml-clear**.

*-ve*
: If negative, the value is negated and used as a bitwise flag whose bits are defined as follows:

  `1`
  : If running in piped mode (see command-line option **-p**) the *message* is written to *standard-out* as per `4` below, otherwise it is written to the message line as normal. This allows the same macro code to run correctly in both environments.

  `2`
  : If running in piped mode the *message* is written to *standard-error* as per `4` below, otherwise it is written to the message line as normal.

  `4`
  : Force the *message* to be written to *standard-out* if bit `1` is also set, or *standard-error* otherwise.

  `8`
  : When set the function will return an `ABORT`, typically halting the execution of the current macro.

  `16`
  : When set the function waits for the user to press a key, this can be used to ensure the user will see the message.

## NOTES

If the argument value is not `-1`, `-2`, `-4` or `-5` calls to **ml-write** are ignored in standard MicroEmacs piped mode (command-line option **-p**). They are printed to *standard error* when debug piped mode is used (command-line option **-P**, see **me**(1)), making the debugging and maintenance of macro based piped processing easier.

In previous releases the command **ml-clear** was used to clear the message line, this command was replaced by `0 ml-write` in December 2004.

## EXAMPLE

The following call displays a message on the message-line with a 2 second pause:

```
2000 ml-write "Hello World!"
```

The following **start-up**(3) macro may be used to add a header and footer line to a file piped to *standard in*, writing the output to *standard out*.

```
define-macro start-up
    find-buffer "*stdin*"
    ml-write &spr "DEBUG: Read %d lines from stdin" $window-eline
    ml-write &spr "DEBUG: About to write header"
    -1 ml-write "<<<HEADER>>>"
    ml-write &spr "DEBUG: About to write body"
    0 save-buffer
    ml-write &spr "DEBUG: About to write footer"
    -1 ml-write "<<<FOOTER>>>"
    -2 ml-write &spr "%d lines written to stdout" &add $window-eline 2
    quick-exit
!emacro
```

To run this example save this macro in a new macro file (such as `mlwrite_test.emf`) and in a shell run:

```
cat input_file | me -n -p @mlwrite_test - > output_file
```

Note the `DEBUG` lines will be printed if **-P** option is used. Also see example given for **command-wait**(2).

## SEE ALSO

**command-wait**(2), **create-callback**(2), **me**(1).

## NAME

name-kbd-macro - Assign a name to the last keyboard macro

## SYNOPSIS

**name-kbd-macro** "*command*"

## DESCRIPTION

**name-kbd-macro** labels the last defined keyboard macro with the given *command* name. The command name must be either unique or the name of an existing macro. A keyboard macro is deleted when another keyboard macro is defined, but when named, it is preserved. A named keyboard macro can also be bound to its own command key sequence, and may be inserted into a buffer enabling it to be saved and thus re-loaded and re-used at a later date.

## SEE ALSO

**execute-file**(2), **execute-kbd-macro**(2), **global-bind-key**(2), **insert-macro**(2), **start-kbd-macro**(2).

Ref: name-kbd-macro(2) File: m2cmd089.2 Date: 2022/06/12

## NAME

narrow-buffer - Hide buffer lines

## SYNOPSIS

*n* **narrow-buffer**
n **narrow-buffer** [*line-scheme*] [”*out-markup-line*” —
”*to-top-markup-line*” ”*to-bottom-markup-line*”] [”*markup-command*”]

## DESCRIPTION

The **narrow-buffer** command allows lines in a buffer to be hidden (effectively folded up) the scope of the narrow is defined by a region. A region is the text located between the **point** (the current cursor position) and the **mark** defined by set-mark.

**narrow-buffer** supports two types of information hiding. A simple *Narrow* or *Narrow Out* hides lines of the region from display and optionally inserts a new line, called a mark-up line. The mark-up line allows the narrow position to be annotated with a line scheme color and annotation text providing a visual representation of the narrow i.e.

```
Before Narrow Out           After Narrow Out
_____________________________________________________________

... buffer text ...         ... buffer text ..
<region start>               <Optional Narrow Out line mark-up>
.... region text ...        ... more buffer text ...
<region end>
... more buffer text ...
```

The second narrow form is a *Narrow To* hides text around the region leaving the region displayed only. A *Narrow Out* will optionally show a mark-up line at both the top and bottom of the region. The mark-up line allows the *Narrow To* position to be annotated in the buffer with a line scheme color and annotation text providing a visual representation of the narrow i.e.

```
Before Narrow To            After Narrow To
_____________________________________________________________

... buffer text ...         <Optional Narrow Out top mark-up>
<region start>               <region start>
.... region text ...        .... region text ...
<region end>                 <region end>
... more buffer text ...    <Optional Narrow Out bottom mark-up>
```

Once a narrow has been created and information has been hidden **narrow-buffer** may be used to subsequently reveal the hidden text on a line or complete buffer basis.

Narrowed regions do not delete the text contained within the buffer the information is simply hidden. Saving the buffer using **save-buffer**(2), by default, writes both the displayed and hidden information the command may be optionally invoked to save only the visible representation.

While information is hidden in a narrow **copy-region**(2) and **kill-region**(2) operate as expected. If a narrow is copied then all of the information that is contained within the narrow is also copied to the new location. Similarly killing or deleting a narrow removes all of the information that is contained within the narrow.

**narrow-region** performs the low-level information hiding and is used from the macro command language, the command is not generally invoked directly by the user. Within the macro environment the location of narrows in any buffer may be determined using the line variable **$line-flags**(5). This variable allows the narrow state of the current line in a buffer to be determined. The narrow state of a buffer may be determined with **narrow**(2m). When a narrow is

created the buffer mode **narrow**(2m) is automatically set, when the last narrow is removed the mode is deleted.

The effect of **narrow-buffer** depends on the given argument and is defined as follows:

`1` Removes all narrows in the current buffer (the default), it aborts if there are no narrows in the buffer.

`2` Removes the current line's narrow, it aborts if there is no narrow at the current line.

`3` Narrow to region hides all but the lines of text in the current buffer from the mark position to the current cursor position, effectively 'narrowing' the buffer to the remaining text.

`4` Narrow out region hides the lines of text in the current buffer from the mark position to the current cursor position. This is the inverse effect of argument **3**.

The second form of the command takes a variable number of arguments which are determined by the value of *n* when bits in the range `0x0ff0` are defined. These bits are interpreted as a bit-mask and determine how the narrow is to be visually displayed to the user. When these bits are set then a mark-up line is created.

`0x0010` - *line-scheme*
Bit `0x0010` specifies the line **hilight**(2) color scheme to be used on the narrow line when it is an open or closed narrow. The scheme colors used are defined by the scheme variables `scheme.fold-open` and `scheme.fold-closed`.

`0x0020` - "*out-markup-line*" — "*to-top-markup-line*" "*to-bottom-markup-line*"
Bit `0x0020` specifies that mark-up text is used, the text mark up is passed into the call. The number of arguments is determined by the type of markup operation (*to*/*out*). The arguments are placed on the mark-up lines.

`0x0040` - "*markup-command*"
Bit `0x0040` specifies that a mark-up line is used to represent the narrow region. When the narrow region is created then the *markup-command* macro is executed to annotate the narrowed region. The macro is invoked when the narrow is created and is passed a numeric argument *n* which identifies the operation of the narrow. The numeric argument to the mark-up macro is defined below, generally a positive argument means that a narrow is being added and negative means that the narrow is being removed.

`1` - This is a *Narrow Out* line.
`2` - This is the top of a *Narrow To*.
`3` - This is the bottom of a *Narrow To*.

When the macro is executed then the current line is the start of the narrow region, the text of the line is available to the macro via **@wl**(4). The *markup-command* aborts, using **!abort**(4) if the mark-up line is to be removed, otherwise returns a mark-up line in the variable **$result**(5).

## EXAMPLE

For example, if the buffer contains the following text:

```
1 Richmond
2 Lafayette
3 Bloomington
4 Indianapolis
5 Gary
6
```

If the mark is on line 2 and the current point is on line 4, executing:

```
4 narrow-buffer
```

Creates one narrow, narrowing out lines 2 and 3, line 4 becomes the narrow anchor line. When the narrow is removed lines 2 and 3 will be inserted before line 4. The buffer will become:

```
1 Richmond
4 Indianapolis
5 Gary
```

If instead the following command was executed:

```
3 narrow-buffer
```

Two narrows are created, the first narrowing out lines 4 and 5 (line 6, the last line, being the anchor line) the second narrowing out line 1 (line 2 being the anchor line). The buffer will become:

```
2 Lafayette
3 Bloomington
6
```

Executing **narrow-buffer** with an argument of **2** will only work on the anchor lines, namely 4 in the first example and 2 and 6 in the second.

### Marking Up

A pseudo line may be created by **narrow-buffer** which is hilighted and annotated to show that a narrow exists. This markup is used in the language templates in the collapse operations (`collapse.emf`) to show that a narrow exists. The narrow region is applied using the **narrow-buffer** command as follows:

```
0x34 narrow-buffer .scheme.fold-close collapse-markup
```

This defines a narrow region with a color scheme mark-up. The mark-up line is annotated by the macro `collapse-markup` once the narrow is created. The macro is defined as follows:

```
0 define-macro collapse-markup
    !if &les @# 0
        ; no markup line modificaton
        !abort
    !endif
    set-variable $result &cat &trr &lef @wl 40 " ..."
!emacro
```

The `collapse-markup` macro process the numeric argument @#(4) to determine the mark-up type to be applied to the buffer. When the numeric argument is negative then **!abort**(4) is called to remove the mark-up. When the numeric argument is positive then a mark-up line annotation is created by taking the first 40 characters of the line and appending "`...`" to indicate that this is a narrow. The original **narrow-buffer** call included a **hilight**(2) line scheme so the resultant mark-up line will be hilighted and show the first 40 characters of the defined followed by an ellipsis.

### NOTES

Alpha mark set by **set-alpha-mark**(2) in text which is subsequently narrowed out will automatically remove the narrow if the user returns to the mark using **goto-alpha-mark**(2).

**get-next-line**(2) operates on the unnarrowed buffer and will remove any narrows hiding the 'next' line.

When copying a region containing a narrowed region(s) with **copy-region**(2) then the content of the narrowed region(s) is also copied. A numeric argument to **copy-region**(2) may be used to change the behavior to copy the visual representation only.

## FURTHER EXAMPLES

**c-hash-eval**(3) macro defined in *cmacros.emf* uses **narrow-buffer** to hide regions of source code which have been `#defined` out, improving readability.

**vm**(3) defined in *vm.emf* uses **narrow-buffer** with appropriate arguments to **append-buffer**(2) and **write-buffer**(2) to write out only parts of the current buffer.

## SEE ALSO

**!abort**(4), **$line-flags**(5), **$result**(5), **@#**(4), **@wl**(4), **append-buffer**(2), **c-hash-eval**(3), **collapse-all**(3), **collapse-current**(3), **copy-region**(2), **fold-close**(3), **fold-open**(3), **get-next-line**(2), **goto-alpha-mark**(2), **hilight**(2), **kill-region**(2), **narrow**(2m), **narrow-search-backward**(3), **narrow-search-forward**(3), **save-buffer**(2), **set-alpha-mark**(2), **set-mark**(2), **vm**(3), **write-buffer**(2).

Ref: narrow-buffer(2) File: m2cmd068.2 Date: 2022/06/12

## NAME

newline - Insert a new line

## SYNOPSIS

*n* **newline** (**return**)

## DESCRIPTION

**newline** inserts *n* new lines into the text, move the cursor down to the beginning of the next physical line, carrying any text that was after it with it. The next line may automatically be indented depending on the current buffer mode, see **indent**(2m), and **wrap**(2m).

## SEE ALSO

**indent**(2m), **wrap**(2m), **buffer-mode**(2).

Ref: newline(2) File: m2cmd090.2 Date: 2022/06/12

## NAME

next-buffer - Switch to the next buffer
find-buffer - Switch to a named buffer

## SYNOPSIS

*n* **next-buffer** (**C-x x**)
*n* **find-buffer** "*buffer-name*" (**C-x b**)

## DESCRIPTION

**next-buffer** switches to the *n*th next buffer in the buffer list in the current window, the default *n* is `1`, if *n* is negative then the `0`−*n*th previous buffer is selected. If `0` or a number greater than the number of buffers is specified then the command fails.

**find-buffer** switches to buffer "*buffer-name*" in the current window. If the buffer does not exist and a zero argument *n* is supplied then the command fails. If the buffer does not exist but no argument or a positive argument *n* is specified then a new buffer is created, at which point the file-hook is evaluated.

If a negative argument *n* is given to **find-buffer** then the buffer will be hidden. Any window displaying "*buffer-name*" will find another buffer to display. This functionality is often used with the **hide**(2m) buffer mode. If a value of -1 is given then the buffer will not be hidden in a window whose **$window-flags**(5) are set to lock the buffer to the window. If a value of less than -1 is given then the buffer is hidden from all windows.

The numeric argument of **find-buffer** is treated as a bit mask as follows:

`0x01` Bit set then the buffer is created if it does not exist. Bit clear then the command fails if the buffer does not exist.

`0x02` Locate the buffer whose buffer file name matches the argument *buffer-name*.

`0x04` Bit set then an inactive buffer (i.e. a buffer whose contents have not been loaded) shall remain inactive and the buffer contents shall not be restored from file. This bit setting is used to allow the buffer variables to be interrogated without causing the file to load. Setting of this bit must be used with care.

`0x08` Bit set causes the buffer to be hidden.

`0x10` Forces the buffer to be hidden from all windows.

`0x20` Stop the history of the outgoing buffer from being incremented.

**0x40** If this bit is set and the buffer is created then the buffer's major-mode evaluation is not performed (i.e. the **$buffer-major-mode**(5) is not set and the file hook is not executed).

If the current buffer has an *$buffer-ehook* command set then this command is executed before the new buffer is switched in. If the new buffer has a $*buffer-bhook* command set then this command is automatically executed after the new buffer is switched in but before control returns to the user.

## SEE ALSO

**next-window-find-buffer**(2), **hide**(2m), **$buffer-major-mode**(5), **major-mode**(2).

Ref: next-buffer(2) File: m2cmd091.2 Date: 2026/06/19

## NAME

next-frame - Change the focus to the next frame

## SYNOPSIS

*n* **next-frame** (**C-x 5 o**)

## DESCRIPTION

**next-frame** changes the focus to the next frame. The numerical argument *n* can be used to select the type of frame to change to, it is a bit based flag defined as follows:

**0x01** Allow the selection of an internal frame.

**0x02** Allow the selection of an external frame.

The default operation when *n* is omitted is to allow the selection of either type of frame, but an external frame is selected in preference. This is not the same as giving the command an argument of `3` where all frames are circled in creation order (i.e. no priority given to an internal or external frame).

## SEE ALSO

**create-frame**(2), **delete-frame**(2).

Ref: next-frame(2) File: m2cmd163.2 Date: 2022/06/12

## NAME

next-window - Move the cursor to the next window
previous-window - Move the cursor to the previous window

## SYNOPSIS

*n* **next-window** (**C-x o**)
*n* **previous-window** (**C-x p**)

## DESCRIPTION

**next-window** makes the next window down the current window, if the current window is the last one in the frame the first one is selected. The numeric argument *n* can be used to modify this default behaviour, it is a bitwise flag where the bits are defined as follows:

**0x01** If there is no 'next' window because this is the last then if this bit is set the search for the next window is allow to continue with the first window of the frame. As the default argument *n* is `1` this is the default behaviour.

**0x02** When this bit is set windows whose **$window-flags**(5) are set to be ignored by this command are not skipped. The setting of bit `0x010` of a windows **$window-flags** will make the default action of this command skip it which means the the command may not simply select the next window but the next window without this flag set. Setting this bit of the numeric argument will force the command to always select the next window.

**0x04** When set the search for the next window starts at the first window instead of the current window, this can be used to find the first window in the current frame.

**previous-window** makes the next window up the current window. The numeric argument *n* has the same effect on this command as for **next-window** except bit **0x04** starts the search at the last window of the frame.

## EXAMPLE

The following example visits every window in the current frame printing the buffer it displays with a second pause between each one:

```
; go to the first window
!force 6 next-window
!while $status
    1000 ml-write $buffer-bname
    ; go to the next window - fail if this is the last
    !force 2 next-window
!done
```

## NOTES

Both commands fail if a suitable window cannot be for, see the example on how this can be used.

## SEE ALSO

**next-window-find-buffer**(2), **next-window-find-file**(2), **set-position**(2), **goto-position**(2), **$window-flags**(5).

Ref: next-window(2) File: m2cmd095.2 Date: 2022/06/12

## NAME

next-window-find-buffer - Split the current window and show new buffer

## SYNOPSIS

**next-window-find-buffer** "*buffer*" (**C-x 3**)

## DESCRIPTION

**next-window-find-buffer** splits the current window into two near equal windows, and swaps the current windows buffer to the given *buffer*. It is effectively a **split-window-vertically**(2) command followed by a **find-buffer**(2). When there is insufficient space in the current window to perform the split, then the current window is replaced. The requested *buffer* is always displayed, if the buffer does not already exist it is created.

## SEE ALSO

**find-buffer**(2), **split-window-vertically**(2), **next-window-find-file**(2).

Ref: next-window-find-buffer(2) File: m2cmd126.2 Date: 2022/06/12

## NAME

next-window-find-file - Split the current window and find file

## SYNOPSIS

**next-window-find-file** "*file*" (**C-x 4**)

## DESCRIPTION

**next-window-find-file** splits the current window into two near equal windows, and loads the given *file* into the current window. It is effectively a **split-window-vertically**(2) command followed by a **find-file**(2).

When there is insufficient space in the current window to perform the split, then the current window is replaced. The requested *file* is always displayed, if the file does not already exist it is effectively created within MicroEmacs (although it will not exist on the disk until a save operation is performed).

The numeric argument *n* can be used to modify the default behaviour of the command, where the bits are defined as follows:

**0x01** If the file does not exist and this bit is not set the command fails at this point. If the file does not exist and this bit is set (or no argument is specified as the default argument is `1`) then a new empty buffer is created with the given file name, saving the buffer subsequently creates a new file.

**0x02** If this bit is set the file will be loaded with **binary**(2m) mode enabled. See help on **binary** mode for more information on editing binary data files.

**0x04** If this bit is set the file will be loaded with **crypt**(2m) mode enabled. See help on **crypt** mode for more information on editing encrypted files.

**0x08** If this bit is set the file will be loaded with **rbin**(2m) mode enabled. See help on **rbin** mode for more information on efficient editing of binary data files.

## SEE ALSO

**find-file**(2), **next-window-find-buffer**(2), **split-window-vertically**(2), **binary**(2m), **crypt**(2m), **rbin**(2m).

Ref: next-window-find-file(2) File: m2cmd127.2 Date: 2022/06/12

## NAME

osd - Manage the On-Screen Display

## SYNOPSIS

*-4* **osd**
*-3* **osd**
*-2* **osd**
*-1* **osd** *n*
**osd -1** *flag*
**osd** *n* **0** *flags* ["*scheme*"] ["*x-pos*" "*y-pos*"] ["*min-width*" "*min-depth*" "*max-wid*" "*max-dep*"] ["*default-button*"] ["*initial-item*"] [["*title-bar-scheme*"] ["*Text*"]] ["*resize-command*"] ["*control-command*"] ["*init-command*"]
**osd** *n i flags* ["*item-scheme*"] ["*width*" "*depth*"] ["*text*"] ["*argument*" "*command*"]
*n* **osd**

## DESCRIPTION

The **osd** command manages the On-Screen Display, menu and dialogs. For information about the main menu see Top Main Menu.

The command takes various forms as defined by the arguments. Each of the argument configurations is defined as follows:

### Main Menu-Bar Status

**osd -1** *flag*

This invocation determines the state of the top main menu bar. The state is set by the argument *flag* defined as:

`1` - enable.
`0` - disable.
`-1` - disable and destroy.

### Dialog Creation and Redefinition

**osd** *n* **0** *flags* ["*scheme*"] ["*x-pos*" "*y-pos*"] ["*min-width*" "*min-depth*" "*max-wid*" "*max-dep*"] ["*default-button*"] ["*initial-item*"] [["*title-bar-scheme*"] ["*Text*"]] ["*resize-command*"] ["*control-command*"] ["*init-command*"]

This invocation creates or resets the base properties of dialog *n*. The *flags* argument determines the arguments and are defined as follows:

**A** Defines dialog as an alpha type dialog, items are added according to their string text value. Alpha dialogs may not have separator or child items.

**i** Used with the **A** flag, sets the alpha ordering to be case insensitive.

**G** Create a Grid dialog. Every item in the dialog is given a single character boarder around it. If one of the dialogs items is also given a '**G**' flag, the boarder is drawn as a box around it, otherwise spaces are used.

**N** Create a Note-Book (or tabs) dialog. The dialog can only contain one dialog inclusion item ('**I**') and Note-Book pages ('**P**'). Pages added before the Inclusion item (page item number is lass than the inclusion page item number)

will be drawn at the top of the note-book, those added after will be drawn at the bottom.

**b** Draw boarder, draws a boarder around the outside of the dialog. See also *flag* **t** (title) as flag effects the boarder.

**a** Defines the absolute start-up position of the dialog in the arguments *x-pos* and *y-pos*, which are the column and row positions respectively of the dialog from the top left-hand corner of the display. The arguments must be specified. e.g. the main menu is defined with an absolute position of (0,0). If the dialog can not be fully drawn on the screen at the given position it will be moved to a position which shows the most.

**o** Specifies an offset to the dialog position calculated by MicroEmacs in the arguments *x-pos* and *y-pos*, which are the column and row offsets. This flag is ignored when flag **a** is also specified. If the dialog can not be fully drawn on the screen at the new position it will be moved to a position which shows the most.

**s** Sets the size of the dialog. **osd** automatically resizes a dialog to fit the contents, this flag should be considered as a size hint for **osd**, and is not guaranteed to be honored. If the dialog has a boarder (flag **b**) the size given should include the boarder size.

The arguments, *min-width*, *min-depth*, *max-width* and *max-depth* must be specified, as

**+ve** The actual size of the dialog, minimum and maximum sizes.

**0** *min* value should be specified as desired window size, *max* may be 0 which specifies the screen size.

**-ve** *min* defines the maximum size. *max* is unlimited.

The following table shows possible combination of the sing parameters and their effect:

*min*=0, *max*=0
: Default setting, makes dialog as small as possible, with a maximum size of the screen.

*min*=0, *max*=50
: Make dialog as small as possible with a max of 50 characters.

*min*=50, *max*=0
: Make dialog as small as possible, but make it at least 50 characters big and no larger than the screen.

*min*=30, *max*=-1
: Make dialog at least 30 characters big with no upper limit, very useful for dialogs being used as scrolled children.

*min*=-1, *max*=50
: Make dialog 50 characters big.

*min*=-1, *max*=0
: Make dialog the same size as the screen.

*min*=-1, *max*=-1
: Make dialog as big as possible (do not do this unless you have a large amount of memory to throw away).

**S** Sets the main dialog scheme, The default scheme when not specified is **$osd-scheme**(5) See macro file `fileopen.emf` for an example.

**D** Sets "*default-button*" to be the button executed when the `return` key is pressed and no other button has the focus. The current default button is typically denoted by "`> Button <`", other buttons are denoted by "`[ Button ]`" or have no denoting characters, this is configured via the **$window-chars**(5) variable.

**I** Set "*initial-item*" to be selected when the dialog first opens, if this item is an automatically opened sub-menu then the child menu will also be opened.

**t** Title bar is present - draws the title bar. The *text* argument is optional Also see flags **H**, **c** and **r**.

**H** Defines the title bar color scheme if flag **t** is specified. If *t* is absent the option is ignored.

**c** Centers the title bar text if specified. Option **t** must be specified, otherwise the option is ignored.

**r** Right justifies the title bar text if specified. Option **t** must be specified, otherwise the option is ignored.

**R** Defines the dialog as re-sizable. The *resize-command* argument must be specified and the command should resize the dialog to the sizes given in **$result**(5) in the format "`wwwwdddd`", where `w` is width and `d` the depth. If the *resize-command* is aborted then that resize operation is abandoned.

**M** Identifies the dialog as the main menu dialog.

**C** Binds a command to the dialog, which is automatically executed when the dialog is opened. When the dialog with a **C** attribute is opened, it is rendered on the screen and then a command, defined by *control-command* is invoked, when the command completes the dialog is closed.

The command dialog is typically used to create status messages. e.g. a "`Busy - Please Wait`" dialog box, such a dialog may be implemented when saving the current buffer then create the simple dialog and sent the *control-command* to **save-buffer**(2). The dialog would be defined as:

```
osd 200 0 "btcHC" %osd-title-scheme "Saving Buffer" save-buffer
osd 200 1 ""
osd 200 2 "" "Busy - Please Wait"
osd 200 3 ""
200 osd
```

If the dialog has buttons which need to become active then control can be returned to **osd** by calling **osd** with no arguments, e.g. in the above example the dialog can be made to stay on the screen until the user selects `okay` by:

```
define-macro test-osd
    save-buffer
    osd 200 2 "" "Save Complete"
    osd 200 4 "BcfH" %osd-ebtt-scheme "  \HOkay  " f void
    -4 osd
!emacro

osd 200 0 "btcHC" %osd-title-scheme "Saving Buffer" test-osd
osd 200 1 ""
osd 200 2 "" "Busy - Please Wait"
osd 200 3 ""
osd 200 4 "BcfHS" %osd-dbtt-scheme "  Okay  "

200 osd
```

The above mechanism is how **spell-buffer**(3) operates.

**B** Makes the mouse right Button have the same behaviour as the left, by default the right mouse button simply closes the dialog. This is useful for some dialogs which are opened using the right mouse button.

**f** Automatically uses the first letter of an item's text as the hot key. Unlike the normal hot keys, the letter is not hi-lighted and when typed by the user the item is only selected, not executed. This flag also disables the normal hot-keys for the dialog, so all text strings are copied literally.

**n** Disables '\n' characters in text fields leading to multi lines. By default a text item of "Hello\nWorld" will create an item 5 by 2 characters big.

If "*init-command*" is given then this function is always called just prior to the dialog being displayed so it can be used to configure the dialog.

### Dialog Destruction

**-1 osd** *n*

This invocation destructs a dialog *n*. The dialog is only destroyed if it is not currently being displayed.

### Dialog Item Creation and Redefinition

**osd** *n i flags* ["*item-scheme*"] ["*width*" "*depth*"] ["*text*"] ["*argument*" "*command*"]

This invocation type adds a new item *i* to a dialog *n*, the operation of the invocation is controlled by the *flags* as follows:

**D** Disable item *i*, the item is ignored and is not rendered in the dialog.

**I** Include dialog "*argument*" into this dialog. If "*command*" is specified then it is called prior to the child being constructed and can be used to define the child. This is similar to the **M**s command. See also flag **b**.

**P** Item is a Note-Book page, the item must have text and have an argument which is the osd dialog to be show when the page is activated.

**M** Item is a sub-menu, The argument "*argument*" specifies the sub-menus osd dialog number. A "*command*" may also be specified which is executed first, this can actually re-define the item and set the dialog number, e.g.

```
; To start with the dialog number is unknown
osd 1 1 "M" f submenu-setup

define-macro submenu-setup
    osd 200 0 ....
    ....
    ; Now the sub-menu number is known redefine parent item,
    ; note the setup  command  is not given as we have now set
    ; it up!
    osd 1 1 "M" 200
!emacro
```

See also options **m**, **s** and **d**.

**m** Sub-menu must be manually opened, using hot-key, the return key or the left mouse button.

**s** Specifies the sub-menu is to be placed to the south (under) the parent item, The default when omitted is to the right of the parent.

**d** Display sub-menu type, i.e. "**. .**" for auto opening and "  >" for a manual opening sub-menu.

**-** Fill a non-defined chars with '−'s instead of ' 's, used to draw the lines across menus, typically with no text given, e.g.

```
osd 200 5 "-"
```
But could also be specified as:
```
osd 200 5 "-c" "Lined"
```

**C** Item is a check-box. The setting of the check-box is evaluated when the dialog is first drawn, re-draw and whenever any item is executed. A "*command*" must be specified which must both return the current setting when the given argument (of 1) is given (!abort if false, !return if true) and change the value if the argument value is negated. The text string must also be specified, the first 6 characters are used in the drawing of the check box. The format can be shown as follows:

```
String\State        Off          On
"123456"             "12356"      "12456"
" (-+)^"             " (-)"       " (+)"
"^[ *] "             "[ ] "       "[*] "
"^^NY^^"             "N"          "Y"
"^^^^^^"             ""           ""
```

Note that no character is rendered when a '^' character is used. See also **p** for prepending the check-box.

**p** Prepend the check-box box. By default a check box is drawn as:

```
"Check box12?56"
```
This option changes it to:
```
"12?56Check box"
```

**x** When the item is executed do not exit the dialog. Often used with Check-boxes.

**i** The command given is a command line string which is executed in a similar fashion to **execute-line**(2). Note that if an argument is required it is usually specified in the string, i.e.

```
osd "i" "text" 5 "1000 ml-write @#"
```
writes the argument (i.e. 5) for 1 second.
```
osd "i" "text" 5 "my-command"
```
in this case *my-command* will not be given an argument,
```
osd "i" "text" 5 "10 my-command"
```
in this case *my-command* will be given an argument of 10,
```
osd "i" "text" 5 "@# my-command"
```
in this case *my-command* will be given an argument of 5.

**h** Horizontally add the next item, e.g.

```
osd "h" "1st on line "
osd ""  "2nd on line"
```

Will produce "`1st on line 2nd on line`". If there is not enough room on a single dialog line to display all the horizontally added items then the line is split and as many lines as needed are used.

**c** Center the text for the item in the middle of the dialog.

**r** Right hand justify the text for the item.

**t** Set the item to be in the tab order of the dialog. The order of tab items in the dialog is auto-generated, based on item *i* numbers.

**b** Child inclusion is a scroll box type. By default a child inclusion simply draws the whole child dialog at the position. If this flag is specified then arguments "*width*" and "*depth*" must also be supplied and a window displaying "*width*" by "*depth*" of the child is created. The size of this item will be "*width*"+1+*ss* by "*depth*"+1+*ss* where *ss* is the scroll bar size which is 1 or 2 depending on the setting of **$scroll-bar**(5). It is up to the user to ensure that the child dialog being displayed is at least "*width*" by "*depth*" characters in size, if this is not true then the effect is undefined, (a crash dump is not out of the question).

**f** Fix the item size to the given "*size*", by default an item is expanded to the width of the dialog.

**E** Item is an entry box type. Use the '**z**' flog or set the text to a string of `#`'s to set the size of the entry text box. Similar to Check-boxes, the command given must both return and set the value depending on value of the argument given. The value must be returned in **$result**(5) if the given argument (or 1 for '`f`') is given, and the value must be set (usually using **@ml**(4) or **@mc**(4)) if the argument is negated. The absolute value of the argument is maintained.

```
set-variable %entry-value "Hello world"

define-macro my-entry-set
    !if &equ @# -1
        set-variable %entry-value @ml "" %entry-value
    !else
        set-variable $result %entry-value
    !endif
!emacro

osd 200 1  "S" " \HEnter text" 2
osd 200 2  "ExHf" %osd-entry-scheme "########" 1 my-entry-set
```

**O** Item is a read-only Combo-box type of entry. As with entry items (type **E**) the command given must return the current value to be displayed in the entry box when it is called with a non-negated argument. However, when the entry box is selected the item behaves like a sub-menu (type `M`), the command is called (with a negated argument) which can be used to create the sub-menu. The type of item is often used with a button item (type **B**) to the right of it which also opens the sub-menu by executing this item.

**B** Item is a Button type. Add the last 2 characters of **$window-chars**(5) to the text string given, one on each side, i.e. if the last two chars are "[]" then:

```
osd "B" " Okay "
```

will be drawn as "`[ Okay ]`". See also flag **T**.

**T** Item is a repeat type, this is typically used with buttons, altering their execution behavior. By default an item is only executed when the left mouse button is released while over the item. However when this flag is specified the item is executed as soon as the left mouse button is pressed and is repeatedly executed until the button is release or the mouse moves off the item. The delay between repeated executions is determined by the variables **$delay-time**(5) and **$repeat-time**(5).

**S** Item is a separator type. This is not often required as any item without anything to execute is automatically set to be a separator. Occasionally a mouse-insensitive item which can be executed is required, typically a text string with a hot key, e.g.

```
osd 200 1  "S" " \HEnter text" 2
osd 200 2  "ExHf" %osd-entry-scheme "########" 1 my-entry-set
```

will be drawn as "`[ Okay ]`". The hot-key is denoted by \H so for osd item 1 then this is the **E** of *Enter*.

Item 1 will have a hot-key which executes item 2 (as no command is given), but it will not hi-light if the mouse is placed over it.

**R** Redraw dialog. Forces a redraw of the dialog when the item is executed. This is not usually required as **osd** generally works out for itself whether a redraw is needed, however, sometimes it does not, most notably when the item sets a variable that is displayed by another item as an entry, e.g.

```
set-variable %entry-value "Hello world"

define-macro my-entry-set
    !if &equ @# -1
        set-variable %entry-value @ml "" %entry-value
    !else
        set-variable $result %entry-value
    !endif
!emacro

osd 200 1  "S" " \HEnter text" 2
osd 200 2  "ExHf" %osd-entry-scheme "########" 1 my-entry-set
osd 200 3  "BxHcfiR" %osd-ebtt-scheme  " \HReset " f "set-variable %entry-value\"\""
```

If item 3 did not have flag **R** set when executed, **osd** would not realize that the change to value `%entry-value` affects the display and the button would not appear to operate.

**H** Sets the item color scheme. Note that for scrolled child items this only sets the scroll-box color scheme, the dialog scheme is used for the rest of the boarder.

**u** Disables the high-lighting of the item when the mouse is over it.

**G** This flag is only applicable in grid dialogs (see flag **G** in dialog creation). The current item will be drawn with a box around it using **$box-chars**(5).

**z** Sets the item size, arguments "*width*" and "*depth*" must be given.

**N** This flag only has an effect on entry item types, it selects 'New-line' style text entry which allows the user to enter multiple line of text using the return key and to end the input using the tab key.

Note that for a non-sub-menu item type, if an argument is given with no command then it is assumed that the number given is the item number to be executed, see flag **S** for an example.

### Dialog Execution

*n* **osd**

This invocation with a single positive numeric argument executes the *n*th dialog.

### Returning Command Control

*-4* **osd**

An invocation of **osd** with an argument of -4 returns control back to the **osd** from a *control-command*. Refer to the **C** flag in the create/reset dialog property for information and an example.

### Current Dialog Redraw

*-2* **osd**

Calling osd with an argument of -2 forces the complete redrawing of current dialog and any sub-dialogs. This is very useful when the execution of one item may effect the appearance of another.

### Redraw All Active Dialogs

*-3* **osd**

Calling osd with an argument of -3 forces the complete redrawing of all currently active osd dialogs. This is better than calling **screen-update**(2) when only the osd dialogs need updating as it suffers less from flickering.

## EXAMPLE

Refer to `osd.emf`, `userstp.emf`, `search.emf`, `spell.emf` and `organize.emf` for examples of the OSD.

## NOTES

In December 2004 the **d** default item flag in dialog creation was replaced with two new flags, **D** and **I**. At the same time the dialog tab order was changed to an automated calculation so a tab index value is no longer required with the item tab **t** flag.

## SEE ALSO

Top Main Menu, **$osd-scheme**(5), **$result**(5), **$scroll-bar**(5), **$window-chars**(5).

Ref: osd(2) File: m2cmd066.2 Date: 2026/06/19

## NAME

osd-bind-key - Create key binding for OSD dialog
osd-unbind-key - Remove key binding from OSD dialog

## SYNOPSIS

**osd-bind-key** *n* "*command*" "*key*"
**osd-unbind-key** *n* "*key*"

## DESCRIPTION

**osd-bind-key** creates a local key binding for a given osd dialog, binding the command *command* to the keyboard input *key*. Only the current root dialog's local bindings are used, local bindings of included dialogs or other root dialogs currently displayed are ignored.

Osd local bindings take priority over default osd bindings, local bindings created using **ml-bind-key**(2) are also used, but any current buffer local bindings created using **buffer-bind-key**(2) are ignored.

## NOTES

The prefix commands cannot be rebound with this command.

Key response time linearly increases with each osd binding added.

As only the root dialog's bindings are used, creating note-book page specific bindings can be awkward. Typically all required keys are bound to the same command which, depending on the page that is currently being displayed, checks if the key pressed is bound on the current page and if so calls the required command. See **organizer**(3), defined in `organize.emf` for an example of this operation.

## SEE ALSO

**osd**(2), **global-bind-key**(2), **ml-bind-key**(2), **buffer-bind-key**(2), **global-unbind-key**(2).

Ref: osd-bind-key(2) File: m2cmd004.2 Date: 2022/06/12

## NAME

pipe-shell-command - Execute a single operating system command
$ME_PIPE_STDERR - Command line diversion to stderr symbol

## SYNOPSIS

*n* **pipe-shell-command** "*command*" [ "*buffer-name*" — "*variable-name*" ] (**esc** @)
*[MS-DOS and Win32s Only]*
**$ME_PIPE_STDERR** "*string*"; Default is undefined.

## DESCRIPTION

**pipe-shell-command** executes one operating system command *command* and pipes the resulting output into a buffer with the name of ***command***. The **$result**(5) variable is set the return or exit-code of the shell command and can be used to test the result.

The argument *n* can be used to change the default behavior of pipe-shell-command described above, *n* is a bit based flag where:

**0x01** Enables the use of the default buffer name ***command*** (default). If this bit is clear the user must supply a buffer name. This enables another command to be started without effecting any other command buffer.

**0x02** Hides the output buffer, default action pops up a window and displays the output buffer in the new window.

**0x04** Disable the use of the command-line processor to launch the program (windows versions only). By default the "**command**" is launched by executing the command:

```
%COMSPEC% /c command
```

Where `%COMSPEC%` is typically command.com. If this bit is set, the "**command**" is launched directly.

**0x08** Detach the launched process from MicroEmacs (windows versions only). By default the command is launched as a child process of MicroEmacs with a new console. With this bit set the process is completely detached from MicroEmacs instead.

**0x10** Disable the command name mangling (windows versions only). By default any '/' characters found in the command name (the first argument only) are converted to '\' characters to make it Windows compliant.

**0x20** Displays the new process window (windows versions only), by default the window is hidden.

**0x40** Raw mode, disable text annotation at the start of the pipe output.

**0x80** Not used by this command.

**0x100** When set the command-line to be run is the first line of buffer *string*, this feature can be used to overcome the default 1024 character command-line limit. When a multiple-line command-line is required, a negative argument can be given to **insert-string**(2).

**0x200** Not used by this command.

**0x400** When this bit is set the command sets the value of the given *variable-name* to the output of the shell command rather than insert it into a buffer. Note that the output length is limited to the maximum variable length, typically 1024 or 2048 bytes, and the output is copied verbatim so a '\0' in the output will terminate the value.

**0x800** When the terminal is operating in UTF-8 mode (i.e. **$system**(5) bit 0x20000000 is set, auto-detected at startup from the locale), pipe output is automatically decoded from UTF-8: multi-byte sequences are converted to the nearest character in the current codepage, box-drawing characters are mapped to the MicroEmacs box-drawing set, and common typographic characters (curly quotes, em-dash, bullet etc.) are approximated with ASCII. Setting bit **0x800** suppresses this automatic decoding so that raw UTF-8 bytes are stored in the buffer instead. Bit **0x40** (raw mode) also suppresses UTF-8 decoding. Note that UTF-8 decoding is not applied when bit **0x400** directs output to a variable.

## NOTES

On MS-DOS and *Win32s* the standard shell **command.com**(1) does not support the piping of *stderr* to a file. Other shells, such as **4Dos.com**(1), do, using the command-line argument ">&". If the environment variable "ME_PIPE_STDERR" is defined (the value is not used) then MicroEmacs assumes that the current shell supports piping of stderr.

## SEE ALSO

**ipipe-shell-command**(2), **shell-command**(2).

Ref: pipe-shell-command(2) File: m2cmd098.2 Date: 2024/07/16

## NAME

popup-window - Pop-up a window on the screen

## SYNOPSIS

*n* **popup-window** "*name*"

## DESCRIPTION

**popup-window** manages the display of a new window on the screen. If only one window exists then it will be split else the current window will changed to one of the other existing visible windows. If the given buffer name "*name*" is not null ("") then the buffer is created, if it does not exist, and swapped in.

The numeric argument *n* alters the behavior of the command as follows:

`0` When *n* is zero is given then the command only succeeds if the given buffer is already being displayed in an existing window, this window is made current.

`1` When *n* is 1 and the given buffer is not visible then a window displaying a system buffer is chosen in preference. A system buffer is one who's name starts with a '`*`' character, e.g. "`*help*`". window used to display

`2` When *n* is 2 the frame containing the buffer is made current, this will interact with the window manager and bring MicroEmacs to the top of the window stack and bring it into focus. The *name* argument is not required.

`3` When *n* is 3 then the frame position on the Desktop Window is allowed to be re-positioned, this is typically called after a font change or screen re-size operation and ensures that the MicroEmacs window is positioned on the screen. The *name* argument is not required.

`4` When *n* is 4 then both the frame position and frame size on the Desktop Window is adjusted, this is the same as `3` except the frame size may also be adjusted to fit on the desktop. The *name* argument is not required.

## SEE ALSO

**find-buffer**(2).

Ref: popup-window(2) File: m2cmd100.2 Date: 2022/06/12

## NAME

prefix - Key prefix command
prefix2 - Control(2) prefix
prefix3 - Control(3) prefix
prefix4 - Control(4) prefix

## SYNOPSIS

*n* **prefix**

Default prefix bindings:

**prefix 1** (**esc**)
**prefix 2** (**C-x**)
**prefix 3** (**C-h**)
**prefix 4** (**C-c**)

## DESCRIPTION

**prefix** sets up to 8 prefix key sequences, allowing two stroke key bindings. The command does not do anything, it is used to create double barrel key bindings such as such as **goto-line**(2) (**esc g**). This binding may be redefined, redefining ALL meta bindings. If the meta bindings are not required the command should first be unbound using the **global-unbind-key**(2).
The prefix key can only be defined using the **global-bind-key**(2), passing the command the prefix number required, for example:

```
1 global-bind-key "prefix" "esc"
2 global-bind-key "prefix" "C-x"
```

Binds the first prefix to the Escape key and the second prefix to Control-x.

The first prefix key (**prefix 1**) differs from the other prefixes since it permits entry of the numeric argument at the message line, e.g. "`esc 1 0 C-f`" will move forward 10 characters.

## NOTES

Invocating this command via **execute-named-command**(2) or by a macro has no effect. It can be bound to only one key sequence which must be a single key stroke such as **C-x** etc. Re-binding the command to another key will not only unbind the new key but also the current **prefix ?** key bindings.

## SEE ALSO

**global-bind-key**(2), **global-unbind-key**(2).

Ref: prefix(2) File: m2cmd028.2 Date: 2026/02/16

## NAME

print-buffer - Print buffer, with formatting
print-region - Print region, with formatting

## SYNOPSIS

*n* **print-buffer**
*n* **print-region**

## DESCRIPTION

**print-buffer** and **print-region** print the current buffer or region, respectively, using highlighting where appropriate. The hilighting assigned to a buffer is defined by the variable **$buffer-hilight**(5) the print scheme is defined with **print-scheme**(2), the **scheme-editor**(3) should be used to create printer schemes.

The printing is typically configured using **print-setup**(3), which can be found in the main menu under **File->Printer Setup**.

The numerical argument *n* is generally used for macro development, it changes the default behaviour of these commands as follows:

**-2** Configures the printer and, on windows platforms, opens a Windows printing dialog box enabling the user to configure the printer, font and page layout. The configuration is stored in the "`/print`" registry.

**-1** Configures the printer, the configuration is stored in the "`/print`" registry.

**0** Configures the printer and, on windows platforms, using the Windows printer, opens a Windows printing dialog box enabling the user to configure the printer, font and page layout. The required printing is then performed.

**1** Configures the printer and performs the required printing.

### Printing Process

When either of these commands are executed the macro file `print.emf` is executed to configure the printer (in a same vain as `me.emf` is executed to configure MicroEmacs for general usage). After the macro file has been executed the "`/print`" registry must contain the information required for printing. Following is a list of registry entries and their use:

**flags** (*integer*)
The setup flags, defined as a bit mask as follows:

`0x0f` - Destination of the printer output.

`0x00` - Buffer only.
`0x01` - Internal queue.
`0x02` - To file only.
`0x03` - To file and command line.

`0x10` - Bit set, header enabled.
`0x20` - Bit set, footer
`0x40` - Bit set, enable line numbers.
`0x80` - Bit set, Enable truncated line character (typically \).

**paper-x** (*integer*)
Paper page width in character cells.

**paper-y** (*integer*)
Paper page depth in character cells.

**page-x** (*integer*)
The logical page width in character cells.

**page-y** (*integer*)
The logical paper depth in character cells.

**specifier-x** (*integer*)
Windows only.

**specifier-y** (*integer*)
Windows only.

**font-face** (*string*)
The name of the font face (Windows only).

**rows** (*integer*)
Number rows per output page.

**cols** (*integer*)
Number of columns per output page.

**mtop** (*integer*)
The size of the top margin in character cells (i.e. where printing may commence).

**mbottom** (*integer*)
The size of the bottom margin in character cells (i.e. where printing stops).

**mleft** (*integer*)
The number of characters of space forming the left magin of the physical page.

**mright** (*integer*)
The number of characters of space forming the right magin of the physical page.

**header** (*string*)
The ASCII text string for the header line.

**footer** (*string*)
The ASCII text string for the footer line.

**port** (*string*)
Printer port identity.

**buffer** (*string*)
The name of the destination buffer.

**file** (*string*)
The name of the destination file.

**strip** (*integer*)
If *integer* value strip spaces from eol.

**device** (*string*)
The ASCII name of the device (i.e. `/dev/lp`).

**eof** (*string*)
The printer codes for the end of the file, may be the empty string if not reqired.

**eol** (*string*)
The printer codes for the end of line character.

**eop** (*string*)
The printer codes for the end of a page.

**sof** (*string*)
The printer codes for the start of a file, may be the empty string if not required.

**sol** (*string*)
The printer codes for the start of a line.

**sop** (*string*)
The printer codes for the start of a page.

**scont** (*string*)
The printer codes for a start of row continuation.

**econt** (*string*)
The printer codes for the end of row continuation.

**hsep** (*string*)
The horizonal logical page separator character.

**vsep** (*string*)
The vertical logical page separator character.

**wsep** (*string*)
The depth in character cells of the vertical logical page separator.

**xsep** (*string*)
The width in character cells of the logical horizontal separator.

**bg-color** (*integer*)
The background colour number.

**command-line** (*string*)
The command line to perform a print operation.

### Printing Under Microsoft Windows Environments

Printing under Microsoft Windows Environments automatically invokes a dialog box to assign and configure the printer page characteristics. The dialog box allows the printer to be selected, enables line numbering, headers and footers.

The dialog allows the user to select the font size, by defining the number of characters that appear on a logical page, and the number of logical pages that appear on a physical page. Selecting the logical and physical page characteristics determine the size of the font. For dense pages with a small typeface then a point size of 6 is appropriate. For clarity, a larger typeface of 10 or 12 points is advised.

## NOTES

The last printer configuration selected by the user is held in the registry file "`print.erf`" which is loaded into the */print-history* registry section. This feature is implemented in the macro file `print.emf`.

## BUGS

Landscape printing under Microsoft Windows environments is temperamental.

Font selection under Microsoft Windows environments does not always determine the most appropriate font size.

The printer interface does not support native postscript generation. (In progress).

## SEE ALSO

**print-setup**(3), **scheme-editor**(3), **print-scheme**(2), **hilight**(2), **printall**(3f), **$buffer-hilight**(5).

Ref: print-buffer(2) File: m2cmd101.2 Date: 2024/07/16

## NAME

print-color - Create a new printer color
print-scheme - Create a new printer color and font scheme

## SYNOPSIS

*n* **print-color** "*col-no*" "*red*" "*green*" "*blue*"
*n* **print-scheme** "*schemeNum*" "*fore*" "*back*" "*font-mask*"

## DESCRIPTION

**print-color** and **print-scheme** are similar to **add-color**(2) and **add-color-scheme**(2) except they configure MicroEmacs's printer scheme.

**print-color** creates a new printer color and inserts it into the printer color table, where *red*, *green* and *blue* are the color components and *col-no* is the printer color index. The printer color table contains 256 entries indexed by *col-no* in the range 0-255. **print-color** may also be used to modify an existing *col-no* index by re-assignment, the existing color definition is over-written with the new color definition.

An argument *n* of 0 to **print-color** resets the printer color table, removing all currently defined colors.

**print-scheme** creates a new printer scheme. A printer scheme maps the **hilight**(2) buffer's text into a print scheme. For example key words could be printed in *bold* or in *blue* etc. **print-scheme** arguments comprise an identifying index number "*schemeNum*", two color values, "*fore*" and "*back*" (defined by **print-color**) and a font setting "*font-mask*". The *font-mask* is a bit mask where each bit is defined as follows:

`0x01` Enable bold font.
`0x02` Enable italic font.
`0x04` Enable light font.
`0x08` Enable reverse font.
`0x10` Enable underlining.

An argument *n* of 0 to **print-scheme** resets the printer scheme table, removing all currently defined printer schemes.

## NOTES

Printer schemes may be created and altered using the **scheme-editor**(3) dialog, the created printer scheme may then be used directly in the **print-setup**(3) dialog. Therefore direct use of these commands is largely redundant.

## SEE ALSO

**scheme-editor**(3), **print-setup**(3), **print-buffer**(2), **hilight**(2), **$buffer-hilight**(5).

Ref: print-color(2) File: m2cmd129.2 Date: 2026/02/17

## NAME

query-replace-string - Search and replace a string - with query

## SYNOPSIS

**query-replace-string** (**esc C-r**)

## DESCRIPTION

**query-replace-string** operates like the **replace-string**(2) command. replacing one string with another. However, it allows you to step through each string and ask you if you wish to make the replacement. The user is prompted for a replacement response as follows:

**?** Help - get a list of options.

**y** or **space**
Make the replacement and continue on to the next string.

**n** Do not make the replacement, and continue.

**a** or **!** Replace the rest of the strings without asking.

**e** Edit the replacement string.

**l** Last replacement, do next and stop.

**u** Undo last replacement.

**^G** Stop the command.

**$result**(5) is set to a list giving the number of substitutions, the final replace mode ('a' for all, 'l' for last or 'q' for query) and the replace with string.

## SEE ALSO

Refer to **search-forward**(2) for a description of the magic mode search characters.

**replace-string**(2), **$result**(5).
Regular Expressions

Ref: query-replace-string(2) File: m2cmd102.2 Date: 2022/06/12

## NAME

quick-exit - Exit the editor writing changes
save-buffers-exit-emacs - Exit the editor prompt user to write changes

## SYNOPSIS

*n* **quick-exit** (**esc z**) [*exitCode*]
**save-buffers-exit-emacs** (**C-x C-c**) [*exitCode*]

## DESCRIPTION

**quick-exit** writes out all changed buffers to the files they were read from, saves all changed dictionaries, killing any running commands and exits the editor.

The numeric argument is defined as follows:

`0` A zero numeric argument then the editor is closed down and any unsaved buffers are lost.

`1` Prompting the user before saving any files, writes out all changed buffers to the files they were read from. Saves all changed dictionaries, killing any running commands and exits the editor.

`2` Discard changed buffers (except the history and registry) and delete any backup files associated with the edited buffer(s). Only valid with **quick-exit**.

**save-buffers-exit-emacs** prompts the user before saving any files, writes out all changed buffers to the files they were read from. Saves all changed dictionaries, killing any running commands and exits the editor.

## MACRO INTERACTION

If the macro **shut-down**(3) is defined then it is invoked by the editor when it is closed. A user defined shutdown action may be associated with the close operation. It is the callers responsibility that the macro terminates correctly otherwise the editor will hang.

For macros development then the **exit-emacs**(2) command provides better control of the exit process than **quick-exit** and **save-buffers-exit-emacs** and should be used instead. **save-buffers-exit-emacs** is actually implemented as **3 exit-emacs**.

## NOTES

All buffers with a name starting with a '*' are assumed to be system buffer (i.e. ***scratch***) and are not saved.

## SEE ALSO

**exit-emacs**(2), **save-buffer**(2), **shut-down**(3).

Ref: quick-exit(2) File: m2cmd103.2 Date: 2022/06/12

## NAME

quote-char - Insert literal character

## SYNOPSIS

*n* **quote-char** "*key*" (**C-q**)

## DESCRIPTION

**quote-char** inserts the next typed character *n* times, default is 1, ignoring the fact that it may be a command character. **quote-char** obeys the current buffer setting of **over**(2m) mode.

## SEE ALSO

**insert-string**(2), **insert-symbol**(3).

Ref: quote-char(2) File: m2cmd104.2 Date: 2022/06/12

## NAME

rcs-file - Handle Revision Control System (RCS) files

## SYNOPSIS

*n* **rcs-file** (**C-x C-q**)

## DESCRIPTION

MicroEmacs RCS support command. The action of this command depends on the current buffer **view**(2m) mode state, the argument *n*, and the existence of an RCS file.

**view-mode ON; RCS file does not exist**
: Removes buffer view mode to enable the user to edit the file.

**view-mode ON; RCS file exists**
: MicroEmacs attempts to check out the file using the command line given by the variable **$rcs-cou-com**(5) (co unlock). The file is then reloaded and the view mode status re-evaluated.

**view-mode OFF; RCS file does not exist**
: MicroEmacs attempts to check-in the file into RCS for the first time using the command-line given by the variable **$rcs-cif-com**(5) (ci first). The file is then reload.

**view-mode OFF; RCS file exists**
: MicroEmacs attempts to check-in the file into RCS using the command-line given by the variable **$rcs-ci-com**(5). The file is then reload.

**negative argument given**
: MicroEmacs attempts to unedit any changes made to the file using the command-line given by the variable **$rcs-ue-com**(5). The file is then reload.

## SEE ALSO

**rcs**(1), **$rcs-file**(5), **buffer-mode**(2), **find-file**(2), **view**(2m).

Ref: rcs-file(2) File: m2cmd105.2 Date: 2026/06/19

## NAME

read-file - Find and load file replacing current buffer

## SYNOPSIS

*n* **read-file** "*file-name*" (**C-x C-r**)

## DESCRIPTION

**read-file** operates like **find-file**(2), this command either finds the file in a buffer, or creates a new buffer and reads the file in. The command destroys the current buffer before the new buffer is created making this command ideal to use when the wrong file was entered on a **find-file**(2). This command is also useful for re-loading files that have changed on disk.

The numeric argument *n* can be used to modify the default behaviour of the command, where the bits are defined as follows:

**0x01** If the file does not exist and this bit is not set the command fails at this point. If the file does not exist and this bit is set (or no argument is specified as the default argument is `1`) then a new empty buffer is created with the given file name, saving the buffer subsequently creates a new file.

**0x02** If this bit is set the file will be loaded with **binary**(2m) mode enabled. See help on **binary** mode for more information on editing binary data files.

**0x04** If this bit is set the file will be loaded with **crypt**(2m) mode enabled. See help on **crypt** mode for more information on editing encrypted files.

**0x08** If this bit is set the file will be loaded with **rbin**(2m) mode enabled. See help on **rbin** mode for more information on efficient editing of binary data files.

**0x10** If this bit is set then the file hook evaluation is not performed when the file is loaded (i.e. the buffer's file hook is not executed).

**0x20** Disables loss of work checks, if this bit is set then the contents of the buffer is destroyed without any user prompts (useful in macros).

## SEE ALSO

**reread-file**(3), **find-file**(2), **view-file**(2), **binary**(2m), **crypt**(2m), **rbin**(2m).

Ref: read-file(2) File: m2cmd106.2 Date: 2022/06/12

## NAME

read-registry - Read in a registry definition file

## SYNOPSIS

**read-registry** "*root*" "*file*" "*mode*"

## DESCRIPTION

**read-registry** loads a registry file **erf**(8) into the internal registry memory, where the information may be queried via the registry macro commands. The arguments are defined as follows:

*root* The root node in the registry to into which the registry contents are attached. The root name is limited to 32 characters in length and is specified without a leading forward slash '/'. The node *root* is created at the root of the registry.

*file* The name of the registry file **erf**(8) to load. This may be an absolute, relative or a **$search-path**(5) specified file; typically it is located on $search-path.

*mode* The *mode* is string specifying the registry node loading and saving modes, each mode is represented by a character. Lower case characters add a mode, upper case characters delete a mode. The modes are defined as follows:

**a** - Autosave
: Automatically saves the registry when it is deleted or unloaded from the registry. The user is not prompted for a save.

**b** - Backup
: Automatically performs a backup of the registry file whenever a save operation is performed.

**c** - Create
: If the registry file cannot be loaded then the *root* node is created and the invocation succeeds. If this mode is omitted then the call fails if the *file* cannot be found.

**d** - Discard
: Marks the registry as discardable. This is typically used for registries that are not saved.

**r** - Reload
: If the registry node already exists then it is deleted and reloaded, see also the merge flag (**m**). By default, when both the **r** and **m** flags are omitted and the registry node already exists the read operation is not performed and the existing node is used.

**m** - Merge
: The registry file is merged with the contents of any existing registry node. (i.e. the existing registry tree nodes are not deleted if they already exist). See also the reload flag (**r**).

**h** - Hidden
: The registry node is created in the *Hidden* state. (i.e. children will not be shown in **list-registry**(2) output). Refer to **mark-registry**(2) for hiding modes.

**y** - Auto-encrypt
: The registry file will be automatically encrypted/decrypted with a key generated from current data, don't change the file name (can be different path) or your **$MENAME**(5). This flag can be used with flag `p`.

**p** - Prompt encrypt
: The registry file will be encrypted/decrypted with a key prompted for, the key is not stored the key must be re-entered when saving any changes. This flag can be used with flag `y`.

**u** - Updated
: Marks the registry as modified. The modified bit is removed when the registry file is saved. If the modified bit is applied to a registry node the user will be prompted to save the registry when it is deleted (or it will be automatically saved when the *Autosave* mode is used).

Multiple modes may be applied.

## EXAMPLE

The following example is a typical call made from a macro using a registry file where the user may edit the registry file. In this case this a reload of the registry is forced to ensure that the most up-to-date contents are retrieved. Note that the name of the registry file is actually retrieved from the *history* registry.

```
set-variable #l1 &reg "/history/address" $user-name
!if &seq &set #l0 &find #l1 ".ab" "ERROR"
    set-variable #l0 &reg "/history" ""
    set-variable #l0 &spr "%s%s.ab" &lef #l0 &rsin "/" #l0 #l1
!endif
read-registry "AddressBook" #l0 "rc"
```

## BUGS

At exit only registry nodes attached to the root are saved.

## SEE ALSO

**copy-registry**(3), **delete-registry**(2), **find-registry**(2), **list-registry**(2), **mark-registry**(2), **save-registry**(2), **set-registry**(2), **&reg**(4), **$search-path**(5), **erf**(8).

## NAME

recenter - Recenter the window (refresh the screen)

## SYNOPSIS

*n* **recenter** (**C-l**)

## DESCRIPTION

**recenter** scrolls the current window so that the cursor position is at the center of the window and redraws the whole screen. If *n* is given then scrolls the window so that the cursor is *n* lines from the top if *n* is positive or from the bottom if negative.

**recenter** is typically used to refresh the screen if it is out of date (i.e. needs to be redrawn).

## SEE ALSO

**screen-update**(2).

Ref: recenter(2) File: m2cmd108.2 Date: 2022/06/12

## NAME

replace-string - Replace string with new string

## SYNOPSIS

*n* **replace-string** (**esc r**)

## DESCRIPTION

**replace-string** replaces all occurrences of one string with another string. The replacement starts at the current location of the cursor and goes to the end of the current buffer.

A numeric argument positive *n* limits the number of strings replaced to *n*. A negative argument *n* limits the number of lines in which the replacement may take place, e.g. a value of -15 restricts the replacement of the string to the next 15 lines from the current cursor position.

**$result**(5) is set to a list giving the number of substitutions, a replace mode of 'a' for all and the replace with string.

## NOTES

When using regular expressions the replacement string can be dynamically generated creating a far more powerful tool for automating complex changes, see section on *Syntax of Replacement Expressions*.

## SEE ALSO

See Operating Modes for a description of the **magic**(2m) and **exact**(2m) modes which change the search space.

**buffer-mode**(2), **query-replace-string**(2), **search-forward**(2), **$result**(5).
Regular Expressions

Ref: replace-string(2) File: m2cmd109.2 Date: 2026/02/04

## NAME

resize-all-windows - Automatically resize the windows

## SYNOPSIS

*n* **resize-all-windows**

## DESCRIPTION

**resize-all-windows** performs an automatic layout of the windows on the screen, reorganizing the windows such that each window has an equal amount of space. The argument *n* determines which axes reorganization is performed in.

- A positive argument reorganizes the windows vertically, leaving the horizontal arrangement as is.
- A negative argument rearranges the windows horizontally, leaving the vertical arrangement as is.
- An argument of zero performs no vertical or horizontal arrangement.
- No argument re-arranges both the vertical and horizontal window layout.

## SEE ALSO

**resize-window-vertically**(2), **resize-window-horizontally**(2), **split-window-vertically**(2).

Ref: resize-all-windows(2) File: m2cmd153.2 Date: 2022/06/12

## NAME

reyank - Restore next yank buffer

## SYNOPSIS

*n* **reyank** (**esc y**)

## DESCRIPTION

Every region killed goes onto a stack, with the most recent at the top. Immediately after yanking text out into the current buffer using **yank**(2), the user may **reyank** which deletes the region just yanked and replaces it with *n* insertions of the next region on the kill stack. Another call to reyank deletes that region and replaces it with the next in the stack etc.

The default behavior of **reyank** is to wrap around the kill chain in a loop (i.e. returns to the start of the list when the end of the kill chain is reached). If a negative argument *n* is specified then **reyank** fails when the end of the kill chain is reached rather than looping back to the start of the list.

The last 15 kills are stored.

## SEE ALSO

**copy-region**(2), **kill-region**(2), **set-mark**(2), **yank**(2).

Ref: reyank(2) File: m2cmd111.2 Date: 2022/06/12

## NAME

save-buffer - Save contents of changed buffer to file

## SYNOPSIS

*n* **save-buffer** (**C-x C-s**)

## DESCRIPTION

**save-buffer** saves the contents of the current buffer if the contents have been changed, writing the buffer back to the file it was read from.

On saving the file, if **time**(2m) mode is enabled then the time stamp string is searched for in the file and modified if located, to reflect the modification date and time.

If **backup**(2m) mode is enabled then a backup copy of the file existing is created and the contents of the buffer are written to the file. Any automatic save copies of the file are deleted.

If the buffer contains a **narrow**(2m) it will automatically be removed before saving so that the whole buffer is saved and restored when saving is complete

If **auto**(2m) mode is enabled the the file is written out in the style indicated by modes **cr**(2m), **lf**(2m) and **ctrlz**(2m). Otherwise the file is written out in the style on the current platform.

The argument *n* can be used to change the default behavior of save-buffer described above, *n* is a bit based flag where:

**0x01** Enables validity checks (default). These include check that the buffer has been modified, if not an error occurs. Also the time stamp of the file to be written is checked, if the file systems file exists and is newer the confirmation of writing is requested from the user. If this flag is not supplied then the buffer is written whenever possible and without any prompts to the user.

**0x02** Disables the expansion of any narrows (see **narrow-buffer**(2)) before saving the buffer. That is, the buffer is saved as it appears on the screen and the content of any narrow regions is discarded.

## NOTES

- **undo**(2) information is discarded when the file is saved.
- Refer to **$auto-time**(5) for a description of the file extensions used by MicroEmacs for backup and temporary files.
- Buffers may also be saved via the **list-buffers**(2) command.

## SEE ALSO

**$auto-time**(5), **$timestamp**(5), **buffer-mode**(2), **find-file**(2), **narrow-buffer**(2), **save-some-buffers**(2), **undo**(2), **backup**(2m), **time**(2m), **undo**(2m), **narrow**(2m), **auto**(2m), **cr**(2m), **ctrlz**(2m), **lf**(2m), **write-buffer**(2), **append-buffer**(2).

Ref: save-buffer(2) File: m2cmd113.2 Date: 2025/08/28

## NAME

save-dictionary - Save changed spelling dictionaries

## SYNOPSIS

*n* **save-dictionary** ["*dictionary*"]

## DESCRIPTION

**save-dictionary** may be used to save one, or all changed, dictionaries back to disk. By default **save-dictionary** prompts for a single dictionary, which is then saved. If the dictionary to be saved has been created within the session (rather than read from disk) the user is always prompted to save and enter a full dictionary file name (pathname) to save to. If the dictionary was not created then the user is only prompted to save if,

- a non-zero argument is supplied
- and the users history registry node "*/history/spell/autosave*" does not exist or its value is zero.

Otherwise the dictionary is automatically saved.

The argument *n* may be used to control the effect of the command, *n* is a bit based flag defined as follows:

**0x01** Enables prompting before saving, only used when saving all dictionaries.

**0x02** Save all changed dictionaries.

## NOTES

This command is called to save all dictionary changes whenever MicroEmacs is exited.

The dictionary auto-save registry value can be changed via the **user-setup**(3) dialog.

## SEE ALSO

**add-dictionary**(2), **delete-dictionary**(2), **spell**(2).

Ref: save-dictionary(2) File: m2cmd112.2 Date: 2022/06/12

## NAME

save-registry - Write a registry definition file

## SYNOPSIS

*n* **save-registry** ["*root*" "*file*"]

## DESCRIPTION

**save-registry** saves a registry tree, defined by *root*, to a registry file *file* in the **erf**(8) format. By default the user is prompted for the registry *root* to save, which must already exist. If the *file* given is the empty string `""`, the registry node *root* must be a root node with an associated file name stored, this file name is used.

The argument *n* may be used to control the effect of the command, *n* is a bit based flag defined as follows:

**0x01** Enables prompting before saving, only used when saving all registries.

**0x02** Save all changed registries except the history node which should be saved using the command **save-session**(3).

## NOTES

This command is called to save all registry changes whenever MicroEmacs is exited.

## SEE ALSO

**copy-registry**(3), **delete-registry**(2), **list-registry**(2), **mark-registry**(2), **read-registry**(2), **save-session**(3), **set-registry**(2), **&reg**(4), **erf**(8).

Ref: save-registry(2) File: m2cmd008.2 Date: 2022/06/12

## NAME

save-some-buffers - Save contents of all changed buffers to file (with query)

## SYNOPSIS

*n* **save-some-buffers**

## DESCRIPTION

**save-some-buffers** cycles through all visible buffers (buffers without mode **hide**(2m) set) and attempts to save all modified ones, writing the contents back to the file from where it was read. For each buffer that has been modified the user is prompted to save the buffer, the following responses can be given:

**y** Saves the buffer and moves to the next modified buffer.

**n** Does not save the buffer and moves to the next modified buffer.

**a** Saves all remaining modified buffers including the current one without further prompting.

**o** Skips all remaining buffers including the current one.

**g** Display the buffer and aborts the command.

The argument *n* can be used to change the default behavior of save-some-buffers described above, *n* is a bit based flag where:

**0x01** Enables the user prompt before the buffer is saved (default). If this flag is not supplied then all modified visible buffers will be written.

**0x02** If set then all modified buffers which are not linked to a file are skipped.

## SEE ALSO

**save-buffer**(2), **save-buffers-exit-emacs**(2), **write-buffer**(2), **hide**(2m).

Ref: save-some-buffers(2) File: m2cmd150.2 Date: 2026/08/20

## NAME

screen-poke - Immediate write string to the screen

## SYNOPSIS

*n* **screen-poke** *row column colorScheme* "*string*"

## DESCRIPTION

**screen-poke** writes a *string* to the screen at position (*row*, *column*) using the given color scheme. The screen coordinates are defined with (0,0) at the top left of the screen.

**screen-poke** by-passes the conventional buffer update and writes directly to the screen buffer. The command has no effect on buffers already showing on the screen and is erased on the next screen update. The *string* is clipped to the screen area hence the caller need not continually check on the size of the client area.

The numeric argument *n* is a bitwise flag which has the following meaning:

**0x01** Don't mark the poke area for update.

**0x02** Don't flush poke to screen.

**0x04** colorScheme is an array of values, one for each letter.

**0xf0** colorScheme pair offset to use.

If the **0x01** flag is absent then the parts of the screen over written by **screen-poke** are marked and refreshed on the next **screen-update** operation, thereby erasing the poked information. If the flag is present the poked information remains on the screen until a forced refresh is performed (i.e. **1 screen-update**) or the window information under the poked screen data is modified.

In macros using many consecutive screen-pokes (e.g. **Patience**(3) to display a pack of cards) most pokes use the 'No flush' flag to improve performance and look on some platforms.

When using the 'No flush' flag for performance, it must not be used on the last poke of the drawing sequence, otherwise nothing that was poked appears at all. It is the flush that puts the accumulated pokes on the screen; **screen-update** does not do this, it is a redraw of the underlying window and will not display poked data. A page is therefore drawn with flags such as **3**, **67** or **99** throughout except for the final poke which would use **1**, **65** or **97**. **Hearts**(3) uses this arrangement, with the status line drawn last as the single flushing poke of each drawing path.

The use of **screen-poke** has largely been reduced to games such as **Metris**(3) since the introduction of **osd**(2) to create dialogs.

## NOTES

Some platforms do not allow all character values to be poked, illegal characters are replaced with a '**.**'.

## SEE ALSO

**osd**(2), **screen-update**(2), **Mahjongg**(3), **Metris**(3).

Ref: screen-poke(2) File: m2cmd099.2 Date: 2026/08/06

## NAME

screen-update - Force screen update

## SYNOPSIS

*n* **screen-update** (**redraw**)

## DESCRIPTION

**screen-update** updates the current screen, usually used in macros. The argument *n* can be used to change the behaviour of this command as follows:

*negative n*
: Disables the next -*n* screen updates, i.e. if *n* is `-1` then the next time the screen needs to be redrawn nothing will happen.

`0`
: Resets the screen update disable count to zero, useful to remember when the the disable feature has been used incorrectly.

`1`
: Full screen update (default), the screen is completely cleared and redrawn (as if garbled).

`2`
: Partial screen update, only the parts of the screen which require updating are redrawn.

`3`
: No screen redraw, only window variables are up-dated. This feature is provided for macros which manipulate the screen view and need to know where the cursor is in the window without redrawing the screen (which may cause unwanted flickering). Note that as the screen is not redrawn not all variables may have the correct value, for example the frame store variable **@fs**(4) could be out of date.

## EXAMPLES

The following macro demonstrates the problems encountered when trying to use screen variables in macros after the current position has changed. The first value printed is the starting cursor Y position and the next value should be one less than the first value due to the call to **backward-line**(2). But it is the same as the first because the screen (and its variables) have not been updated. The subsequent call to screen-update ensures that the third value is the correct one although by giving it an argument of 3 the screen is not visibly updated thus avoiding any annoying screen flicker:

```
define-macro test-screen-update
    set-variable #l0 $cursor-y
    backward-line
    set-variable #l1 $cursor-y
    3 screen-update
    set-variable #l2 $cursor-y
    forward-line
    ml-write &spr "%d %d %d" #l0 #l1 #l2
!emacro
```

## NOTES

Every time the screen requires updating, MicroEmacs executes the *redraw* key, it is similar in mechanism to the user pressing *C-l* to refresh the screen. The user can therefore re-bind the *redraw* key to another command or macro, thereby allowing the user complete control of what is displayed. For example if *redraw* was bound to **void**(2) the screen would not be up-dated (**Note**: this is difficult to get out of and may require MicroEmacs to be killed).

This feature is often exploited by macros which take control of the input and output, such macros include **gdiff**(3), **Metris**(3), and **Mahjongg**(3).

## SEE ALSO

**recenter**(2), **screen-poke**(2).

Ref: screen-update(2) File: m2cmd139.2 Date: 2022/06/12

## NAME

scroll-down - Move the window down (scrolling)
scroll-up - Move the window up (scrolling)

## SYNOPSIS

*n* **scroll-down** (**C-n**)
*n* **scroll-up** (**C-p**)

## DESCRIPTION

**scroll-down** moves the window in the current buffer down by *n* lines, the default when *n* is omitted is 1 windows worth of lines i.e. a next page operation. A negative value of *n* causes the window to move up.

**scroll-up** moves the window in the current buffer up by *n* lines, default when *n* is omitted is 1 windows worth of lines, i.e. a previous page operation. A negative value of *n* causes the window to move down.

## SEE ALSO

**scroll-left**(2), **scroll-right**(2), **$window-y-scroll**(5).

Ref: scroll-down(2) File: m2cmd087.2 Date: 2022/06/12

## NAME

scroll-left - Move the window left (scrolling)
scroll-right - Move the window right (scrolling)

## SYNOPSIS

*n* **scroll-left** (**C-x** <)
*n* **scroll-right** (**C-x** >)

## DESCRIPTION

**scroll-left** moves the window in current buffer left by 1 screen width. If an argument *n* is supplied then the resolution of movement is specified in characters relative to the current displacement. Moving the window in the current buffer left by *n* characters (that is if the current left-hand margin of the screen is column 0, the left hand margin becomes column *n*).

**scroll-right** moves the window in current buffer right by 1 screen width. If an argument *n* is supplied then the resolution of movement is specified in characters relative to the current displacement.

The ends of the lines of a scrolled screen are delimited with a dollar (**$**) character indicating that the text continues. When no scroll is in effect the left hand margin of the screen does not show the **$** symbol. i.e. The line `This text is scrolled on this line` with a current scroll offset of 2 in a 22 column window would appear as follows:

```
         22
|<------------------->|

|$s text is scrolled $|
```

The amount of scroll (*n*) is effectively unlimited, it is possible to scroll all of the text in a buffer out of the window, when only **$**'s appear in the left margin, in the last highlighting color of the line (blank lines always remain blank and are not delimited with a **$**). Text on the current line is handled according to the value of **$scroll**(5) as follows:

**$scroll 0**

The current line ONLY is scrolled (about the current scroll position) to enable the current buffers cursor position to be viewed. To enable the user to determine where the current line is in relation to the scrolled lines then the first character of the current line is interpreted as follows:

**All of user text appears**

```
|$f line of te$|
|At start of l$|
|$f line of te$|
```

Surrounding lines commence with "$" indicates at the start of the line.

**$ in column 0**

```
|$f line of te$|
|$f line of te$|
|$f line of te$|
```

Text column is the same as the surrounding text i.e. the line and window scroll are the same.

### > Left of scroll position

```
|$f line of te$|
|>f line of te$|
|$f line of te$|
```

The current line is to the left of the scrolled position. forward-char (i.e. interpret as −−> indicating the direction of travel) moves the cursor, and therefore the line, towards the natural scroll position (**$** in column).

### < Right of scroll position

```
|$f line of te$|
|<f line of te$|
|$f line of te$|
```

The current line is to the right of the scrolled position. backward-char (i.e. interpret as <−− indicating the direction of travel) moves the cursor, and therefore the line, towards the natural scroll position (**$** in column).

**$scroll 1**

The position of the cursor on the line determines the scrolled position. In this case all lines in the window are scrolled to ensure that the cursor is always visible. This mode is only useful when dealing with large blocks of text whose line lengths do not vary.

## NOTES

The scrolling is an attribute of the WINDOW and not the BUFFER. If the window is closed, or contents swapped to a different buffer then the scroll setting is reset for the next buffer. A return to the previous buffer does not restore the scroll setting. The only case where scrolling is inherited is when a window is split (see **split-window-vertically**(2)).
When binding **scroll-left** to the keyboard then it is important to note that when no argument is specified the resolution is *frame-width*'s. A key binding would operate on character multiples, hence the command should be bound with a numeric argument to perform the perform the keyboard action. e.g.

```
1 global-bind-key scroll-left  "A-left"
1 global-bind-key scroll-right "A-right"
```

To move 5 columns on a key stroke, for an accelerated scroll, then the binding may be re-written as:

```
5 global-bind-key scroll-left  "A-left"
5 global-bind-key scroll-right "A-right"
```

## SEE ALSO

**$scroll**(5), **scroll-up**(2), **scroll-down**(2), **$window-x-scroll**(5).

Ref: scroll-left(2) File: m2cmd149.2 Date: 2022/06/12

## NAME

scroll-next-window-down - Scroll next window down
scroll-next-window-up - Scroll next window up

## SYNOPSIS

*n* **scroll-next-window-down** (**esc C-v**)
*n* **scroll-next-window-up** (**esc C-z**)

## DESCRIPTION

**scroll-next-window-down** scrolls the next window down *n* lines, if *n* is omitted then the next window is scrolled by *window* number of lines (i.e. next screen page).

**scroll-next-window-up** scrolls the next window up *n* lines, as **scroll-next-window-down**.

These commands are useful in macros to control other windows.

## SEE ALSO

**scroll-up**(2), **scroll-down**(2).

Ref: scroll-next-window-down(2) File: m2cmd115.2 Date: 2022/06/12

## NAME

search-buffer - Search the current buffer for a string

## SYNOPSIS

*n* **search-buffer** "*flags*" "*string*"

## DESCRIPTION

**search-buffer** searches for a *string* from the current cursor position, in a forwards or backwards direction, to the end or start of the buffer, respectively. The search direction and mode is specified by the by *flags* argument.

The **search-buffer** command is designed to be used from the macro language rather than the command line as the search may be defined independently of the buffer setting of **magic**(2m) mode, which allows regular expression pattern matching, or **exact**(2m) mode, which makes the search case sensitive. The *flags* argument is used to explicitly set the **magic**(2m) and **exact**(2m) search modes.

The *flags* argument is defined as a character string which defines the search behavior:

`b` - backward search
: Specifies the direction of the search as backwards, when omitted then a forward search is performed.

`e` - enable exact search
: A **exact**(2m) mode search is performed, the characters are matched case sensitively.

`E` - disable exact search
: A **exact**(2m) mode search is not performed, the characters are matched case insensitively.

`m` - enable magic
: A **magic**(2m) mode search is performed, the search *string* is interpreted as a regular expression.

`M` - disable magic
: A **magic**(2m) mode search is not performed, the search string is searched for literally.

When the *flags* are not specified then the default behavior is to search in a forwards direction inheriting the **exact**(2m) and **magic**(2m) modes of the buffer. Within macros it is suggested that the search modes are explicitly specified unless the buffer modes are to be inherited.

The numeric argument *n* is interpreted in the same way as a **search-forward**(2) command, except when 0 is given:

**n > 0** The *n*th occurrence of the *string* is located.

**n = 0** All occurrences of the *string*, from the current location, are counted and returned in **$result**(5). The cursor is left at the location of the last occurrence.

**n < 0** The first occurrence of the *string* is located in the next *n* lines.

## DIAGNOSTICS

The command returns a status of FALSE if the *string* could not be located (or *n*th *string* where *n* occurrences are requested). If the *string* is found within the given search criteria the return status is TRUE.

## EXAMPLE

The following examples show the use of the **search-buffer** command. The **search-buffer** is usually used with the **!force**(4) directive which traps any FALSE return status and may be subsequently tested using the **!if**(4) directive on the status **$status**(5) variable. If the **!force** directive is omitted then the macro will abort with a failure status if the search fails (sometimes this behavior may be required).
Case sensitive search forward for the word "foo", this is not a magic regular expression.

```
!force search-buffer "eM" "foo"
!if $status
    ; found string
!else
    ; not found
!endif
```

Case sensitive search bacward for the word "foo", this is not a magic regular expression.

```
search-buffer "beM" "foo"
```

Regular expression search forward for '<' or '>'.

```
search-buffer "m" "[<>]"
```

Regular expression search catching the regular expression. A[a-z]*. The macro variable **@s1** returns the regular expression string that has been found.

```
!force search-buffer "em" "\\(A[a-z]*\\)"
!if $status
    ; found string
    1000 ml-write &spr "Found string '%s'" @s1
!else
    ; not found
!endif
```

## NOTES

**Search-buffer** was added in November 2004 to simplify macro writing and to speed up macro execution. It replaces the macro commands **regex-forward**(3) and **regex-backward**(3). The command simplifies macro development by allowing the **exact**(2m) and **magic**(2m) modes to be specified on the command line, this ensures that the buffer mode does not need to be saved, set and then restored in order to modify the behaviour of a search operation.

## SEE ALSO

**buffer-mode**(2), **exact**(2m), **hunt-backward**(2), **hunt-forward**(2), **isearch-forward**(2), **narrow-search-forward**(3), **magic**(2m), **replace-string**(2), **search-backward**(2), **search-forward**(2), **$result**(5).
Regular Expressions

Ref: search-buffer(2) File: m2cmd019.2 Date: 2026/01/26

## NAME

search-forward - Search for a string in the forward direction
search-backward - Search for a string in the backward direction

## SYNOPSIS

*n* **search-forward** "*string*" (**C-x s**)
*n* **search-backward** "*string*" (**C-x r**)

## DESCRIPTION

**search-forward** searches for a string from the current cursor position to the end of the file. The string is typed on the bottom line of the screen, and terminated with the <ESC> key. Special characters can be typed in by preceding them with a ^Q. A single ^Q indicates a null string. On successive searches, hitting <ESC> alone causes the last search string to be reused.

Searching is affected by **magic**(2m) mode, which allows regular expression pattern matching, and **exact**(2m) mode which makes the search case sensitive.

The numeric argument *n* is interpreted as follows:

**n > 0** The *n*th occurrence of the *string* is located.

**n < 0** The first occurrence of the *string* is located in the next *n* lines.

**search-forward** does not search through text that is hidden with **narrow-buffer**(2). Where a search through hidden regions is required then **narrow-search-forward**(3) should be used instead.

**search-backward** searches backwards in the file. In all other ways it is like **search-forward**.

## DIAGNOSTICS

The command returns a status of FALSE if the *string* could not be located (or *n*th *string* where *n* occurrences are requested). If the *string* is found within the given search criteria the return status is TRUE.

## SEE ALSO

**buffer-mode**(2), **exact**(2m), **hunt-backward**(2), **hunt-forward**(2), **isearch-forward**(2), **narrow-search-forward**(3), **magic**(2m), **replace-string**(2), **search-buffer**(2).
Regular Expressions

Ref: search-forward(2) File: m2cmd116.2 Date: 2022/06/12

## NAME

set-alpha-mark - Place an alphabetic marker in the buffer

## SYNOPSIS

**set-alpha-mark** "*?*" (**C-x C-a**)

## DESCRIPTION

**set-alpha-mark** places an alpha mark at the current location in the buffer which can be returned to from anywhere in the buffer using the command **goto-alpha-mark**(2). The user is prompted for a mark name which can be any alphabetic character. the mark is destroyed if the line is deleted.

## SEE ALSO

**goto-alpha-mark**(2).

Ref: set-alpha-mark(2) File: m2cmd118.2 Date: 2022/06/12

## NAME

set-char-mask - Set character word mask

## SYNOPSIS

*n* **set-char-mask** "*flags*" ["*value*"]

## DESCRIPTION

**set-char-mask** returns or modifies the setting of MicroEmacs internal character tables. The argument *n* defines the action to be taken, as follows:

**-1** Removes characters from the given set.

**0** Returns characters in the given set in **$result**(5).

**1** Adds characters to the given set.

The first argument "*flags*" determines the required character set as follows:

**M** Character set Map. Internally MicroEmacs uses a standard character set, or code page, for each language so that only one set of spelling rules and dictionaries are required, however the current display character set may be different so a mapping is required to enable MicroEmacs to convert between the two. By convention MicroEmacs uses the most appropriate Windows code page for its internal character set, e.g. Windows CP-1252 is the internal character set used for American, British, French, German, Spanish, Portuguese and Italian while CP-1253 is used for Greek. Windows code pages are used in preference to ISO-8859 character sets because they tend to have more characters available so have slightly better language coverage.

The "*value*" for the **M** flag must be a string containing pairs of characters, an internal character set character followed by its display character set equivalent. For example, if the current internal character set is CP-1252 and the display character set is CP-437 (a common DOS code page), then 'e' acute must be mapped from 0xe9 in CP-1252 to 0x82 in CP-437 so "*value*" should contain "`\xe9\x82`" as well as many other mappings. All characters in the display character set should be mapped if possible rather than just the letters most commonly used by the current language as this creates the best support for any text entered.

Some display character sets may not have all the characters available in the internal character set, for instance DOS code page CP-437 does not have an upper-case 'E' grave. In this case an ordinary 'E' should be used as a sensible replacement, i.e. "'EE" (where 'E is an 'E' grave) as this is the best that can be done given the limitations of the current display character set.

This flag cannot be incrementally altered, any calls to alter this set leads to the resetting of all the character tables so the character mapping must be performed first and in a single call. No other set may be altered in the same call.

**c** Unicode character mapping. This flag is used to define a mapping between the upper half of the current display character set (the lower half is assumed to be ASCII) and the Unicode character set. This flag cannot be incrementally altered, any call to set

this mapping first resets the mapping table to 0's so the mapping must be performed in a single call. No other set may be altered in the same call. The **value** must be a continuous string of up to 128 4-digit hexadecimal numbers, each value being either 0 for no mapping or the U+#### Unicode character value. The values must be given in ascending order with the first being for character 0x80 etc.

The mapping is used by MicroEmacs on Windows and UNIX XTerm systems to better map the system clipboard text to the current display character set. For example XTerm typically uses an A-F code page which does not support a euro currency symbol, however CP-1252 based fonts can be installed and used correctly by MicroEmacs allowing support for the euro and many other characters. This mapping then allows MicroEmacs to correctly handle these characters when copied between different applications. The mapping table is also used by the **&uni**(4) functions, **expand-iso-accents**(3) and **change-buffer-charset**(3) commands.

**d** Is Displayable. Characters in this set can be directly displayed to the screen (as a single character) when occurring in a buffer. When a character not in this set is to be displayed it is performed using more than one character. Characters in the range 1-31 are displayed as "^?" where ? is the ASCII character plus 64, (e.g. 0x01 -> 65, i.e. "^A") otherwise the character is displayed in the form "\xhh" where hh is the hex form of the ASCII value. One notable exception is the tab character (0x09), by default this character is not displayable, instead it is displayed as a sequence of one or more spaces up to the next tab stop.

**p** Is Pokable. Similar to **d**, characters in this set can be poked to the screen when using **screen-poke**(2). When found in a binary file the character is displayed in the right hand column. Unlike **d**, any character outside this set will be displayed as a single period '.', indicating that it cannot be displayed.

**P** Is Printable. Similar to **d**, characters in this set may be printed as a single character when using **print-buffer**(2) or **print-region**(2). Any character not in this set is printed in a similar fashion to **d**.

**I** Internal character set map list. This set cannot be altered using this flag, character mappings must be set up using flag **M**. The order of the characters in the returned **$result** string is the same as the order for flag **D**.

**D** Display character set map list. This set cannot be altered using this flag, character mappings must be set up using flag **M**. The order of the characters in **$result** when returned is the same as the order for flag **I**.

**a** Is Alphabetic letter. Characters in this set are alphabetical characters, used by many MicroEmacs commands such as **forward-word**(2). When setting, the "*value*" must specify pairs of internal character set chars, an upper-case character followed by its lower-case equivalent. This enables commands such as **lower-case-word**(2) to operate correctly regardless of the font and language being used. Some character sets may not have all the characters available for rendering, for instance PC DOS Code page 437 does not have an upper-case 'e' grave. In this case an ordinary 'E' should be used as a sensible replacement, i.e. "E‘e" (where ‘e is an 'e' grave). However, this will lead to all upper-case 'E's to map to a lower-case 'e' grave in a case changing operation, this may be corrected by adding a further mapping of 'E' to 'e' to over-ride the 'e' grave mapping, i.e. "E‘eEe". This technique does fail when changing the case more than once, when all lower case 'e' graves will be lost.

Note that the returned character list will pair all lower-case characters with their upper-case equivalent letters first.

**l** Is Lower case letter. This set cannot be altered using this flag, alterations to the alphabetic set must be performed using flag **a**. Characters in this set are all the lower-case letters, typically the characters 'a' to 'z'. The order may not be the same as returned by flag **u**.

**u** Is Upper case letter. This set cannot be altered using this flag, alterations to the alphabetic set must be performed using flag **a**. Characters in this set are all the upper-case letters, typically the characters 'A' to 'Z'. The order may not be the same as returned by

**h** Is Hex-decimal Digit. The set is rarely used as it is invariably the digits '0' to '9' and the letters 'a' to 'f' in upper and lower case. It is often used in the setting of **$buffer-mask**(5).

**A** Is Alpha-numeric. This set cannot be altered using this flag, alterations to the alphabetic set must be performed using flag **a**. Characters in this set are either alphabetic characters or the digits 0-9.

**s** Is Spell extended word character. The characters in this set are recognized by the spell checker as characters which may be considered part of a word, for example the period '.'s in e.g. or the hyphen '-' in hyphenated-words. Typically this set contains the characters ''', '-' and '.'.

**1**, **2**, **3** & **4**
Is in Word. These user definable sets are used to add characters to a buffer's word character set, affecting the operation of commands like **forward-word**(2). Many different file types operate better with a different word character set, e.g. it is preferable to include the '_' character when editing C files. See variable **$buffer-mask**(5).

**k** Keyboard character map. This flag is used to configure a mapping from a foreign non-latin keyboard (such as Russian) to Latin characters, setting the mapping correctly can greatly improve usability in MicroEmacs as changing keyboards is required less frequently. The mapping is used by character input routines which first check the main key typed (e.g. the Cyrillic character) and if not bound or invalid MicroEmacs uses this keyboard mapping to translate the character into a Latin character. This allows the user to execute binding such as "C-x k" or responding to "[y/n]" prompts without having to change keyboards.

As with flag **M**, this cannot be incrementally altered, any call to set this mapping first resets the mapping table so the mapping must be performed in a single call. No other set may be altered in the same call. When setting, the "*value*" must supply pairs of characters, the keyboard character followed by the character to map it to, typically an ASCII character.

Unless stated otherwise, multiple flags may be specified at the same time returning a combined character set or setting multiple properties for the given "*value*" characters.

## EXAMPLE

For many UNIX XTerm fonts the best characters to use for **$box-chars**(5) (used in drawing **osd**(2) dialogs) lie in the range 0x0B to 0x19. For example the vertical bar is '\x19', the top

left hand corner is '\x0D' etc. These characters are by default set to be not displayable or pokable which renders them useless. They can be made displayable and pokable as follows:

```
set-char-mask "dp" "\x19\x0D\x0C\x0E\x0B\x18\x15\x0F\x16\x17\x12"
```

MicroEmacs variables have either '$', '#', '%', ':' or a '.' character prepended to their name, they may also contain a '-' character in the body of their name. It is preferable for these characters to be part of the variable 'word' so commands like **forward-kill-word**(2) can work correctly. This may be achieved by adding these characters to user set **2** and setting the **buffer-mask** variable to include set **2**, as follows:

```
set-char-mask "2" "$#%:.-"

define-macro fhook-emf
    set-variable $buffer-mask "luh2"
    .
    .
!emacro
```

For the examples below only the following subset of characters will be used:

| Character | Win CP-1252 | Cmd CP-850 | DOS CP-437 |
|---|---|---|---|
| Capital A (A) | A | A | A |
| Capital A grave (`A) | \xC0 | \xB7 | No equivalent |
| Capital A acute ('A) | \xC1 | \x90 | No equivalent |
| Small a (a) | a | a | a |
| Small A grave (`a) | \xE0 | \x85 | \x85 |
| Small A acute ('a) | \xE1 | \xA0 | \xA0 |

As the spell checker for French will operates in Windows CP-1252, the character font mapping (flag **M**) must be correctly setup for spell checking to operate correctly. When CP-1252 is also used as the display character set the mapping is the empty string as the internal and display character set are fully in-sync, but for both Windows Console CP-850 and DOS code page CP-437 the mappings should be set as follows:

```
; CP-850 mapping setup
set-char-mask "M" "\xC0\xB7\xC1\x90\xE0\x85\xE1\xA0"
; CP-437 mapping setup
set-char-mask "M" "\xC0A\xC1AAA\xE0\x85\xE1\xA0"
```

As all the characters in CP-1252 have equivalents in CP-850, the mapping for Windows console is a simple 1-to-1 lossless character list. However the missing capital **A**'s in CP-437 causes problems, for the command **change-buffer-charset**(3) it is preferable for a mapping of **`A** to be given, otherwise the document being converted may become corrupted and unreadable. Therefore a mapping of **`A** to **A** is given to alleviate this problem, similarly **'A** is also mapped to **A** leading to loss of information.

This leads to a further problem with the conversion of CP-437 back to CP-1252, if the mapping the 'A's was left as just "\xC0A\xC1A" the last mapping (**'A** to **A**) would also be the back conversion for **A**, i.e. ALL **A**'s would be converted back to **'A**'s. To solve this problem, a further seemingly pointless mapping of **A** to **A** is given to correct the back conversion.

While ISO-8859-1 (Latin 1) supports a very similar set of characters to CP-1252, it lacks some accented 'S', 'Y' & 'Z' characters must be mapped to their plain letter equivalents.

For languages which use accented characters, the alphabetic character set must be extended to include these characters for letter based commands like **forward-word**(2) and **upper-case-word**(2) to operate correctly. However, the letter set should be fully extended for each code page regardless of the language being used as an 'a' acute should always be considered a letter even though it is unlikely to occur. The addition of extra letters must achieve two goals, firstly to define whether a character is a letter, enabling commands like **forward-word** to work correctly. The second is to provide an upper case to lower case character mapping, enabling commands like **upper-case-word** to work correctly. This is achieved with a single call to **set-char-mask** using the **a** flag as follows:

```
set-char-mask "a" "\xC0\xE0\xC1\xE1"
```

Note that this flag always expects an internal character set based string, this allows the same map character list to be used regardless of the display character set being used, i.e. the above line can be used for CP-1252, CP-850, CP-437 & ISO-8859-1 code pages. But it does mean that the internal to display character set mapping (flag **M**) must already have been provided.

Similar mapping problems are encountered with the **a** flag as with flag **M** above. The problem is not immediately obvious because the mapping is always given in internal character set which will support the widest set of characters, but when CP-437 is used the mapping string of "`A\x85A\xA0`" must be used. As can be seen, **A** is mapped last to **'a** so an upper to lower character operation will convert a **A** to **'a**. A similar solution is used, a further mapping of **A** to **a** is given to correct the default case mapping for both **A** and **a**, i.e. the following line should always be used instead:

```
set-char-mask "a" "\xC0\xE0\xC1\xE1Aa"
```

## SEE ALSO

**forward-word**(2), **upper-case-word**(2), **change-buffer-charset**(3), **&uni**(4), **$buffer-mask**(5), **screen-poke**(2), **spell**(2), **$buffer-tab-width**(5).

## NAME

set-cursor-to-mouse - Move the cursor to the current mouse position

## SYNOPSIS

*n* **set-cursor-to-mouse**

## DESCRIPTION

**set-cursor-to-mouse** sets the current window and cursor position to the location of the mouse on it's last event (button press or release). This command may change the current window. If the line on which the mouse was located was the message line then the no action is taken, if the line was a window mode line the that window is made the current window but the cursor location within the window remains the same. This is usually used in user defined macros that control the functionality of the mouse.

Invocation of this command sets the variable **$mouse-pos**(5) which determines where the mouse is within the window. Interrogation of the variable following the command may be used to determine if the mouse is located on one of the more specialized window or screen regions.

An argument *n* determines if the command is permitted to change windows and cursor position, where the bits are defined as follows:

**0x01** Permit changing of the current window, if not permitted and the mouse is not in the current window **set-cursor-to-mouse** returns an error condition in **$mouse-pos**(5).

**0x02** Permit changing of the cursor position.

The default argument when omitted is *3*, allowing the changing of both.

When writing macros to cut and paste using the mouse, care should be taken to ensure that the window at the button release is the same is at the button press. If this is not undertaken, undesired effects could result. The use of **set-position**(2) and **goto-position**(2) are most usefully used with this command to restore existing window context.

## SEE ALSO

**$mouse-pos**(5), **$mouse-x**(5), **$mouse-y**(5), **$window-mode-line**(5), **$window-scroll-bar**(5), **set-scroll-with-mouse**(2), **set-position**(2), **goto-position**(2).

Ref: set-cursor-to-mouse(2) File: m2cmd119.2 Date: 2022/06/12

## NAME

set-encryption-key - Define the encryption key

## SYNOPSIS

**set-encryption-key** (**esc e**)

## DESCRIPTION

**set-encryption-key** sets the encryption key for files loaded or saved with **crypt**(2m) mode enabled. This must be performed for each file, key is not entered into the history. The key can be set for each file on the command line using the **-k** flag. When saving a buffer in encryption mode the key will be prompted for if not already set.

## SEE ALSO

**buffer-mode**(2), **crypt**(2m), **find-file**(2), **find-cfile**(3).

Ref: set-encryption-key(2) File: m2cmd120.2 Date: 2023/02/21

## NAME

set-mark - Set starting point of region

## SYNOPSIS

**set-mark** (**esc space**)

## DESCRIPTION

**set-mark** is used to delimit the beginning of a marked region. Many commands are effective for a region of text. A region is defined as the text between the mark and the current cursor position. To delete a section of text, for example, one moves the cursor to the beginning of the text to be deleted, issues the **set-mark** command by typing **esc space**, moves the cursor to the end of the text to be deleted, and then deletes it by using the **kill-region**(2) (**C-w**) command. Only one mark can be set in one window or one buffer at a time, and MicroEmacs will try to remember a mark set in an off screen buffer when it is called back on screen.

A region is a block of text to be acted upon by some MicroEmacs commands. It is demarcated by the **POINT** on one end and the **MARK** at the other. The point is the primary location identifier where most of the action takes place and is always between two characters. The point is indicated by the cursor position in that it is just left of the cursor. The point is also significant in that it defines one end of the region. The mark, on the other hand, is invisible, and is used to demarcate the other end of the region and is set through **set-mark**.

The current buffer mark line and column position may be obtained from within a macro using the variables **$window-mark-line**(5) and **$window-mark-col**(5).

## SEE ALSO

**copy-region**(2), **exchange-point-and-mark**(2), **kill-region**(2), **reyank**(2), **yank**(2), **$window-mark-col**(5), **$window-mark-line**(5).

Ref: set-mark(2) File: m2cmd121.2 Date: 2026/06/19

## NAME

set-position - Store the current position
goto-position - Restore a stored position

## SYNOPSIS

*n* **set-position** "*label*"
*n* **goto-position** "*label*"

## DESCRIPTION

**set-position** stores current window, buffer, cursor and mark position information against the given '`label`' (a single alpha-numeric character). **goto-position** takes the positional information stored against the given '`label`' and restores the window, buffer and cursor positions from those previously **set**.

A call to **set-position** with the same label over-writes the previous stored information, a call to **goto-position** does not alter the information and may be restored multiple times.

The numerical argument to **set-position** is used to define the information that is stored in the position item. The argument is intrepreted as a bitmask, flagging what information is to be stored. The bit mask is defined as follows:

`0x001` Store the current window.

`0x002` Store the current window's horizonal scroll (value of **$window-x-scroll**(5)).

`0x004` Store the current window's current line horizonal scroll (value of **$window-xcl-scroll**(5)).

`0x008` Store the current window's vertical scroll (value of **$window-y-scroll**(5)).

`0x010` Store the current buffer.

`0x020` Store the current window's current line using an alpha mark.

`0x040` Store the current window's current line number (value of **$window-line**(5)).

`0x080` Store the current window's current column offset (value of **$window-col**(5)).

`0x100` Store the current window's mark line using an alpha mark.

`0x200` Store the current window's mark line number (value of **$window-line**(5) when mark was set).

`0x400` Store the current window's mark column offset (value of **$window-col**(5) when mark was set).

When *n* is not specified, the default value is `0x03f`, i.e. store all information required to return to the window, buffer and cursor position.

The argument supplied to **goto-position** similarly interpreted as a bitmask, restoring the positional information. When the numerical argument *n* is omitted the same default is used when omitted on the store. On restoring a position, information stored during the call to **set-position** which is not requested in corresponding **goto** is ignored, similarly information requested in a **goto** which was not stored in the **set** is also ignored.

## EXAMPLE

The following example shows the typical use of these commands:

```
set-position "a"
    .
    .
goto-position "a"
```

The following example stores the current position at the start of a macro sequence, if `my-command` is not successful (**$status** equals 0) the original position is restored:

```
set-position "\x80"
!force my-command
!if &equ $status 0
    ; command failed, return to the original position
    goto-position "\x80"
!endif
```

Note '`\x80`' is interpreted as the character with the ASCII value of 0x80 which is a non-alphanumeric character, this is permitted in macros to avoid using alphanumerics.
The following example shows how the current position can be restored after re-reading a file:

```
0xce set-position
read-file $buffer-fname @mna
; a numeric argument of 0xce is not
; required as this is the default
goto-position
```

## NOTES

The position item may store and restore the current line by using an alpha mark or the line number. The restrore strategy will determine what is required, as follows:

The main benefit from using an alpha mark is that the position is maintained even when the buffer is edited, for example if the position is stored at line 10 and a line is subsequently inserted at the top of the buffer, if the line number was used then it would return back to the 10th line which is the old 9th line whereas if an alpha mark were used it would correctly return to the 11th line, as expected.

The disadvantage of using an alpha mark is that it is only associated with that buffer. In some cases a position may need to be restored in another buffer (e.g. when re-reading a buffer the original buffer may be deleted first), in this situation the buffer line number must be used.

Commands **set-window** and **goto-window**, which simple stored and returned to the current window, were replaced by set-position and goto-position in August 2000. The following macro implementations can be used as a replacement:

```
define-macro set-window
    1 set-position "\x80"
!emacro

define-macro goto-window
    goto-position "\x80"
!emacro
```

## SEE ALSO

**set-alpha-mark**(2), **find-buffer**(2), **$window-x-scroll**(5), **$window-xcl-scroll**(5), **$window-y-scroll**(5), **$window-line**(5), **$window-col**(5).

Ref: set-position(2) File: m2cmd110.2 Date: 2024/10/06

## NAME

set-registry - Modify a node value in the registry.

## SYNOPSIS

**set-registry** "*root*" "*value*"
*2* **set-registry** "*root*" "*new-name*"

## INTRODUCTION

The *registry* is a hierarchical tree of node names optionally with associated node values that is used for storing and retrieving values. The registry mechanism is used for saving and retrieving persistent user information between sessions.

## DESCRIPTION

**set-registry** adds a new value or modifies an existing value in the registry. *root* is an absolute path node name to the new entry. The first path component MUST exist or the call fails, subsequent path components (a '/' separated path) are created if they do not already exist. *value* is the value to assign to the node.

If a numeric argument of 2 is specified the *root* is renamed to *new-name*, the *root* must exist or the call fails. If the *new-name* contains a path (i.e. "`new/path/name`") the registry node is moved to the new path which must be an existing absolute path, i.e. not relative to its current location.

The value assigned to a node with **set-registry** may be retrieved using **&reg**(4).

## EXAMPLE

The following call (where the registry node `/history` already exists):

```
set-registry "/history/foo/windows/printer" "foo-bar"
```

constructs a registry hierarchy of the form:

```
"history" {
  "foo" {
    "windows" {
      "printer"="foo-bar";
    }
  }
}
```

The value of the registry node may be retrieved using:

```
&reg "/history/foo/windows/printer"
```

which would return the value "`foo-bar`".

## HISTORY

The **get-registry**(2) command was removed in January 2005 in preference to the **&reg**(4) function, the existing get command was never used.

## SEE ALSO

**copy-registry**(3), **delete-registry**(2), **find-registry**(2), **list-registry**(2), **mark-registry**(2), **read-registry**(2), **&reg**(4), **erf**(8).

Ref: set-registry(2) File: m2cmd042.2 Date: 2024/11/11

## NAME

set-scroll-with-mouse - Scroll the window with the mouse

## SYNOPSIS

*n* **set-scroll-with-mouse**

## DESCRIPTION

The **set-scroll-with-mouse** command controls the scrolling of a window by the mouse. This is a two stage process, the first stage locks the cursor to the mouse, the second stage scrolls the screen.

The first stage (locking) is performed when the mouse is located on the scroll-box (typically when the left button is depressed i.e. **pick-mouse-1**). **set-scroll-with-mouse** is invoked with an argument *n*, this causes the mouse position to be recorded ready for a scroll. Depending on the scroll method, the blank lines present at the end of the buffer are scrolled off the screen.

Subsequent calls to the **set-scroll-with-mouse** are made with no argument, the window is scrolled by the relative displacement of the mouse from it's locked position, motion is limited at the end of the scrolling region. Scrolling is proportional to the buffer length. The command is typically bound to **move-mouse-1** which results in an update whenever the mouse is moved by the user.

When the button is released **drop-mouse-1** then the scrolling is stopped by unbinding **move-mouse-1**, thereby breaking the binding between the mouse moving and the scroll command.

The scrolling utilizes fractional mouse positional information (i.e. units smaller than a character cell), if available, resulting in a smoother scrolling motion.

## EXAMPLE

The following example shows how the command is used.

```
0 define-macro mouse-scroll-pick
    1 set-scroll-with-mouse         ; Lock mouse position to scroller
    global-bind-key set-scroll-with-mouse "mouse-move-1"
!emacro

0 define-macro mouse-scroll-drop
    global-unbind-key "mouse-move-1"
!emacro

global-bind-key mouse-scroll-pick "mouse-pick-1"
global-bind-key mouse-scroll-drop "mouse-drop-1"
```

When the left button is 'picked', **mouse-scroll-pick** lock the cursor to the mouse and binds mouse movement to **set-scroll-with-mouse** so that whenever the mouse is moved the cursor will be repositioned appropriately. When the button is 'dropped', the mouse movement is unbound so that the cursor will no longer be locked to the mouse.

## SEE ALSO

**$mouse-pos**(5), **$scroll-bar**(5), **set-cursor-to-mouse**(2).

## NAME

set-variable - Assign a new value to a variable
unset-variable - Delete a variable

## SYNOPSIS

**set-variable** "*variable*" "*value*" (**C-x v**)
**unset-variable** "*variable*"

## DESCRIPTION

**set-variable** sets the given register (**#** name), system (**$** name), global (**%** name), buffer (**:** name) or command (**.** name) variable to the given value, erasing its current value. The returned value of an undefined variable is the string "ERROR", this maybe used to determine whether a variable has been set.

**unset-variable** unsets the given variable so that it no longer exists. The variable must be a global (**%**), buffer (**:**) or command (**.**) variable, system (**$**) variables cannot be unset.

The *value* may be quoted or unquoted, if there are any white space characters, or characters open to other interpretation (e.g. **@wc**) in *value* then quotes should be used.

*value* may contain control characters which are delimited by a back slash (\) which include:

\n newline
\t tab
\\ backslash

Confusion sometimes arises in macros with the back slash, as the back slashes are dereferenced when set. Commands such as **replace-string**(2) where the command itself utilizes back slashes. In this case the number of back slashes should be doubled as the variable contents under go two stages of dereferencing.

## SEE ALSO

**describe-variable**(2), **list-variables**(2), **&set**(4).

Variables
Introduction to Variable Functions
Register Variables

Ref: set-variable(2) File: m2cmd122.2 Date: 2022/06/12

## NAME

shell - Create a new command processor or shell

## SYNOPSIS

**shell** (**C-x c**)

## DESCRIPTION

**shell** creates a new command processor or shell. Upon exiting the shell, MicroEmacs redraws its screen and continues editing. The exceptions to this are as follows:

**X-Windows**
: A new **xterm** is spawned off and editing control is returned to MicroEmacs once the **xterm** has initialized.

**Microsoft Windows**
: A new MS-DOS shell is created and control is returned to MicroEmacs once the DOS console window has initialized. The shell created is determined by the MS-DOS environment variable `COMSPEC`, this may be a replacement shell e.g. `4DOS`.

## SEE ALSO

**ipipe-shell-command**(2), **pipe-shell-command**(2), **shell-command**(2), **suspend-emacs**(2).

Ref: shell(2) File: m2cmd071.2 Date: 2022/06/12

## NAME

shell-command - Perform an operating system command

## SYNOPSIS

*n* **shell-command** "*string*"

## DESCRIPTION

**shell-command** performs an operating system call with the given *string* as its argument. The command only fails if the shell-command call returns -1. The **$result**(5) variable is set the return or exit-code of the shell command and can be used to test the result.

The argument *n* can be used to change the default behavior of the shell-command, *n* is a bit based flag where:

**0x01** When set shell-command waits for the command to complete, this is the default behavior. When not set the command is launched and forgotten about, the value returned in **$result**(5) will not reflect the exit code of the executed process itself.

**0x04** Disable the use of the command-line processor to launch the program (windows versions only). By default the "**command**" is launched by executing the command:

```
%COMSPEC% /c command
```

Where `%COMSPEC%` is typically command.com. If this bit is set, the "**command**" is launched directly.

**0x08** Detach the launched process from MicroEmacs (windows versions only). By default the command is launched as a child process of MicroEmacs with a new console. With this bit set the process is completely detached from MicroEmacs instead.

**0x10** Disable the command name mangling (windows versions only). By default any '/' characters found in the command name (the first argument only) are converted to '\' characters to make it Windows compliant.

**0x20** Displays the new process window (windows versions only), by default the window is hidden.

**0x100** When set the command-line to be run is the first line of buffer *string*, this feature can be used to overcome the default 1024 character command-line limit. When a multiple-line command-line is required, a negative argument can be given to **insert-string**(2).

**0x200** No wait option, when set the the command is run without waiting for any completion status. This allows processes to be run and then forgotten about e.g. a Word processor, Adobe Acrobat reader etc.

## SEE ALSO

**$result**(5), **ipipe-shell-command**(2), **pipe-shell-command**(2), **shell**(2), **suspend-emacs**(2).

Ref: shell-command(2) File: m2cmd131.2 Date: 2024/07/16

## NAME

show-region - Show the current copy region

## SYNOPSIS

*n* **show-region**

## DESCRIPTION

**show-region** manipulates the currently defined region, it can be used to inquire the state of the current region, if any. It can also be used to define a region, enable and disable the region hilighting, as well as move the cursor to the start or end of the region.

Region hilighting occurs between the *mark* (see **set-mark**(2)) and *point* (current cursor) positions within the current buffer. A region is defined when text is copied to the kill buffer, by using any of the kill commands such as **kill-region**(2), or **copy-region**(2). However, the kill region is only visible after a **copy-region**(2) or a **yank**(2) operation. A hilight region is also created on a successful search using commands like **search-forward**(2), the region encloses the search matching string. **Spell**(2) also creates a hilight region around the current spell word. The user can also define their own region using the numeric argument to **show-region**.

The argument *n* supplied to the command indicates the require functionality and can take the following values:

`-3` - Set the start position of the region.
`-2` - Move the cursor the Mark position.
`-1` - Disable the hilighting of the current region.
`0` - Return the current status of the region in **$result**(5).
`1` - Enable the hilighting of the current region.
`2` - Move the cursor the Dot position.
`3` - Set the end position of the region.
`4` - Reactivate the current region.

Where an argument of 0 is used to return the current state the value of $result is a bit based flag where:

**0x01** Indicates a region is currently active (visible).

**0x02** Indicates a region has been fixed (may not visible).

**0x04** Indicates the region is in the current buffer.

**0x08** Indicates the cursor is in the current region.

The color of the selection hilight is defined by **add-color-scheme**(2) and is determined by **$buffer-scheme**(5), **$global-scheme**(5) or **$buffer-hilight**(5).

## DIAGNOSTICS

The following errors can be generated, in each case the command returns a FALSE status:

**[No current region]**
There is no current defined region on which to operate.

**[Current region not in this buffer]**

An argument of 2 or -2 was used and the defined region isn't in the current window so the cursor can not be moved to it.

## NOTES

If no argument is given to the command it hilights the current region, similar to an argument of 1. But the properties of the hilight, namely how long it will be hilighted for, are inherited from the setting of **$show-region**(5), whereas if an argument of 1 is passed in then the hilighting is set to be kept until the region becomes invalid (i.e. as if **$show-region**(5) is set to 3).

## SEE ALSO

**$show-region**(5), **$buffer-hilight**(5), **$buffer-scheme**(5), **$global-scheme**(5), **add-color-scheme**(2), **copy-region**(2), **yank**(2), **search-forward**(2), **spell**(2), **set-mark**(2).

Ref: show-region(2) File: m2cmd132.2 Date: 2026/02/28

## NAME

sort-lines - Alphabetically sort lines

## SYNOPSIS

*n* **sort-lines**

## DESCRIPTION

**sort-lines** alphabetically sorts lines of text in the current buffer from the mark position to the current cursor position. If the buffer mode **exact**(2m) is enabled then the sort is case sensitive, otherwise the sort is case insensitive. By default the text is compared from left to right from column 0 (the left hand edge), if a positive argument *n* is given then the text is compared left to right from the *n*th column, any lines shorter than *n* characters are moved to the top and sorted from column 0.

If a negative argument *n* is given then the text is sorted in reverse order. The comparison starts at column `-1-n`, i.e. an argument of -1 sorts in reverse order from column 0.

## EXAMPLE

The following table gives the results of **sort-lines** for different exact modes and values of *n*.

| Original | | Sorted Lines | | | | |
|---|---|---|---|---|---|---|
| exact | - | n | n | y | y | n | n |
| n | - | - | 1 | - | 1 | -1 | -2 |
| | B | a2 | B | Aa | B | CA | Aa |
| | CA | Aa | c | B | c | c | CA |
| | b1 | B | b1 | CA | b1 | b1 | a2 |
| | Aa | b1 | a2 | a2 | a2 | B | b1 |
| | c | c | CA | b1 | CA | Aa | c |
| | a2 | CA | Aa | c | Aa | a2 | B |

## NOTES

Typically MicroEmacs is executed with **exact**(2m) mode enabled, the macro command **sort-lines-ignore-case** provides a command to sort lines case insensitively while **exact** mode is enabled. The macro is defined as follows:

```
define-macro sort-lines-ignore-case
    set-variable #l0 &bmod exact
    -1 buffer-mode "exact"
    !if @?
        @# sort-lines
    !else
        sort-lines
    !endif
    &cond #l0 1 -1 buffer-mode "exact"
!emacro
```

**sort-lines-ignore-case**(3) is a macro defined in format.emf.

## SEE ALSO

**buffer-mode**(2), **exact**(2m), **sort-lines-ignore-case**(3), **transpose-lines**(2), **uniq**(3).

Ref: sort-lines(2) File: m2cmd124.2 Date: 2022/06/12

## NAME

spell - Spell checker service provider

## SYNOPSIS

*n* **spell** ["*word*"] ["*rules*"] ["*correction*"] ["*rule*"]

## DESCRIPTION

**spell** is a low level command which provides spell checking capabilities for MicroEmacs, it is not designed to be used directly. The action of **spell** depends on the argument given, which is a bitwise flag defined as follows:

**0x001** If set then gets the input word from the user, i.e. "*word*" must be supplied. Otherwise the word input is taken from the current buffer.

**0x002** If set then keeps getting words from the current buffer until either the end of the buffer is reached or an error is found. If the end of the buffer is reached then the command succeeds setting **$result**(5) to the value "*F*". This bit is ignored if bit 0x001 is set. **spell** sets the current show-region to enclose the problematical word and the command **show-region**(2) can be used to move around the word.

**0x004** Adds the given word to a dictionary determined by the state of bit 0x008. If the word is flagged as erroneous (see bit 0x010) then a "*correction*" word must be supplied, otherwise a list of "*rules*" which can be applied to the word must be given, this list can be empty. Note that if the word is not flagged as erroneous and it already exists in the dictionary, the word is not removed, instead a combined rule list is created.

**0x008** When set flags that word additions (bit 0x004) and deletions (bit 0x200) should be made to the ignore dictionary. Otherwise word additions are made the last added dictionary and deletions are made to all main dictionaries.

**0x010** When set flags that the given word is erroneous, used solely by word additions to create auto-corrections.

**0x020** Returns a '—' separated guess list for the given word in **$result**.

**0x040** If bit **0x100** is also set a complete list of valid words derivable from the given word are inserted into the current buffer. Otherwise spell returns **$result**(5) set to the derivative word created when the given "*rule*" is applied to "*word*". The rule applied is the first found of the given rule letter with a matching base ending (see **add-spell-rule**(2)). The word need not exist as not tests for the legality of the resultant word is used, for example in American, executing

```
65 spell "spelling" "V"
```

returns "`spellingive`" in **$result**. Returns the empty string if no rule could be applied.

**0x080** Used with bit 0x002 to enable double word checking.

**0x100** Return information in **$result** about the given word, or the word which is used to derive the given word. The information consists of the spell status, the word as stored in the dictionary, and either the list of valid rules, or the correction word. See also bit **0x040**.

**0x200** Delete the given word from a dictionary determined by bit 0x008

If none of the main functions are used (bits 0x004, 0x020, 0x040 & 0x200) then the status flag is returned in the first column of **$result**. These are defined as follows:

**A** Auto-replace. The word was found and flagged as erroneous. The correction word is given in **$result**, either next to the flag, or if bit 0x100 is set then after the '>' character.

**D** Double word. Indicates that the first problem found is a double occurrence of the same word one after the other.

**E** Erroneous. The word was not found, so is Erroneous

**N** Not a word. The current word found contains no alphabetic characters so is not deemed to be a word, e.g. 3.141593.

**O** Okay. The word was found and is not an erroneous word.

## SEE ALSO

**add-dictionary**(2), **add-spell-rule**(2), **delete-dictionary**(2), **save-dictionary**(2), **show-region**(2), **spell-buffer**(3), **spell-word**(3), Locale Support.

Ref: spell(2) File: m2cmd125.2 Date: 2025/08/28

## NAME

split-window-horizontally - Split current window into two (horizontally)

## SYNOPSIS

*n* **split-window-horizontally** (**C-x 3**)

## DESCRIPTION

**split-window-horizontally** splits the current window horizontally into two near equal windows, each displaying the buffer displayed by the original window.

A numeric argument *n* of 1 forces the left window to be the new current window, and an argument of 2 forces the right window to be the new current window. The default when omitted is the left window.

## SEE ALSO

**$scroll-bar**(5), **$scroll-bar-scheme**(5), **$window-chars**(5), **grow-window-horizontally**(2), **split-window-vertically**(2).

Ref: split-window-horizontally(2) File: m2cmd151.2 Date: 2022/06/12

## NAME

split-window-vertically - Split the current window into two

## SYNOPSIS

*n* **split-window-vertically** (**C-x 2**)

## DESCRIPTION

**split-window-vertically** splits the current window vertically into two near equal windows, each displaying the buffer displayed by the original window. A numeric argument *n* of 1 forces the upper window to be the new current window (default), and an argument of 2 forces the lower window to be the new current window.

## SEE ALSO

**grow-window-vertically**(2), **next-window-find-buffer**(2), **next-window-find-file**(2), **resize-window-vertically**(2), **split-window-horizontally**(2).

Ref: split-window-vertically(2) File: m2cmd128.2 Date: 2022/06/12

## NAME

start-kbd-macro - Start/stop recording keyboard macro end-kbd-macro - Stop recording keyboard macro

## SYNOPSIS

**start-kbd-macro** (**C-x (**)
**end-kbd-macro** (**C-x )**)

## DESCRIPTION

A keyboard macro is a short hand way to repeat a series of characters. In effect, a *recording* is made of the sequence of keys that you hit while defining a keyboard macro. The recording is started with **start-kbd-macro** and ended with **end-kbd-macro**. The recording is then repeated whenever you execute the keyboard macro using **execute-kbd-macro**(2).

Since it is key-strokes that are being saved, you can freely intermix commands and text to be inserted into the buffer.

You can save a keyboard macro for later using the **name-kbd-macro**(2) command, which saves the keyboard macro as a named macro. Otherwise if you start another keyboard macro recording session, the previously defined macro is lost. So make sure that you are done with the current keyboard macro before defining another one. If you have a series of commands that you would like to *record* for later use, **insert-macro**(2) can be used to insert the macro into a text file and can be reloaded using the **execute-file**(2) or **execute-buffer**(2) commands.

Recording commences with **start-kbd-macro** (`C-x (`) and terminates when an **end-kbd-macro** (`C-x )`) is encountered.

## NOTES

Once **start-kbd-macro** has been executed, the mouse is disabled until **end-kbd-macro** is executed. This is because the mouse events cannot be successfully recorded in macros. The main menu can still be used, but only via the keyboard bindings and hot-keys (note that the layout of the menu may change).

## SEE ALSO

**execute-kbd-macro**(2), **insert-macro**(2), **kbd-macro-query**(2), **name-kbd-macro**(2).

Ref: start-kbd-macro(2) File: m2cmd011.2 Date: 2022/06/12

## NAME

suspend-emacs - Suspend editor and place in background

## SYNOPSIS

*n* **suspend-emacs**

## PLATFORM

Supported on UNIX platforms - *irix*, *hpux*, *sunos*, *freebsd* or *linux*.

## DESCRIPTION

**suspend-emacs** suspends the editing processor and puts it into the background. The "*fg*" command restarts MicroEmacs. The prompt to suspend is disabled if a `0` numeric argument *n* is given to the command.

## SEE ALSO

**shell**(2).

Ref: suspend-emacs(2) File: m2cmd130.2 Date: 2022/06/12

## NAME

tab - Handle the tab key

## SYNOPSIS

*n* **tab** (**tab**)

## DESCRIPTION

**tab** manages the `tab` key, typically inserts *n* tabs. The effect of the command is determined by:

**$buffer-indent**
: If **$buffer-indent**(5), is non-zero then the effect of tab is defined by the setting of bit `0x1000` of variable **$system**(5), typically it resets the current line indentation or inserts a tab.

**tab**
: If a tab is to be inserted and this mode is enabled then multiple spaces are used instead of tab characters, see **tab**(2m) mode.

## SEE ALSO

**$buffer-indent**(5), **$buffer-tab-width**(5), **$tab-width**(5), **backward-delete-tab**(2), **indent**(2), **insert-tab**(2), **normal-tab**(3), **tab**(2m).

Ref: tab(2) File: m2cmd064.2 Date: 2022/06/12

## NAME

translate-buffer - Translate buffer from one format to another

## SYNOPSIS

*n* **translate-buffer** [ "*buffer-name*" ]

## DESCRIPTION

**translate-buffer** converts the text in the current buffer to another format, the conversion performed is controlled be the given numeric argument *n* which is bit based where the bits are defined as follows:

**0x003** The lower 2 bits define the base conversion, see below.

**0x004** When this bit is set and converting from UTF-16 or UTF-32 then the buffer is in big-endian byte order, if clear the buffer is in little-endian byte order.

**0x008** When set and converting from a Unicode UTF format the buffer start is checked for a Unicode 'BOM' character. If found it is removed but flagged to be readded on save, see bit **0x08** of variable **$buffer-xlate**(5).

**0x100** When set MicroEmacs attempts to auto-detect the Unicode UTF format of the current buffer, setting bits **0x01** to **0x08** appropriately. MicroEmacs will not attempt to detect when a lossless text format might be best, it always chooses a UTF format. Note that when using this bit, bits **0x01** through **0x08** must be clear.

**0x200** Output the converted text to a new buffer with given **buffer-name** rather than replace the text of the current buffer. With this bit is not set, the current buffer must be clean and unedited.

**0x400** Used with bit **0x200**, when set the new buffer is not automatically displayed.

The base conversion, determined by the first 2 bits, must be one of the following:

**0x00** Convert a Reduced-binary (see **rbin**(2m)) formatted buffer to a lossless text format, in this mode line termination is always a single `\x0A` byte (UNIX style EOL) and a minimum set of bytes (`\x00`, `\x01`, `\x02` and `\x07`) are protected by encoding to the byte sequence "`\x0100#`" (byte `\x01` followed by 2 ASCII '0' (`\x30`) characters followed by # which is the value of the byte being protected, i.e. '0', '1', '2' or '7'). In this format every byte is preserved so it can be treated more like a text file.

**0x01** Convert a Unicode UTF-8 formatted file, loaded in rbin format, to a Unicode preserving text mode. In this mode line termination is auto-detected and handled as normal but the same spacial bytes as lossless text are protected. Then any Unicode character which as a value of `U+0080` or more is then handled in one of 4 ways, if the character is supported by the current MicroEmacs charset it is converted to that single byte value. Otherwise, if the hex value of the character is less than `0x01000` then it is converted to a byte sequence of "`\x01###`" where ### is the hex value of the character in ASCII text, if less than `0x100000` it is converted to byte sequence of "`\x02#####`", if the value is `0x100000` or greater the character is replace with a literal `\x07` byte (MicroEmacs's undefined character) and the value is lost. At the

time of writing there is no use of characters in thiis Unicode plane so this should not be an issue. If the file is invalid, e.g. has invalid character encodings, the command throws an error and the conversion halts.

**0x02** As with conversion **0x01** above except the source must be in UTF-16 format, see bit **0x04** for details on setting the endianness.

**0x03** As with conversion **0x01** above except the source must be in UTF-32 format, see bit **0x04** for details on setting the endianness.

When omitted, the default value on *n* is **0x100** which means the buffer must be a UTF file loaded in **rbin** format, MicroEmacs will then attempt to detect the format being used and convert the current buffer to a lossless text form.

When translate-buffer is used, the resultant buffer will have **xlate**(2m) buffer mode enabled and variable **$buffer-xlate**(5) set so that MicroEmacs can correctly convert the buffer back to its original format when it is written back out.

## EXAMPLE

The following can be used to load a Unicode file of any UTF encoding:

```
25 find-file "unicode_file.txt"
translate-buffer
```

The following can be used to load any file and preserve all bytes, i.e. lossless text mode:

```
25 find-file "any.file"
0 translate-buffer
```

## NOTES

In lossless text mode, long strings without a UNIX new line character will be split at just under the 64KB length, these lines are flagged as not having a line terminator (see bit **0x010** of variable **$line-flags**(5) for more details) so these line breaks do not affect the saved output.

It is important to avoid breaking the encoded byte sequences when editing these 'lossless' buffers, the 4 or 6 byte sequences are simple text bytes and any addition or removal of bytes will alter their value.

## SEE ALSO

**xlate**(2m), **$buffer-xlate**(5), **find-file**(2), **rbin**(2m).

Ref: translate-buffer(2) File: m2cmd054.2 Date: 2025/03/25

## NAME

translate-key - Translate key

## SYNOPSIS

*n* **translate-key** [ "*from*" ["*to*"] ]

## DESCRIPTION

**translate-key** may be used to convert any given input key sequence to another single key. **translate-key** operates at a very low level, before MicroEmacs attempts to evaluate keyboard bindings, so it may be used to solve a variety of keyboard problems such as special language characters and UNIX termcap key sequence bindings (see below).

If a positive numeric argument *n* is given it is used to set the time in milliseconds MicroEmacs waits for another key to be pressed before continuing, the default time use when no argument is supplied is 250ms.

If a numeric argument *n* of -1 is specified then the "*to*" argument is not required and the "*from*" character sequence is removed from the translate key table.

If a numeric argument *n* of -2 is specified then the "*to*" argument is not required; the command returns the key sequence "*from*" is currently mapped to in variable **$result**(5) or returns an error if it is not currently mapped. Note that if the given character sequence is only part of a mapping and is not directly mapped itself (for example, "`C-[ [`" in the listing below) the mapping string returned is an encoded null character ("`\x00`").

If a numeric argument *n* of 0 is specified then no arguments are required; the current translation table is dumped to buffer "**translate-key**". Following is a sample output:

```
"C-h" ........................ "backspace"
"C-[" ........................ "esc"
"C-[ [ 1 ~" .................. "delete"
"C-[ [ 1 1 ~" ................ "f1"
"C-[ [ 1 2 ~" ................ "f2"
"C-[ [ 1 3 ~" ................ "f3"
"C-[ [ 1 4 ~" ................ "f4"
"C-[ [ B" ..................... "down"
"C-[ [ 4 ~" .................. "end"
"C-[ [ 2 ~" .................. "insert"
"C-[ [ 3 ~" .................. "home"
"C-[ [ D" ..................... "left"
"C-[ [ 6 ~" .................. "page-down"
"C-[ [ 5 ~" .................. "page-up"
"C-[ [ C" ..................... "right"
"C-[ [ A" ..................... "up"
"C-[ [ V" ..................... "page-up"
"C-[ [ U" ..................... "page-down"
"C-m" ......................... "return"
"C-i" ......................... "tab"
"\x7F" ....................... "backspace"
```

## FOREIGN KEYBOARDS

Foreign keyboards (non-US/UK) use a variety of key sequences, not recognized by MicroEmacs, to expand the keyboard character range to cope with accented characters. For example, on a German keyboard '`AltGr-m`' (recognized as '`A-C-m`') is used to insert a Greek mu (or micro sign). On a Belgian keyboard '`AltGr-9`' inserts a '{' character.

Many foreign keyboards are already directly supported by MicroEmacs and the keyboard specifics

of a country have been understood and resolved. In these cases the **Keyboard** configuration in **user-setup**(3) may be used for the country location.
If MicroEmacs does not support your keyboard, **translate-key** may be used to fix any key input problems. For the aforementioned examples the following **translate-key** commands would be required:

```
; translate AltGr-m to a Greek mu (char 0xb5)
translate-key "A-C-m" "\xB5"
; translate AltGr-9 to a '{'
translate-key "A-C-9" "{"
```

The problem is complicated further on Microsoft Window's platforms by the simultaneous generation of 2 keys for some Alt-Gr key combinations (this is a side effect of endeavoring to capture all key combinations in this environment). For the Belgian keyboard example, on Win32 platforms an '`AltGr-9`' generates an '`A-C-9`' key first followed immediately by an '`A-C-{`'. As both keys are generated in quick succession this is unexpected and confusing.

When the key is first pressed on a poorly configured system the error "*[Key not bound "A-C-{"]*" is given even when using the command **describe-key**(2) as the key described will be '`A-C-9`' and then the '`A-C-{`' key is generated and interpreted creating the error message.
The variable **$recent-keys**(5) can be used to diagnose this problem and to obtain the 2 keys generated; alternatively use the macro below:

```
define-macro report-2-keys
    ml-write "Press key 1"
    set-variable #l0 @cgk
    ml-write "Press key 2"
    set-variable #l1 @cgk
    ml-write &spr "[The following keys where pressed: \"%s\" \"%s\"]" #l0 #l1
!emacro
```

When executed the user is prompted for the first key; press the required key sequence (in this case '`AltGr-9`'), if you are not prompted for the second key and the result is immediately returned then the key you pressed has generated 2 keys, both of which will be given in the print out, i.e.:

```
"[The following keys where pressed: "A-C-9" "A-C-{"]"
```

The translate-key required to fix this type of problem would be:

```
translate-key "A-C-9 A-C-{" "{"
```

If your keyboard is not directly supported by MicroEmacs, please submit the keyboard name and platform with a working translate-key configuration to JASSPA as a **BUG**.

### UNIX TERMCAP

**translate-key** may also be used to interpret non-standard key sequences for UNIX termcap platforms to standard MicroEmacs keys. Non-standard keys, such as the cursor keys, have system dependent key sequences. The output from these keys usually take the form:

^ [ [X or ^ [ [DX or ^ [ [DDX or ^ [ [DDD

where **^[** is the escape key (27), **D** is a digit and **X** is any character. These keys may be bound to the standard keys, for example the typical output of the cursor keys may be translated as follows:

^ [ [A = **up**, ^ [ [B = **down**, ^ [ [C = **right** and ^ [ [D = **left**

The "*from*" string is specified as this key sequence and the "*to*" string is simply the key it is to be bound to, see **global-bind-key**(2) for a guide to the string format. For the above example the following set of translations are required:

```
translate-key "esc [ A" "up"
translate-key "esc [ B" "down"
translate-key "esc [ C" "right"
translate-key "esc [ D" "left"
```

Note that MicroEmacs interprets `\e` as an escape key. More obscure keys tend to be very platform specific, following are some examples:

```
translate-key "esc [ 2 ~" "insert"
translate-key "esc [ 5 ~" "page-up"
translate-key "esc [ 5 ^" "C-page-up"
```

## EXAMPLE

Using the positive numeric argument it is possible to reduce the delay and there by increase usability is some features. For instance, in the Mouse configuration of **user-setup** there is an option to 'Simulate 3 Buttons' which translates a rapid left and right button press into a middle button press. This is implemented using **translate-key** as follows:

```
10 translate-key "mouse-pick-1 mouse-pick-3" "mouse-pick-2"
10 translate-key "mouse-pick-3 mouse-pick-1" "mouse-pick-2"
10 translate-key "mouse-drop-1 mouse-drop-3" "mouse-drop-2"
10 translate-key "mouse-drop-3 mouse-drop-1" "mouse-drop-2"
```

When a `mouse-pick-1` key is generated MicroEmacs must wait to see if a `mouse-pick-3` key is next and therefore translate both to a single `mouse-pick-2` key. This wait time is usually a quarter of a second but this makes the left button unusable for dragging regions etc as the delay is too long. By giving a argument of 10ms the delay is long enough for a simultaneous left and right button press but short enough for the left button to still be usable on its own.
The positive numeric argument can be very useful for delaying MicroEmacs as well, for example, the character string "' e" can be converted to e-accute using **expand-iso-accents**(3). This could be performed automatically using translate-key as follows:

```
1000 translate-key "' e" "\xE9"
```

The larger 1 second delay give the user enough time to type the 'e' after the "'" character.

### Known Translations

The following are a set of known translations courtesy of **-Th**.

```
; fix Backspace using Putty
-1 translate-key "C-h"   ; remove translation to "backspace"
translate-key "\x7F"  "backspace"  ; Backspace key

; fix del, home keys termcap/Putty/Solaris ME20040403
translate-key "C-[ [ 3 ~" "delete"  ; ^[[3~ ('cat' output)
translate-key "C-[ [ 1 ~" "home"  ; ^[[1~

; add support for termcap/Putty/Solaris
translate-key "C-[ O A" "C-up" ; ctrl up ^[OA
translate-key "C-[ O B" "C-down" ; ctrl down ^[OB
translate-key "C-[ O D" "C-left" ; ctrl left ^[OD
translate-key "C-[ O C" "C-right" ; ctrl right ^[OC

;-----
; SuSE xterm console mode ('me -n')
; 12-Jan-06 SuSE10/kde/gnome
;-----
translate-key "C-[ [ 1 ; 5 A" "C-up"
```

```
translate-key "C-[ [ 1 ; 5 B" "C-down"
translate-key "C-[ [ 1 ; 5 C" "C-right"
translate-key "C-[ [ 1 ; 5 D" "C-left"

translate-key "C-[ [ 1 ; 2 A" "S-up"
translate-key "C-[ [ 1 ; 2 B" "S-down"
translate-key "C-[ [ 1 ; 2 C" "S-right"
translate-key "C-[ [ 1 ; 2 D" "S-left"

translate-key "C-[ [ H" "home"
translate-key "C-[ [ F" "end"
translate-key "C-[ [ 1 ; 5 H" "c-home"
translate-key "C-[ [ 1 ; 5 F" "c-end"

translate-key "C-[ [ 5 ; 5 ~" "C-page-up"
translate-key "C-[ [ 6 ; 5 ~" "C-page-down"
translate-key "C-[ [ 2 9 ~" "menu" ; context menu key
```

## NOTES

The concept of standardized key-bindings is very important for cross platform use and maintenance.

Refer to **global-bind-key**(2) for a list of standard bindings.
One of the easiest ways of obtaining a key sequence is to run **sh**(1) which does not attempt to interpret these keys so when a key is pressed (followed by <RETURN>) the following type of error message is usually generated:

```
sh: ^[[2~:  not found.
```

where `^[[2` is the required key sequence. Another method of obtaining these key sequences is to start MicroEmacs, use **start-kbd-macro**(2) to start a macro definition, press the required keys and then use **end-kbd-macro**(2) followed by **name-kbd-macro**(2) and **insert-macro**(2) to display the keys pressed.

The key sequences generated for these keys are dependent on the machine displaying MicroEmacs as opposed to the machine running it. Often they are the same machine, but when they are not there is no easy method of determining the displaying machine and therefore correctly configuring MicroEmacs.
A better way of obtaining this cross platform consistency is to create an XTerm app-defaults setup file with the correct VT100 key translations, e.g. the setup file could contain the following

```
*vt100.translations: #override \
        Shift<Key>Tab:          string("\033[Z") \n\
        <Key>BackSpace:         string("\177") \n\
        <Key>Delete:            string("\033[1~") \n\
        <Key>Insert:            string("\033[2~") \n\
        <Key>Home:              string("\033[3~") \n\
        <Key>End:               string("\033[4~") \n\
        <Key>Prior:             string("\033[5~") \n\
        <Key>Next:              string("\033[6~") \n\
        Ctrl<Key>Up:            string("\033Oa") \n\
        Ctrl<Key>Down:          string("\033Ob") \n\
        Ctrl<Key>Right:         string("\033Oc") \n\
        Ctrl<Key>Left:          string("\033Od") \n\
        Shift<Key>Up:           string("\033[a") \n\
        Shift<Key>Down:         string("\033[b") \n\
        Shift<Key>Right:        string("\033[c") \n\
        Shift<Key>Left:         string("\033[d") \n
```

By using the environment variable *XUSERFILESEARCHPATH* to ensure that this configuration file is found instead of the system one (found in `/usr/lib/X11/app-defaults`), the key sequences will then be the same across all platforms. See manual page on **xterm**(1) for more information.

## SEE ALSO

**expand-iso-accents**(3), **user-setup**(3), **describe-key**(2), **global-bind-key**(2), **start-kbd-macro**(2), **$result**(5), **xterm**(1), **sh**(1).

Ref: translate-key(2) File: m2cmd133.2 Date: 2026/02/17

## NAME

transpose-chars - Exchange (swap) adjacent characters transpose-lines - Exchange (swap) adjacent lines

## SYNOPSIS

**transpose-chars** (**C-t**)
*n* **transpose-lines** (**C-x C-t**)

## DESCRIPTION

**transpose-chars** exchanges (swaps) the current character under the cursor with the previous character. **transpose-characters** does not operate in column 0 (since there is no previous character). If the cursor is at the end of a line when the command is initiated then the cursor is moved to the previous character and the operation performed from the new position.

**transpose-lines** swaps the next line for the current line and moves to the next line, effectively retaining the same text position. Repeating this *n* times moves the current line *n* lines down.

## EXAMPLE

**transpose-character** performs the following operations (cursor at ^):

```
abcde  => acbde         [Middle of line]
  ^        ^

abcde  => abced         [End of line]
    ^          ^
```

## SEE ALSO

**sort-lines**(2), **transpose-region-line**(3).

Ref: transpose-chars(2) File: m2cmd135.2 Date: 2025/07/23

## NAME

undo - Undo the last edit

## SYNOPSIS

*n* **undo** (**C-x u**)

## DESCRIPTION

**undo** removes the last *n* edits made to the current buffer. The **undo**(2m) buffer mode must be enabled for this command to operate.

When *n* is `-1`, **undo** discards all of the current undo history for the buffer without performing any undo operation. This can be used to free the memory consumed by the undo history after completing significant editing.

When *n* is less than `-1` (e.g. `-2`), **undo** inserts an undo boundary. By default, all edits performed within a single macro execution are treated as one undo group and are removed by a single call to **undo**. Calling `-2 undo` from within a macro creates a break point so that edits made before the call form one undo group and edits made after form a new, separate undo group, each requiring their own call to **undo** to reverse. Multiple boundaries may be inserted by calling `-2 undo` more than once.

The undo information is retained up until the next save operation, at which point the undo information is discarded. When editing large files with gross changes then it is advisable to either disable undo mode, or save frequently to flush the undo buffer, thereby keeping MicroEmacs memory requirements reasonable (most UNIX users have restrictions on the amount of memory that may be consumed by a single process. Windows is restricted by the amount of virtual memory (or swap space)).

## EXAMPLE

The following macro demonstrates the use of the undo boundary. After executing **test-multi-step-undo**, the first call to **undo** removes only `CD`; a second call removes `AB`.

```
define-macro test-multi-step-undo
    insert-string "A"
    insert-string "B"
    -2 undo
    insert-string "C"
    insert-string "D"
!emacro
```

## SEE ALSO

**buffer-mode**(2), **save-buffer**(2), **undo**(2m).

Ref: undo(2) File: m2cmd136.2 Date: 2026/06/19

## NAME

universal-argument - Set the command argument count

## SYNOPSIS

**universal-argument** (**C-u**)

## DESCRIPTION

**universal-argument** sets the argument number passed to a command to 4^*n* (4 to the power of *n*) where *n* is the number of calls to **universal-argument**, e.g. the key sequence "`C-uC-n`" moves down 4 lines, "`C-uC-uC-uC-n`" moves down 4*4*4 = 64 lines.

After invoking the **universal-command** a '-' character can be pressed to negate the argument value, and an alternative numeric argument can be entered using the '*0*' to '*9*' keys.

Invoking this command via **execute-named-command**(2) or by a macro has no effect. The command should be treated as a command key prefix (like **prefix**(2)) in that it may be bound to only one key sequence which must be a single key stroke. Re-binding this command to another key unbinds the new key and also the current **universal-argument** key.

The **prefix 1** key (by default bound to `esc`) may also be used to enter a numeric argument at the message line, e.g. "`esc 1 0 C-f`" will move forward 10 characters.

## SEE ALSO

**prefix**(2).

Ref: universal-argument(2) File: m2cmd137.2 Date: 2022/06/12

## NAME

view-file - Load a file read only

## SYNOPSIS

*n* **view-file** "*file-name*" (**C-x C-v**)

## DESCRIPTION

**view-file** is like **find-file**(2), and either finds the file in a buffer, or creates a new buffer and reads the file in. A new file is left in **view**(2m) mode if the file was found (i.e. cannot be edited).

The numeric argument *n* can be used to modify the default behaviour of the command, where the bits are defined as follows:

**0x01** If the file does not exist and this bit is not set the command fails at this point. If the file does not exist and this bit is set (or no argument is specified as the default argument is `1`) then a new empty buffer is created with the given file name, saving the buffer subsequently creates a new file.

**0x02** If this bit is set the file will be loaded with **binary**(2m) mode enabled. See help on **binary** mode for more information on editing binary data files.

**0x04** If this bit is set the file will be loaded with **crypt**(2m) mode enabled. See help on **crypt** mode for more information on editing encrypted files.

**0x08** If this bit is set the file will be loaded with **rbin**(2m) mode enabled. See help on **rbin** mode for more information on efficient editing of binary data files.

**0x10** If this bit is set then the file hook evaluation is not performed when the file is loaded (i.e. the buffer's file hook is not executed).

## SEE ALSO

**buffer-mode**(2), **find-file**(2), **read-file**(2), **view**(2m), **binary**(2m), **crypt**(2m), **rbin**(2m), **browse**(3).

Ref: view-file(2) File: m2cmd140.2 Date: 2022/06/12

## NAME

void - Null command

## SYNOPSIS

*n* **void**

## DESCRIPTION

**void** does nothing except return `FALSE` if the given argument *n* is zero, `TRUE` otherwise. Used to bind any frequently miss hit keys to something harmless.

## SEE ALSO

**global-bind-key**(2).

Ref: void(2) File: m2cmd141.2 Date: 2022/06/12

## NAME

wrap-word - Wrap word onto next line

## SYNOPSIS

**wrap-word**

## DESCRIPTION

**wrap-word** wraps the current word onto the next line, justifying the current line if the **justify**(2m) mode is enabled. The justification method is defined by **$fill-mode**(5).

## SEE ALSO

**buffer-mode**(2), **fill-paragraph**(2), **$fill-mode**(5), **justify**(2m).

Ref: wrap-word(2) File: m2cmd142.2 Date: 2022/06/12

## NAME

write-buffer - Write contents of buffer to named (new) file

## SYNOPSIS

*n* **write-buffer** "*file-name*" (**C-x C-w**)

## DESCRIPTION

**write-buffer** is used to write the contents of the buffer to a NEW file, use **save-buffer**(2) if the buffer is to be written to the existing file already associated with the buffer.

**write-buffer** writes the contents of the current buffer to the named file *file-name*. The action of the write also changes the file name associated with the current buffer to the new file name.

Unlike **append-buffer**(2), **write-buffer** always replaces an existing file and the new file inherits the buffers file characteristics instead of the old file's.

On writing the file, if **time**(2m) mode is enabled then the time stamp string is searched for in the file and modified if located, to reflect the modification date and time.

If the buffer contains a **narrow**(2m) it will automatically be removed before saving so that the whole buffer is saved and restored when saving is complete

If **backup**(2m) mode is enabled and the buffer is associated with a different file (compared with *file-name*) then any automatic save copies of the file associated with the *buffer* are deleted.

The argument *n* may be used to change the default behavior of write-buffer described above, *n* is a bit based flag where:

**0x01** Enables validity checks (default). These include a check that the proposed file does not already exist, if so confirmation of writing is requested from the user. Also MicroEmacs checks all other current buffers for one with the proposed file name, if found, again confirmation is requested. Without this flag the command will always succeed wherever possible.

**0x02** Disables the expansion of any narrows (see **narrow-buffer**(2)) before saving the buffer.

## NOTES

**undo**(2) information may be discarded when the file is written.

## SEE ALSO

**$auto-time**(5), **append-buffer**(2), **backup**(2m), **buffer-mode**(2), **change-file-name**(2), **file-attrib**(3), **save-buffer**(2), **time**(2m), **write-region**(3).

Ref: write-buffer(2) File: m2cmd143.2 Date: 2025/08/28

## NAME

yank - Paste (copy) kill buffer contents into buffer

## SYNOPSIS

*n* **yank** (**C-y**)

## DESCRIPTION

When a non negative argument is supplied to **yank**, the command copies the contents of the kill buffer *n* times into the current buffer at the current cursor position. This does not clear the kill buffer, and therefore may be used to make multiple copies of a section of text. On windowing systems which support clipboards, such as windows and X-terms, MicroEmacs will also cut to and paste from the global clipboard.

If *yank* is IMMEDIATELY followed by a **reyank**(2) then the *yanked* text is replaced by text of the next entry in the kill ring. (another **reyank** replaces the text with the previous reyank text and so on).

If an negative argument is given, **yank** removes the last 0-*n* items from the kill ring unless *n* is -98 or less.

When *n* is -98 or -99 yank checks for and gets the current global clipboard text regardless of whether MicroEmacs is currently running a macro, this can be useful for macros that need to interact with the system clipboard. If a system clipboard text is available it is inserted into the kill buffer and *not* inserted into the current buffer, if *n* is -99 the text is inserted as a UTF8 encoded string rather than converted to the current MicroEmacs charset.

When *n* is -100 the last search string match is inserted/yanked into the current buffer instead. If *n* is -101 then the matches first regex group is inserted, -102 the 2nd group etc., see help on **@s0**(4) for more information about search groups. This numerical interface can be used to avoid the 2KB variable text limit imposed on all MicroEmacs variables when a macro needs to insert or copy these value. The text of the match or search group is not inserted into the kill buffer.

Text is inserted into the kill buffer by one of the following commands:

> **backward-kill-word**(2), **copy-region**(2), **forward-kill-word**(2), **kill-line**(2), **kill-paragraph**(2), **kill-region**(2), **forward-delete-char**(2), **backward-delete-char**(2).

All the above commands (except **copy-region**) cut text out of the buffer, the last two character delete commands require the presence of a numeric argument to add the deleted text to the kill buffer. If any of these commands are executed immediately after any other (including itself) or the **@cl**(4) variable is set to one of these command, the new kill text is appended to the last kill buffer text.

## NOTES

Windowing systems such as X-Windows and Microsoft Windows utilize a global windowing kill buffer allowing data to be moved between windowing applications (*cut buffer* and *clipboard*, respectively). Within these environments MicroEmacs automatically interacts with the windowing systems kill buffer, the last MicroEmacs kill buffer entry is immediately available for a *paste* operation into another application (regardless of how it was inserted into the kill buffer). Conversely, data placed in the windowing kill buffer is available to MicroEmacs, via

**yank**, until a new item has been inserted into the kill buffer (the data may still be available via **reyank**(2)).

## EXAMPLE

The following example is a basic macro code implementation of the **transpose-lines**(2) command,

```
beginning-of-line
kill-line
forward-line
yank
-1 yank
backward-line
```

Note that similar to **transpose-lines** it does not leave the moved line in the kill buffer, effectively tidying up after itself.

## SEE ALSO

**backward-delete-char**(2), **backward-kill-word**(2), **copy-region**(2), **forward-delete-char**(2), **forward-kill-word**(2), **kill-line**(2), **kill-paragraph**(2), **kill-region**(2), **reyank**(2), **yank-rectangle**(2), @**y**(4), @**cc**(4), @**s0**(4).

Ref: yank(2) File: m2cmd146.2 Date: 2025/06/21

## NAME

auto - Automatic source file line type detection

## SYNOPSIS

### auto Mode

**A** - mode line letter.

## DESCRIPTION

When this mode is enabled the line style of the source file, with respect to `CR/LF/CTRL-Z` characters, are automatically detected and the file (if saved) is written out in the same style as it was read in. This mode is designed to solve the problems of MS-DOS which utilize a '**\r\n**' with every new line and a **ctrl-Z** as a file terminator, conversely UNIX utilizes only '**\n**' as a line terminator.

**auto** mode therefore allows files to be edited across file system types without corrupting the line style of the native platform.

At load time, if **auto** detects CR/LF style line feeds then it enables the buffer modes **cr**(2m) and **lf**(2m); if a CTRL-Z is found at the end of the file then mode **ctrlz**(2m) is enabled. Otherwise these modes are cleared.

At write time, if **auto** mode is enabled then the file is written out is a style determined by modes **cr**, **lf** and **ctrlz**. For example, if **cr** and **lf** were enabled and **ctrlz** disabled then the file would be written out with new lines as '**\r\n**' and with no ending ctrl-z.

If auto is not enabled then the file is written out in the style of the current platform, regardless of the current settings on modes **cr**, **lf** and **ctrlz**.

## SUMMARY

The operation on the modes may be summarized as follows:

### UNIX Systems

- **auto Enabled** UNIX and MS-DOS files may be edited normally, edits are saved in the format read by the system.
- **auto Disabled** UNIX files may be edited normally, files saved as UNIX files. MS-DOS files show a `^M` character at the end of each line (editing is not advised if the purity of the MS-DOS is to be maintained), any edits are written back as displayed on the screen.

### MS-DOS Systems

- **auto Enabled** UNIX and MS-DOS files may be edited normally, edits are saved in the format read by the system.
- **auto Disabled** on reading all files are read and editing may be undertaken normally. On writing, '**\r**'s and a **ctrl-Z** are automatically added. The act of reading a UNIX file and re-writing it translates it to an MS-DOS file.

## NOTES

This mode MUST be enabled globally when the file is loaded for the file style to be correctly detected.

It is **strongly advised** that auto mode is permanently enabled.

Windows systems tend to use a '**\r\n**' style line feed but with no trailing ctrl-z.

## SEE ALSO

Operating Modes, **global-mode**(2), **buffer-mode**(2), **cr**(2m), **lf**(2m), **ctrlz**(2m), **$buffer-fmod**(5).

Ref: auto(2m) File: m2mod000.2 Date: 2022/06/12

## NAME

autosv - Automatic file save

## SYNOPSIS

**autosv Mode**

**a** - mode line letter.

## DESCRIPTION

When this mode is enabled when the buffer is changed it will be automatically saved to a temporary file **$auto-time**(5) later.

Automatic saving for a buffer will not occur if

The buffer name starts with a '*', this is considered a temporary system buffer.

**$auto-time**(5) is set to 0, this disables auto-saving for all buffers.

The buffer does not a file name from which to generate a temporary file name. When this occurs the error message:

```
[Auto-writeout failure for buffer xxxxx]
```

MicroEmacs can not write to the generated temporary file name. When this occurs the error message:

```
[Auto-writeout failure for file xxxxx#]
```

On unlimited length file name systems (UNIX), the temporary file naming convention used for file `xxxxx`:

```
xxxxx -> xxxxx#
```

On systems with an `xxxxxxxx.yyy` file name (DOS etc), the following file naming convention is used:

```
xxxxxxxx     -> xxxxxxxx.###
xxxxxxxx.y   -> xxxxxxxx.y##
xxxxxxxx.yy  -> xxxxxxxx.yy#
xxxxxxxx.yyy -> xxxxxxxx.yy#
```

## NOTES

This mode MUST be enabled globally when the file is loaded for the file style to be correctly detected.

It is **strongly advised** that autosv mode is permanently enabled.

Auto-save files of URL files (i.e. "`ftp://...`" and "`http://...`") are written to the system's temporary directory. This avoids potentially slow auto-saves. This can however lead to recovery problems as the buffer name must be used to avoid auto-saving conflict with other buffers with the same base file name but different paths.

## SEE ALSO

Operating Modes, **$auto-time**(5), **backup**(2m), **find-file**(2), **ftp**(3).

Ref: autosv(2m) File: m2mod019.2 Date: 2025/08/28

## NAME

backup - Automatic file backup of last edit

## SYNOPSIS

### backup Mode

**B** - mode line letter.

## DESCRIPTION

**backup** mode, when enabled, performs an automatic backup of the last edit when a save file operation is performed. The backup file name is derived from the base file name and is written into the same directory as the source file, the backup file name can be obtained from the variable **$buffer-backup**(5).

On unlimited file name length systems the naming convention used depends on bit **0x400** of variable **$system**(5), if this bit is set then the system simulates a DOS style 8.3 file naming convention. If this bit is clear then variable **$kept-versions**(5) can be used to create multiple backup files.

Where an existing backup file already exists, then the old backup file is removed and replaced by the newer backup file. The naming convention for backup files is defined as follows:

On unlimited length file name systems (UNIX and some Windows systems) with a single backup file, the following file naming conventions are used for file `xxxxx`:

```
xxxxx -> xxxxx
```

On unlimited length file name systems with multiple backup files, the following file naming conventions are used for file `xxxxx`:

```
xxxxx -> xxxxx. ?
```

Where `?` is the backup number, the most recent backup file is always "`. 0 `".

On systems with an `xxxxxxxx.yyy` file name (MS-DOS etc), the following file naming conventions are used:

```
xxxxxxxx      -> xxxxxxxx.
xxxxxxxx.yyy  -> xxxxxxxx.yy
xxxxxxxx.yyyy -> xxxxxxxx.yyy
```

The environment variables **$MEBACKUPPATH**(5) and **$MEBACKUPSUB**(5) can be used to modify this behaviour.

## NOTES

**backup** is enabled by default.

Reference should also be made to **undo**(2) which allows previous edits to be removed. Also see **$auto-time**(5) and **autosv**(2m) which allows a timed backup of a running edit to be periodically performed.

The user is warned to be extra careful if files ending in ’ ’ or ’#’s are used, it is advisable to disable backup creation (see **global-mode**(2)) and auto-saving (`$auto-time = 0`). The author denies all responsibility (yet again) for any loss of data! Please be careful.

## SEE ALSO

Operating Modes, **$buffer-backup**(5). **$system**(5), **$kept-versions**(5), **$MEBACKUPPATH**(5), **$MEBACKUPSUB**(5), **buffer-mode**(2), **global-mode**(2), **undo**(2m), **autosv**(2m), **$auto-time**(5).

Ref: backup(2m) File: m2mod001.2 Date: 2022/06/12

## NAME

binary - Binary editor mode

## SYNOPSIS

**binary Mode**

**b** - mode line letter.

## DESCRIPTION

**binary** mode is enabled when a file is edited in binary mode. The mode is automatically enabled when a file is loaded as a binary file via **find-bfile**(3).

When a file is loaded using binary mode, every 16 bytes is converted into a line of text giving the hex address of the current position in the file, the bytes as hexadecimal numbers and all printable characters at the end of the line (all non-printable characters are displayed as a '**.**'). However, This format makes it memory hungry in that every 16 bytes of the file requires a 78 character line (approximately 5 times more memory!). For a more memory efficient binary mode see **rbin**(2m).

When writing out a file which has binary mode enabled the format of each line must have the binary mode format which is as follows:

```
<address>: XX XX XX XX XX .... XX XX | <text-form>
```

Only the hex values (the XX's) between the starting '*:*' marker and the terminating '—' character are used, the <*address*> and <*text-form*> are ignored.

## EXAMPLE

Given a single line MSDOS file:

```
Live long and prosper.
```

When loaded in using **binary** mode the following 2 line buffer will be produced:

```
00000000: 4C 69 76 65 20 6C 6F 6E 67 20 61 6E 64 20 70 72  |  Live long and pr
00000010: 6F 73 70 65 72 2E 0D 0A 1A                       |  osper....
```

Note the "0D 0A 1A" at the end, this is due to MSDOS's "\n\r" carriage returns and ^Z file termination, these characters are unprintable and are shown as '**.**' at the end of the line.

When saving a binary file, only the text between the '**:**' and '**|**' is considered and it may contain any number of hexadecimal numbers. The given file could be made UNIX compatible by editing the buffer to:

```
00000000: 4C 69 76 65 20 6C 6F 6E 67 20 61 6E 64 20 70 72  |  Live long and pr
00000010: 6F 73 70 65 72 2E 0D                             |  osper....
```

If the word "**long**" is required to be removed, the following would not work:

```
00000000: 4C 69 76 65 20 6C 6F 6E 67 20 61 6E 64 20 70 72  |  Live and pr
00000010: 6F 73 70 65 72 2E 0D 0A 1A                       |  osper....
```

The ASCII end letters are ignored, instead the following operation must be performed which removes the characters from the binary representation:

```
00000000: 4C 69 76 65 20 61 6E 64 20 70 72  |  Live long and pr
00000010: 6F 73 70 65 72 2E 0D 0A 1A                       |  osper....
```

One could be nasty by doing the following:

```
00000000: 4C 69 76 65 20 73 68 6F 72 74 20 61 6E 64 20 |
00000010: 64 6F 6E 27 74 20 70 72 6F 73 70 65 72 2E 0D 0A 1A          |
```

("Live short and don't prosper").

## NOTES

**rbin** and **binary** modes are mutually exclusive, i.e. they cannot both be enabled at the same time.

## SEE ALSO

Operating Modes, **find-bfile**(3), **find-file**(2), **rbin**(2m).

## NAME

cr - File's carriage return (CR) style

## SYNOPSIS

**crlf Mode**

**c** - mode line letter.

## DESCRIPTION

When enabled **cr** indicates that a carriage return should be written out at the end of the line. Typically enabled independently for different line ending styles used in DOS and UNIX. When 1 then **r** is included in the line ending; when 0 then **r** is not included in the line ending.

## NOTES

This mode only effects the style in which the buffer is written if **auto**(2m) mode is enabled. **cr** mode is used in conjunction with **lf**(2m) to control the line feed.

## SEE ALSO

Operating Modes, **auto**(2m), **lf**(2m), **ctrlz**(2m), **save-buffer**(2), **find-file**(2), **$buffer-fmod**(5).

Ref: cr(2m) File: m2mod011.2 Date: 2022/06/12

## NAME

crypt - Encrypted file mode

## SYNOPSIS

**crypt Mode**

**Y** - mode line letter.

## DESCRIPTION

**crypt** mode enables encrypted files to be loaded and saved for security purposes. The key can be set at any time using the command **set-encryption-key**(2). Warning, take care if setting this as a global mode, it can have side-effects.

The encryption algorithm uses AES in a chained block form and 256 bit key, at time of writing this should be considered as secure as the password used. When you write out text, if `crypt` mode is active and there is no encryption key, MicroEmacs will ask:

```
<file-name> password:
```

Type in a word or phrase of at least five and up to 128 characters for the encryption to use. If you look at the file which is then written out, it should be indistinguishable to random data, also if you save the file again, without any changes to the unencrypted data, the content be completely different. To read such a file later, use **find-cfile**(3) to load *ciphertext* files, you will be asked for the password before the file is read.

## SEE ALSO

Operating Modes, **buffer-mode**(2), **find-cfile**(3), **global-mode**(2), **set-encryption-key**(2), **AES**(2).

Ref: crypt(2m) File: m2mod003.2 Date: 2023/02/21

## NAME

ctrlz - File's termination style

## SYNOPSIS

**ctrlz Mode**

**z** - mode line letter.

## DESCRIPTION

When enabled **ctrlz** indicates that an MS-DOS style '**ctrl-z**' file termination character should be written out. When clear, a UNIX style of no termination character should be used.

## NOTES

This mode only effects the style in which the buffer is written if **auto**(2m) mode is enabled.

## SEE ALSO

Operating Modes, **auto**(2m), **cr**(2m), **lf**(2m), **save-buffer**(2), **find-file**(2), **$buffer-fmod**(5).

Ref: ctrlz(2m) File: m2mod030.2 Date: 2022/06/12

## NAME

cursor - Buffer's cursor visibility

## SYNOPSIS

**cursor Mode**

**C** - mode line letter.

## DESCRIPTION

When enabled the text cursor is drawn while the buffer is current, when disabled the cursor is hidden. **cursor** is enabled by default. As the mode is a property of the buffer, the cursor re-appears as soon as another buffer is made current; there is no need to restore the mode on leaving the buffer.

The mode is typically disabled by macros which draw their own display, see **Metris**(3) and **Patience**(3), where a flashing text cursor is a distraction.

The cursor is always drawn when MicroEmacs is prompting for input on the message line, regardless of the current buffer's **cursor** mode, so that the user can see what they are typing.

In a buffer created by **ipipe-shell-command**(2) the mode is set and cleared by the child process using the terminal sequences **ESC [ ? 25 h** and **ESC [ ? 25 l** respectively. A child which exits while the cursor is hidden leaves the mode disabled, exactly as it would leave a real terminal.

## NOTES

Cursor hiding is not supported on all platforms and does nothing in NanoEmacs 2026.

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2), **ipipe-shell-command**(2), **$cursor-blink**(5).

Ref: cursor(2m) File: m2mod007.2 Date: 2026/08/13

## NAME

edit - Buffer has be changed

## SYNOPSIS

**edit Mode**

**e** - mode line letters.

## DESCRIPTION

**edit** mode indicated that the buffer has been edited. Many commands and typing 'edit' the current buffer, automatically setting this mode. Commands which save these edits, such as **save-buffer**(2), automatically remove this mode.

A '*' character, 3 characters from the left on the mode line is used to indicate that this mode is set, see **$mode-line**(5). **list-buffers**(2) also displays the state of this mode in its output, as a '*' in the second column.

When this mode is set and **undo**(2m) mode is enabled, the **undo**(2) command can be used to undo all edits and the removal of this mode.

## SEE ALSO

Operating Modes, **save-buffer**(2), **undo**(2), **list-buffers**(2), **$mode-line**(5), **undo**(2m).

Ref: edit(2m) File: m2mod029.2 Date: 2022/06/12

## NAME

exact - Searching and sorting case sensitivity

## SYNOPSIS

**exact Mode**

**E** - mode line letter.

## DESCRIPTION

**exact** mode sets the searching and line sorting commands to case sensitive when enabled (case insensitive when disabled). See **search-forward**(2) and **sort-lines**(2).

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2), **search-forward**(2), **sort-lines**(2).

Ref: exact(2m) File: m2mod004.2 Date: 2022/06/12

## NAME

fence - Auto fence matching mode

## SYNOPSIS

### fence Mode

**f** - mode line letter.

## DESCRIPTION

**fence** mode can be used to enable or disable the automatic displaying of and open fence when the corresponding closing fence is typed. When the mode is enabled and the closing fence (')' ']' or '}') is typed the cursor is temporarily move to the position of the opening fence. The duration of the move can be controlled by the **$pause-time**(5) variable; any user input interrupts the display.

If the buffer is configured to use a *C* style auto **indent**(2), the search algorithm used is '*C*' aware and if a matching fence is not found then the bell is rung as a warning. Otherwise any closing fence which cannot be matched is ignored.

The cursor can be moved to the matching fence using the **goto-matching-fence**(2) command.

## NOTES

The following are considered closing fences:

```
}  )  ] > */ #endif
```

These are match with the following opening fences respectively:

```
{  (  [ < /* #if
```

In MicroEmacs the use of fence mode has been depreciated. Matching fence hilighting has been implemented as a macro and configured via the Platform page of **user-setup**(3).

## SEE ALSO

Operating Modes, **$pause-time**(5), **user-setup**(3), **goto-matching-fence**(2), **indent**(2).

Ref: fence(2m) File: m2mod032.2 Date: 2022/06/12

## NAME

hide - Hide buffer

## SYNOPSIS

**hide Mode**

**H** - mode line letter.

## DESCRIPTION

This mode can only be set on a buffer and when enabled the buffer is effectively hidden from the user. When set the buffer is hidden from the buffer completion list used by commands such as **find-buffer**(2), the buffer is also ignored by commands **list-buffers**(2), **save-some-buffers**(2) and **delete-some-buffers**(2).

## SEE ALSO

Operating Modes, **find-buffer**(2), **list-buffers**(2).

Ref: hide(2m) File: m2mod022.2 Date: 2022/06/12

## NAME

indent - Automatic indentation

## SYNOPSIS

**indent Mode**

**I** - mode line letter.

## DESCRIPTION

**indent** mode, when enabled, ensures that a new text line is automatically indented to the same left hand column as the previous line's first non-white character. If the previous line contains no non-white characters then the line will not be indented. Automatic indentation is disabled when using *center* or *right* justification. **Indent** is usually used in conjunction with **wrap**(2m) and **justify**(2m).

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2), **wrap**(2m) **justify**(2m).

Ref: indent(2m) File: m2mod005.2 Date: 2022/06/12

## NAME

justify - Justification Mode

## SYNOPSIS

**justify Mode**

**J** - mode line letter.

## DESCRIPTION

**justify** mode, when enabled, performs paragraph justification as designated by **$fill-mode**(5) - capable of *left*, *right*, *both* or *center* justification of text. Justify removes all white spaces at the end of the line, if there are no non-white characters on the line then the line is always left empty. If the justification method is *center* or *right* then all white spaces are removed at the beginning of the line. If the line is longer than the **$buffer-fill-col**(5) or the method is *left* then nothing more is done, else the line is appropriately justified. The method used is set by the variable **$buffer-fill-mode**(5). Justify is usually used in conjunction with **wrap**(2m) and **indent**(2m).

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2), **wrap**(2m) **indent**(2m), **$buffer-fill-col**(5), **$buffer-fill-mode**(5), **$fill-col**(5), **$fill-mode**(5).

Ref: justify(2m) File: m2mod006.2 Date: 2022/06/12

## NAME

lf - File's line feed style

## SYNOPSIS

### lf Mode

**l** - mode line letter.

## DESCRIPTION

When enabled **lf** indicates that a line feed should be written out including the **n** character. When set then a **n** is included on each line ending. When clear then **n** is not included in the line ending.

## NOTES

This mode only effects the style in which the buffer is written if **auto**(2m) mode is enabled. **lf** mode is used in conjunction with **cr**(2m) to control the carriage return.

## SEE ALSO

Operating Modes, **auto**(2m), **cr**(2m), **ctrlz**(2m), **save-buffer**(2), **find-file**(2), **$buffer-fmod**(5).

Ref: lf(2m) File: m2mod024.2 Date: 2022/06/12

## NAME

lock - Pipe cursor position lock

## SYNOPSIS

**lock Mode**

**k** - mode line letter.

## DESCRIPTION

This mode can only be used while an incremental pipe (started by **ipipe-shell-command**(2)) is running in the current buffer, denoted by the **pipe**(2m) being set. When this mode is enabled and MicroEmacs buffer cursor is at the same location as the process shell cursor, the buffer cursor is automatically moved with the shell cursor.

This mode is automatically enabled for a piped buffer.

## SEE ALSO

Operating Modes, **ipipe-shell-command**(2), **pipe**(2m).

Ref: lock(2m) File: m2mod021.2 Date: 2025/08/28

## NAME

magic - Regular expression search

## SYNOPSIS

### magic Mode

**M** - mode line letter.

## DESCRIPTION

**magic** mode enables the regular expression search capability used in the search and the replace commands such as **search-forward**(2) and **query-replace-string**(2).

In the magic mode of MicroEmacs, certain characters gain special meanings when used in a search pattern. Collectively they are know as regular expressions, and a limited number of them are supported in MicroEmacs. They grant greater flexibility when using the search commands (note that they also affect **isearch-forward**(2) commands).

The symbols that have special meaning in magic mode are `^`, `$`, `.`, `\|`, `*`, `[ ]`, `\( \)`, `\{ \}` and `\`.

The characters `^` and `$` fix the search pattern to the beginning and end of line, respectively. The `^` character must appear at the beginning of the search string, and the `$` must appear at the end, otherwise they loose their meaning and are treated just like any other character. For example, in magic mode, searching for the pattern "`t$`" would put the cursor at the end of any line that ended with the letter '`t`'. Note that this is different than searching for "`t<NL>`", that is, '`t`' followed by a newline character. The character `$` (and `^`, for that matter) matches a position, not a character, so the cursor remains at the end of the line. But a newline is a character that must be matched, just like any other character, which means that the cursor is placed just after it - on the beginning of the next line.

The character '`.`' has a very simple meaning - it matches any single character, except the newline. Thus a search for "`bad.er`" could match "`badger`", "`badder`" (slang), or up to the '`r`' of "`bad error`".

The character `*` is known as closure, and means that zero or more of the preceding character will match. If there is no character preceding, `*` has no special meaning, and since it will not match with a newline, `*` will have no special meaning if preceded by the beginning of line symbol `^` or the literal newline character `<NL>`. The notion of zero or more characters is important. If, for example, your cursor was on the line

```
This line is missing two vowels.
```

and a search was made for "`a*`", the cursor would not move, because it is guaranteed to match no letter '`a`', which satisfies the search conditions. If you wanted to search for one or more of the letter 'a', you would search for "`aa*`", which would match the letter `a`, then zero or more of them, note that this pattern is better searched using "`a+`".

The character "+" is the same as "`*`" except that it searches for one or more occurrences of the preceding character.

The character `[` indicates the beginning of a character class. It is similar to the *any* (`.`) character, but you get to choose which characters you want to match. The character class is ended with the character `]`. So, while a search for "`ba.e`" will match "`bane`", "`bade`", "`bale`", "`bate`", et cetera, you can limit it to matching "`babe`" and "`bake`" by searching for "`ba[bk]e`". Only one of the characters inside the `[` and `]` will match a character. If in fact you want to match any character except those in the character class, you can put a `^` as the first character. It must be

the first character of the class, or else it has no special meaning. So, a search for `[^aeiou]` will match any character except a vowel, but a search for `[aeiou^]` will match any vowel or a `^`. If you have a lot of characters in order that you want to put in the character class, you may use a dash (`-`) as a range character. So, `[a-z]` will match any letter (or any lower case letter if exact mode is on), and `[0-9a-f]` will match any digit or any letter '`a`' through '`f`', which happen to be the characters for hexadecimal numbers. If the dash is at the beginning or end of a character class, it is taken to be just a dash.

The `?` character provides a simple zero or one occurrence test of the previous character e.g. "`ca?r`" matches "`cr`" and "`car`", it will not match "`caar`".

Where a previous item has a range of repetitions then the `\{N,M\}` syntax may be used to denote the minimum and maximum iterations of the previous item. Where a set quantity of repetitions is required then the simpler syntax of `\{N\}` may be used. i.e. "`ca\{2\}r`" matches "`caar`", "`ca\{2,3\}r`" matches "`caar`" and "`caaar`".

The escape character `\` is for those times when you want to be in magic mode, but also want to use a regular expression character to be just a character. It turns off the special meaning of the character. So a search for "`it\.`" will search for a line with "`it.`", and not "`it`" followed by any other character. The escape character will also let you put `^`, `-`, or `]` inside a character class with no special side effects.

In search-replace strings the `\(` `\)` pair may be used to group characters for in the search string for recall in the replacement string. The `\(` `\)` bracket pair is recalled using `\1`-`\9` in the replace string, `\1` is the first pair, `\1` the second and so on. Hence to replace `%dgdg%name%dhdh%` with `%dgdg%names%dhdh%` then we could use the following search replace string `\(%[a-z]+%\)\([a-z]*\` replacing with `\1\2s\3`.

`\0` in the replace string implies the whole string.

A summary of magic mode special characters are defined as follows:

| | |
|---|---|
| **^** | Anchor search at beginning of line |
| **$** | Anchor search at end of line |
| **.** | Match any character except <NL> |
| ***** | Match zero or more occurrences of the preceding item. |
| **\|** | Match either/or i.e. `car\|bike` matches the work `car` and matches the word `bike`. |
| **+** | Match one or more occurrences of the preceding item. |
| **?** | Match zero or one occurrences of the preceeding item. |
| **[]** | Match a class of characters (`[a-z]` would be all alphabetics) |
| **** | Take next literally |
| **\{*N*,*M*\}** | Match a minimum of *N* occurrences and maximum of *M* occurrences of the preceeding item. |
| **\{*N*\}** | Match a *N* occurrences of the preceeding item. |
| **\(...\)** | Delimit pattern to replicate in replace string. Max of 9 allowed. Called in replace string with `\1`,..,`\9`. 1 being 1st etc. `\0` or `\&` in the replace string is the whole string. i.e.<br>Search: **\(ab\)\(dc\)**<br>Replace: **\1\2 \1\2**<br>on "**abdc**" => "**abdc abdc**" |

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2), **query-replace-string**(2), **search-forward**(2).
Regular Expressions

Ref: magic(2m) File: m2mod009.2 Date: 2025/08/28

## NAME

nact - Buffer not active

## SYNOPSIS

**nact Mode**

**n** - mode line letter.

## DESCRIPTION

This mode can not be set and is used to indicate that the buffer has not been activated, i.e. the buffer has not been displayed in a window. If the buffer is linked to a file but has not been displayed, so is not active, the file will not have been loaded into the buffer.

The **list-buffers**(2) command output denotes active buffers with a '@' character in the left hand column, inactive buffers have a ' '.

This mode can not be tested using the more usual **&bmode**(4) macro command as it only operates on the current buffer as which point the mode cannot be set. Instead the **&nbmode**(4) macro command must be used.

## SEE ALSO

Operating Modes, **list-buffers**(2), **&nbmode**(4), **&bmode**(4).

Ref: nact(2m) File: m2mod023.2 Date: 2022/06/12

## NAME

narrow - Buffer contains a narrow

## SYNOPSIS

**narrow Mode**

**N** - mode line letter.

## DESCRIPTION

This mode can not be set and is used to indicate whether the buffer contains a narrow, created by the **narrow-buffer**(2) command. The the location of any narrow region lines contained within the buffer may be determined with **$line-flags**(5).

## SEE ALSO

Operating Modes, **$line-flags**(5), **narrow-buffer**(2), **narrow-search-backward**(3), **narrow-search-forward**(3).

Ref: narrow(2m) File: m2mod025.2 Date: 2022/06/12

## NAME

over - Over-strike Mode

## SYNOPSIS

**over Mode**

**O** - mode line letter.

## DESCRIPTION

**over** mode, when enabled, over writes existing text in a buffer as opposed to inserting text. over maintains the position of text aligned with `tab` characters.

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2).

Ref: over(2m) File: m2mod010.2 Date: 2022/06/12

## NAME

pipe - Incremental Pipe running

## SYNOPSIS

**pipe Mode**

**P** - mode line letter.

## DESCRIPTION

This mode indicates whether an incremental pipe (started by **ipipe-shell-command**(2)) is running in the current buffer. This mode is automatically set and can not be changed by the user. pipe modes main use is in macros.

Modes **lock**(2m) and **wrap**(2m) effect the output behavior of a piped command.

## SEE ALSO

Operating Modes, **ipipe-shell-command**(2), **lock**(2m), **wrap**(2m).

Ref: pipe(2m) File: m2mod020.2 Date: 2022/06/12

## NAME

rbin - Reduced binary editor mode

## SYNOPSIS

**rbin Mode**

**r** - mode line letter.

## DESCRIPTION

**rbin** mode is enabled when a file is edited in reduced binary mode. The mode is automatically enabled when a file is loaded as a binary data file via **find-file**(2).

When a file is loaded using **rbin** mode, every 256 bytes is converted into a line of text, the line is a single list of hex numbers 512 characters long, 2 bytes for each character. This format is not very user friend unlike **binary**(2m) mode, but is much more memory efficient (requiring approximately 2 times more memory than the file size).

When writing out a file which has rbin mode enabled the format of each line must be correct, namely an even number of hex numbers with no other characters.

## EXAMPLE

Given a single line MSDOS file:

```
Live long and prosper.
```

When loaded in using **binary** mode the following 2 line buffer will be produced:

```
4C697665206C6F6E6720616E642070726F737065722E0D0A1A
```

Note the "`0D 0A 1A`" at the end, this is due to MSDOS's "`\n\r`" carriage returns and `^Z` file termination. The given file could be made UNIX compatible by editing the buffer to:

```
4C697665206C6F6E6720616E642070726F737065722E0D
```

## NOTES

**rbin** and **binary** modes are mutually exclusive, i.e. they cannot both be enabled at the same time.

## SEE ALSO

Operating Modes, **find-file**(2), **binary**(2m).

Ref: rbin(2m) File: m2mod031.2 Date: 2022/06/12

## NAME

tab - Tabulation mode

## SYNOPSIS

### tab Mode

**T** - mode line letter.

## DESCRIPTION

**tab** mode, when enabled, simulates all tab stops with spaces. This allows '*variable*' tab sizes and fixes indentation. If **tab** mode is not enabled literal tab characters are inserted, their displayed width is controlled on a buffer basis with the variable **$buffer-tab-width**(5).

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2). **$buffer-tab-width**(5), **$tab-width**(5), **tabs-to-spaces**(3).

Ref: tab(2m) File: m2mod012.2 Date: 2022/06/12

## NAME

time - File time stamping

## SYNOPSIS

**time Mode**

**t** - mode line letter.

## DESCRIPTION

**time** mode, when enabled, performs automatic time stamping of files on file write operations. A time stamp string, defined by **$timestamp**(5) is searched for in the file and updated with the current data and time information, providing a record in the file of the last edit.

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2). **$timestamp**(5).

Ref: time(2m) File: m2mod013.2 Date: 2022/06/12

## NAME

undo - Retain edit modifications

## SYNOPSIS

**undo Mode**

**U** - mode line letter.

## DESCRIPTION

**undo** mode, when enabled, stores a history of all user edits so that the command **undo**(2) may be used to undo the last *n* edits to a buffer. If this mode is not enabled the **undo** command has no effect.

Obviously memory is required to store this information, particularly storing deleted, reformed or replaced text, users editing large files or operating in restricted memory environments may wish to use this mode selectively.

## NOTES

The **undo** information is flushed, and is effectively lost, when a save operation is performed on the buffer.

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2). **undo**(2).

Ref: undo(2m) File: m2mod014.2 Date: 2022/06/12

## NAME

view - Read only

## SYNOPSIS

**view Mode**

**V** - mode line letters.

## DESCRIPTION

**view** mode sets the buffer to read-only, disabling the ability to alter the contents of the buffer. This mode is automatically set for any files attributed with a read-only status on the file system when read into MicroEmacs. Files loaded via **view-file**(2) are also assigned **view** mode.

While in **view** mode, any attempt to alter the buffer contents results in the following message:

```
[Key Illegal in view Mode]
```

## SEE ALSO

Operating Modes, **buffer-mode**(2), **global-mode**(2), **view-file**(2).

Ref: view(2m) File: m2mod016.2 Date: 2025/08/28

## NAME

wrap - Line wrap entered text

## SYNOPSIS

**wrap Mode**

**W** - mode line letters.

## DESCRIPTION

**wrap** mode causes automatic text wrapping when text passes then fill column (see **$buffer-fill-col**(5)), allowing text to be entered non-stop on a standard screen without bothering to use the `RETURN` key.

**wrap** mode is usually used in conjunction with the **justify**(2m) and **indent**(2m) modes for editing text documents.

**wrap** mode also automatically wraps long lines in the output of an **ipipe-shell-command**(2) to the width of the MicroEmacs window.

## SEE ALSO

Operating Modes, **$buffer-fill-col**(5), **$fill-col**(5), **buffer-mode**(2), **global-mode**(2), **ipipe-shell-command**(2) **justify**(2m), **indent**(2m), **pipe**(2m).

Ref: wrap(2m) File: m2mod017.2 Date: 2022/06/12

## NAME

xlate - Translated buffer editor mode

## SYNOPSIS

**xlate Mode**

**X** - mode line letter.

## DESCRIPTION

**xlate** mode is enabled when a buffer needs to be translated before saving. The mode works in conjunction with variable **$buffer-xlate**(5) which defines the translation required.

The mode is typically enabled by the **translate-buffer**(2) command which is used to convert a file loaded using **rbin** mode into a more editable form.

See documentation on variable **$buffer-xlate**(5) for a better understanding of the types of translation preformed.

## NOTES

**xlate**, **rbin** and **binary** modes are mutually exclusive, i.e. they should not be enabled at the same time.

When a file writing command is called on a buffer with **xlate** mode enabled, the command will first execute its normal validity checks, if these succeed the command will internally translate the buffer, as defined by **$buffer-xlate**, creating a temporary second copy of the buffer, it is this version that is then written out and then freed. This leaves the original buffer unchanged, although modes like **edit**(2m) will have been altered as normal.

## SEE ALSO

Operating Modes, **translate-buffer**(2), **find-file**(2), **$buffer-xlate**(5), **rbin**(2m).

Ref: xlate(2m) File: m2mod002.2 Date: 2025/03/25

## NAME

abbrev-list - Create a list of abbreviations

## SYNOPSIS

**abbrev-list**

## DESCRIPTION

**abbrev-list** operates opens a new pop-up window and displays a list of abbreviations that are available. Selecting an abbreviation within the window with the mouse inserts the abbreviation into the current buffer.

The abbreviations are defined by the XXX.eaf abbreviation file, where `XXX` is the buffer template. The abbreviation file is defined to the buffer using **buffer-abbrev-file**(2) which is configured and loaded by **buffer-init**(3).

When the **toolbar**(3) is enabled then the command does not produce a pop-up window, the abbreviation list may be found (or added) to the toolbar.

## SEE ALSO

**buffer-abbrev-file**(2), **buffer-init**(3), **eaf**(8), **expand-abbrev**(2), **toolbar**(3).

Ref: abbrev-list(3) File: m3mac067.3 Date: 2022/06/12

## NAME

alarm - Set an alarm

## SYNOPSIS

**alarm** "*message*" "*hours*" "*minutes*"

## DESCRIPTION

**alarm** creates an alarm notification which displays "*message*" to the user. The alarm time is specified as an elapsed time in "*hours*" and "*minutes*" from the time of creation.

The message is printed on the screen using **osd**(2).

## NOTES

**alarm** is a macro defined in `osdmisc.emf`.

## SEE ALSO

**osd**(2).

Ref: alarm(3) File: m3mac011.3 Date: 2022/06/12

## NAME

Arkanoid - MicroEmacs version of the bat and ball wall game

## SYNOPSIS

**Arkanoid**

## DESCRIPTION

Knock down the wall of bricks with the ball, keeping the ball in play with the bat along the bottom of the screen. Clear every brick and the next wall is loaded. Let the last ball fall past the bat and a life is lost; lose them all and the game is over.

The bat is steered with the cursor keys or with the mouse, whichever was used last. Where the ball strikes the bat decides where it goes - a hit out towards either end sends the ball away at a shallow angle, a hit near the middle sends it back steeply. The speed is the same whichever part of the bat is used.

There are eight walls, which repeat once all eight have been cleared. The ball starts a little faster on each new level and creeps faster still the longer a single wall takes, so the later levels are considerably quicker than the early ones. The current speed is shown on the status line.

Bricks drawn as solid blocks cannot be destroyed and are not counted; they simply deflect the ball. White bricks take two hits, darkening after the first. Every other brick takes one hit.

A brick scores between 60 and 110 points depending on its colour; a white brick scores 50 for the first hit and 100 for the second. A good capsule is worth 200 and a bad one 25. Clearing the wall scores 500 times the level number.

## CAPSULES

Destroying a brick sometimes releases a capsule which drifts down the screen. Catch it with the bat to collect it. Capsules **stack**: a second **D** splits every ball again, a second **E** widens the bat further, a second **M** or **G** adds to the time remaining, and **E** and **R** wind the bat back and forth against each other. Everything collected lasts until the ball is lost or the wall is cleared.

The good capsules are:

**E** Expand the bat.

**C** Catch - the ball sticks to the bat until **space** is pressed.

**L** Laser bat - **space** fires.

**S** Slow the ball down.

**D** Split every ball in play into three.

**P** An extra life.

**M** Mega ball - smashes straight through the wall without bouncing.

**G** A guard along the floor, for a while.

**T** Twin bat - two shorter bats with a gap down the middle.

**B** Break out of the level immediately. Rare.

The bad capsules are all drawn in red:

**R** Reduce the bat.

**F** Speed the ball up.

**V** Reverse the bat controls, for a while.

## KEYS

The keys used to control **Arkanoid** are:

**left** or **right**
Move the bat.

**mouse** The bat will follow your column position of the mouse.

**space** Launch the ball, release a held ball, or fire the laser.

**button 1**
The same as **space**.

**p** Pause the game.

**?** or **h** Show the key help.

> Skip the level, for those who want to see the end.

**q** or **C-g**
Quit the game.

## NOTES

**Arkanoid** is a macro defined in `arkanoid.emf`.

The game uses whatever room the window gives it and adjusts if the window is resized while playing. If the window is too small to lay out a wall the game declines to start and says so on the message line.

## SEE ALSO

Games, **Balls**(3), **Metris**(3).

Ref: Arkanoid(3) File: m3mac110.3 Date: 2026/08/25

## NAME

ascii-time - Return the current time as a string

## SYNOPSIS

**ascii-time** [ "*format*" ]

## DESCRIPTION

**ascii-time** returns the current time as a formatted string in `#p9` which is equivalent to `#l9` for the calling macro. The format of the resultant string is determined by the variable the parameter *format*, if given, otherwise by the variable **%ascii-time**. The format is defined using the following escape characters:

`%%` - `%` char
`%a` - Replaced by abbreviated week day name (3 letters)
`%A` - Replaced by full week day name
`%b` - Replaced by abbreviated month name (3 letters)
`%B` - Replaced by full month name
`%d` - Replaced by two digit day of month (01 to 31)
`%e` - As with `%d` except a leading 0 is replaced by a space ( 1 to 31)
`%H` - Replaced by two digit hour of the day (00 to 23)
`%i` - Replaced by day of month (1 to 31)
`%k` - Replaced by hour of the day (0 to 23)
`%J` - Replaced by day of year (0 to 365)
`%j` - Replaced by three digit day of year (000 to 365)
`%M` - Replaced by two digit minute of the hour (00 to 59)
`%m` - Replaced by two digit month (01 to 12)
`%N` - Replaced by minute of the hour (0 to 59)
`%o` - Replaced by month number (1 to 12)
`%S` - Replaced by two digit second of the minute (00 to 59)
`%v` - Replaced by second of the minute (0 to 59)
`%w` - Replaced by day of the week (0 to 6 Sunday=0)
`%y` - Replaced by the last two digits of the year (00 to 99)
`%Y` - Replaced by the year as a decimal number (e.g. 2024)
`%*` - All other characters are printed literally.

The default value of **%ascii-time** when not defined is:

```
"%Y-%m-%d %H:%M:%S"
```

To change the default date layout then **%ascii-time** should be defined in *user*.**emf**. i.e.

```
set-variable %ascii-time "%a %b %d %H:%M:%S %Y"
```

## EXAMPLE

The following is taken from etfinsrt.emf, it uses **ascii-time** in replacing "`$ASCII_TIME$`" with the current.

```
0 define-macro insert-template
```

```
    .
    .
    ; Change the create date $ASCII_TIME$.
    beginning-of-buffer
    ; Get ASCII time in #l9
    ascii-time
    !force replace-string "\\$ASCII_TIME\\$" #l9
    .
    .
!emacro
```

## NOTES

**ascii-time** is a macro defined in `utils.emf`.

The default format was changed in 2025 to be ISO 8601 compliant. Some of the escape characters, such as **%#d**, were also changed to be more compliant with **$mode-line**(5) and C's **strftime**(3) function.

The current date and time can be easily inserted into the current buffer by using the main Edit menu's **Insert Date/Time** feature.

## SEE ALSO

**time**(2m), **$timestamp**(5), **major-mode**(2), **insert-template**(3), **$buffer-fhook**(5), **$mode-line**(5), **&find**(4).

Ref: ascii-time(3) File: m3mac019.3 Date: 2026/01/18

## NAME

auto-spell - Auto-spell support
auto-spell-menu - Auto-spell context menu
auto-spell-buffer - Auto-spell whole buffer
auto-spell-reset - Auto-spell hilight reset
auto-spell-ignore - Auto-spell ignore current word

## SYNOPSIS

*n* **auto-spell**
**auto-spell-buffer**
**auto-spell-reset**
*n* **auto-spell-ignore**

## DESCRIPTION

**auto-spell** enables and disables the auto spell checking of the current buffer. Auto spell detects word breaks as you type and checks the spelling of every completed word hilighting any erroneous words in the error color scheme (usually red).

The argument *n* determines whether auto-spell is enabled or disabled, a positive argument enables and a negative argument disables. If no argument or *0* is supplied the auto-spell state is toggled.

**auto-spell-menu** opens the auto-spell context menu which lists suggested corrections for the current word and other useful auto-spell commands and configuration options. This menu can also be accessed via the main mouse context menu when the current buffer has auto spell enabled.

**auto-spell-buffer** checks all words within the current buffer for spell, hilighting any unknown or miss-spelled words found.

**auto-spell-reset** resets the buffer hilighting scheme, removing any added erroneous words.

**auto-spell-ignore** gets the current word and deletes the erroneous hilighting and adds the word to the current temporary ignore dictionary, auto-spell and the spelling-checker will now ignore the word. If an argument **n** of 2 is given to the command the word is added to the users personal dictionary instead of the temporary ignore dictionary so the word is 'ignored' in all future sessions of MicroEmacs as well.

## NOTES

**auto-spell**, **auto-spell-buffer**, **auto-spell-reset** and **auto-spell-ignore** are macros defined in `spellaut.emf`.

## SEE ALSO

**user-setup**(3), **spell-buffer**(3), **spell**(2).

Ref: auto-spell(3) File: m3mac039.3 Date: 2022/06/12

## NAME

Balls - MicroEmacs bouncing ball playground

## SYNOPSIS

**Balls**

## DESCRIPTION

A playground rather than a game - there is nothing to win. Balls fall, bounce off the walls and floor, roll, come to rest, and optionally bounce off each other. Up to 100 balls may be in the air at once.

Almost everything about the simulation can be changed while it is running: the strength and direction of gravity, how much speed a ball keeps when it bounces, whether balls collide with one another, and whether there is a bat to knock them about with. The current settings are shown along the status line, together with the frame rate.

Balls may be picked up and thrown by hand. Hold **button 1** over a ball to grab it, move the pointer, and let go - the ball leaves with the speed of the throw. **Button 3** drops a new ball wherever the pointer is.

The pointer may also be turned into a lump of gravity of its own with **a**, either pulling the balls towards it or pushing them away.

## KEYS

The keys used to control **Balls** are:

**?** or **h** Show or hide the key help.

**space** Cycle the bat: off, across, upright, off.

**n** or **x** Add or remove a ball.

**s** Change the size of the next ball added: any size, tiny, small, medium, large.

**r** Reset, back to the starting set of balls.

**p** Pause.

**.** Run a single frame, for watching a collision resolve.

**a** The pointer as a lump of gravity: off, pull, push, off.

**-** or **=** Weaken or strengthen that pull. **+** does the same as **=**.

**g** Cycle gravity: normal, moon, none, upside down.

**wheel** Trim gravity, one unit a click.

**[** or **]** Less or more bounce.

**c** Ball to ball collisions on or off.

**button 1**
Drag a ball to pick it up, let go to throw it.

**button 3**
: Drop a new ball under the pointer.

**q**, **esc** or **C-g**
: Quit.

## NOTES

**Balls** is a macro defined in `balls.emf`.

The simulation runs in whatever room the window gives it and adjusts if the window is resized. Adding a great many balls with collisions turned on will lower the frame rate.

## SEE ALSO

Games, **Arkanoid**(3).

Ref: Balls(3) File: m3mac111.3 Date: 2026/08/25

## NAME

bin-copy-region - Binary supporting copy region
bin-yank - Binary supporting kill buffer insert

## SYNOPSIS

*n* **bin-copy-region** (**esc A-w**)
*n* **bin-yank** (**A-C-y**)

## DESCRIPTION

The **bin-copy-region** command operates the same as the **copy-region**(2) command except it adds the region as a hex strings, 2 hex characters per byte of data. If the current buffer has **binary**(2m) or **rbin**(2m) mode enabled then the hex bytes of the underlying binary file are extracted from the region, whereas if neither of these modes are enabled the region text is converted into a hex string (respecting end of line termination modes). As this command works with the current buffer modes, identifying the type of data being added is not an issue.

The **bin-yank** command operates the same as the **yank**(2) command except it inserts the kill buffer in the most appropriate way considering the format of the data in the kill buffer and the modes of the current buffer. If the kill buffer contains a hex string or binary mode compatible data then the kill buffer is deemed to contain binary data, otherwise it is considered plain text. If the current buffer has **binary** or **rbin** mode enabled then the kill buffer is inserted as binary data in the form of appropriately formatted hex strings, otherwise the data is converted to text if needed and inserted.

As the source of the kill buffer data is unknown **bin-yank** can misidentify the format, for example '`dead`' is both a 4 letter word and a 2 byte hex string. Priority is given to the hex string so if it is used in conjuction with **bin-copy-region** the result should always be correct.

## NOTES

These commands and their global bindings are only defined when the first binary file is loaded.

When inserting binary data into a non-binary buffer bytes may be lost as they cannot be represented in a text buffer, for example the null character (' ') cannot be inserted as it is a string termination character so is discarded. **bin-yank** will report how many characters had to be discarded.

## SEE ALSO

**binary**(2m), **rbin**(2m), **copy-region**(2), **yank**(2), **$buffer-indent**(5).

Ref: bin-copy-region(3) File: m3mac094.3 Date: 2024/05/08

## NAME

browse - Browse a file or buffer in view mode

## SYNOPSIS

**browse**

## DESCRIPTION

**browse** is a simple utility that assists buffer viewing by mapping the basic keys to navigation functions. **browse** may be invoked on any buffer.

On invoking **browse** then the buffer is placed into a simple viewing mode allowing the contents of the file to be viewed and the buffer cannot be edited. The key bindings that are recognized include:

`SPACE`, `RETURN` - Next page.
`BACKSPACE` - Previous page.
`C-n` - Scroll down .
`C-p` - Scroll up.
`C-f` - Scroll right.
`C-b` - Scroll left.
`C-g` - Quit browse mode.
`/` - Search forward, initiates a forwards search.
`w` - Toggle wrap mode, wraps long lines to screen width.

To quit browse mode then issue a **C-g** *abort* command.

## NOTES

**browse** is a macro implemented in `osdmisc.emf`.

## SEE ALSO

**view-file**(2).

Ref: browse(3) File: m3mac079.3 Date: 2022/06/12

## NAME

buffer-init - Initialize a buffer language template.
buffer-init-hooks - Initialize the buffer language template hooks.
buffer-init-fhook - Initialize the buffer file hooks.

## SYNOPSIS

*n* **buffer-init** [*etf* ]
**buffer-init-hooks**
**buffer-init-fhook** *name*

## DESCRIPTION

The **buff-init** commands prepare the operating modes and binding of a buffer for the assigned major-mode. These commands operate with variables defined in the major-mode's hook and automatically configure MicroEmacs to use the template.

The commands are used in the major-mode hook files (hkXXX.emf - see File Hooks) and simply the setting up of File Language Templates by automating common actions and integrating the configuration settings made via **buffer-setup**(3).

The hook file is defined by a call to **major-mode**(2) which is typically declared in `me.emf` or the user start up file, binding a file extension or a content identifier to the major-mode. i.e.

```
5 major-mode "XXX" ".XX1 .XX2"
13 major-mode "YYY" ".YY1 .YY2" "^#!/.*YYY"
```

The first time a file is loaded that is bound to **fhook-XXX** via a **major-mode**(2) definition, the `hkXXX.emf` macro file is executed and then the **fhook-XXX** command is executed with the file's buffer current so that the file hook is applied to it. Only the execution of **fhook-XXX** is done on subsequent loading of files of this type.

When the **fhook-XXX** command is executed it's #l9 register variable is set to the matched magic string if the binding was via a file content hook, or an empty string. This can be useful as the magic string often contains configuration values such as the tab and indent width.
A basic template framework to be used with the **buff-init** commands may be defined as follows for a language template called *XXX*:

```
; File hook - initialize buffer on loading the file.
define-macro fhook-XXX
    set-variable $buffer-mask "luh1"
    @# buffer-init "XXX"
    buffer-init-hooks
!emacro

; buffer-init variables
set-variable .fhook-XXX.name "XXX Macro"
set-variable .fhook-XXX.setup &reg "/history/fhook/XXX" "bdfghnopx"
set-variable .fhook-XXX.setup-mask "abdefghikmnoptux"
; Commenting configuration -- omit if not defined.
set-variable .fhook-XXX.comment "|#||#|# | #||"
; Additional commands -- omit if not defined.
set-variable .fhook-XXX.command-flag  "||"
set-variable .fhook-XXX.command-name  "||"
set-variable .fhook-XXX.command-nbind "||"
set-variable .fhook-XXX.command-kbind "||"
set-variable .fhook-XXX.command-desc  "||"
; OR define the commands one per line uniquely numbered from 0.
set-variable .fhook-XXX.command-0  "|bo||xxx-para|esc o||"
```

```
set-variable .fhook-XXX.command-1  "|bo||xxx-para|esc q||"
; Define additional buffer setup menu items -- omit if not defined
set-variable .fhook-c.setup-flags   "|c|l|"
set-variable .fhook-c.setup-labels  "|Option 1|Option 2|"
; Set up collapse or folding for XXX -- omit if not defined
set-variable .fhook-XXX.collapse-open  "^{"
set-variable .fhook-XXX.collapse-close "^}"
set-variable .fhook-XXX.collapse-mopen  "-1"
set-variable .fhook-XXX.collapse-mclose "1"
set-variable .fhook-XXX.collapse-mnext "-1"
; setup XXX item-list -- omit if not defined
set-variable .fhook-XXX.item-list-s1 "^[ \t]*func\\(\\w+\\)"
set-variable .fhook-XXX.item-list-r1 "^[ \t]*end"
; Set up hilighting
!if &and &sin "h" .fhook-XXX.setup &band .hilight.flags 0x02
    ; Hi-light XXX Mode
    !if &not &exi .hilight.XXX
        set-variable .hilight.XXX &pinc .hilight.next 1
    !endif

    0 hilight .hilight.XXX  2 50
    hilight .hilight.XXX 0 "#.*"          .scheme.comment
    hilight .hilight.XXX 1 "end"          .scheme.keyword
    hilight .hilight.XXX 1 "^[ \t]*func" .scheme.function
    ; ... more ...

    ; Hilight numbers.
    !if &band .hilight.flags 0x08
        ; hilight constants, e.g. numbers
        hilight .hilight.XXX 1 "[[:digit:]]+"       .scheme.constant
        hilight .hilight.XXX 1 "-[[:digit:]]+"      .scheme.constant
        hilight .hilight.XXX 1 "0[xX][[:xdigit:]]+" .scheme.constant
        hilight .hilight.XXX 1 "[[:digit:]]+\.[[:digit:]]+"  .scheme.constant
        hilight .hilight.XXX 1 "-[[:digit:]]+\.[[:digit:]]+" .scheme.constant
        hilight .hilight.XXX 1 "[[:digit:]]+\.[[:digit:]]+f"  .scheme.constant
        hilight .hilight.XXX 1 "-[[:digit:]]+\.[[:digit:]]+f" .scheme.constant
    !endif
!endif

; End the initialisation - call user extensions i.e. myXXX.emf
buffer-init-fhook "XXX"
```

**buffer-init** is invoked from the buffer file hook fhook and is invoked with the numerical argument passed to the *fhook* macro (specified as `$#`). The call takes two arguments, the first is the *name* of the template or hook, in the example this is *XXX*. The second argument is the *etf* template name (see **etf**(8)). The command initializes the buffer, using the *.fhook-XXX* variable information defined in the file.

**buffer-init-hooks** is invoked from the buffer file hook fhook typically following the **buffer-init** call this macro invokes any user defined file hooks of the form **my-fhook-XXX**. User defined **my-bhook-XXX** and **my-ehook-XXX** are configured into the buffer space ready for execution.

**buffer-init-hooks** must be placed in all **fhook-XXX** macros to correctly initialize the file hooks. If this is not added then the "`my-*hook-always`" macros may not be called correctly for buffers where the **fhook-XXX** has not been configured correctly.

**buffer-init-fhook** is placed at the end of the hook file definition and retrieves additional configuration information from the registry and the user settings, it will also locate and load any user additions to the template by loading the file `myXXX.emf`.

A set of global default file hooks may be defined of the form **my-fhook-always**, **my-bhook-always**, **my-ehook-always** and **my-dhook-always**. These file hooks are global and invoked after the

### Execution Sequence

It is important to understand the execution sequence of the buffer hook files. On identifying a new file hook then the `hkXXX.emf` file is fetched from the file system and executed. Execution causes all of the variables to be defined and finally, when all definitions have been made, **buffer-init-fhook** is executed to load in any user defined extensions. At this point then the macro **fhook-XXX** has ONLY been defined it has NOT been executed. This load operation is performed once only and is on demand.

The **fhook-XXX** macro must be defined BEFORE any of the variables associated with the hook as all subsequent variables are command variables and are defined in the context of the macro definition. See **Variables**(4) for more information on macro variables.

**fhook-XXX** is invoked to apply hilighting, buffer key bindings, indentation etc. to a new buffer which has just been created.

**ehook-XXX** is invoked when a buffer is swapped out. This may be used to save global variables that have been modified in the context of the buffer.

**bhook-XXX** is invoked when a buffer is swapped in. This may be used to restore variables that are specific to a buffer to global variables.

**dhook-XXX** is invoked before a buffer is deleted. This may be used to clean up a buffer before it is destroyed.

**Extended file hooks**

The buffer hooks `[fbed]hook-XXX` may be extended by user specific configuration in the form `my-[fbed]hook-any`, these are invoked after the primary hook as been invoked and operate on all buffers rather than a specific buffer type. The *my-*hook-always* mechanism is provided by the *buffer-init* macros of `buffinit.emf` rather than from the base command set of MicroEmacs.

### Configuration Masks

Buffer initialization is controlled by the settings of the variables **.fhook-XXX.setup** and **fhook-XXX.setup-mask**:

```
set-variable .fhook-XXX.setup &reg "/history/fhook/XXX" "bdfghnopx"
set-variable .fhook-XXX.setup-mask "abdefghikmnoptux"
```

The variables use an ASCII configuration string that identifies the modes of operation that are required. Upper and lower case alternatives are presented, an uppercase flag disables the mode, lower case enables. Where the flag is not specified then the mode is not changed.

`a` - Auto mode
: Configure **auto**(2m) mode for automatic source file line type detection. Typically enabled for text files.

`b` - Abbreviation mode
: Configures the abbreviation mode using **buffer-abbrev-file**(2) for the buffer.

`d` - Auto indentation operation.
: Automatic indent is enabled using **indent**(2).

  Where **indent**(2m) mode is required then omit the `d` flag and use flag `i` instead.

`e` - Exact or case sensitivity operation
: Configure **exact**(2m) case sensitive mode. `e` enable exact, `E` disable exact.

`f` - Folding/Collapsing operation
: Configures folding support, if enabled then the variables *.fhook-XXX.collapse-OPERATION*, should be defined to declare the folding points.

`g` - Fence matching operation
: Configure fence matching mode with **fence**(2m). Note that fence matching is now typically performed in the macro environment using **display-matching-fence**(3) rather than the in-built fence matching. This colorizes the fences that are matched. `g` enable fence matching, `G` disable fence matching.

`h` - Hilighting
: Support syntax hilighting, when defined then the hilighting scheme defined by *.hilight.XXX* is attached to the buffer.

`i` - Automatic indentation
: Support indentation using **indent**(2m). Automatic indentation (see **indent**(2)) is defined with the `d` flag. `i` enables indent, `I` disables indent.

`j` - Justify
: Configure paragraph justification using **justify**(2m). `j` enable justify, `J` disable justify.

`k` - Backup mode
: Configure backups with **backup**(2m) mode. `k` enable backup, `K` disable backup.

`m` - Magic mode
: Configure regular expression **magic**(2m) mode. `m` enable magic, `M` disable magic.

`n` - New buffer header
: Enable the insertion of a buffer header from the **etf**(8) template file.

`o` - Tool menu
: Enable the tool menu for the buffer.

`p` - Help page
: Produce a help page for the buffer describing the bindings.

`s` - Auto spell
: Configure automatic spelling for the buffer using **auto-spell**(3).

`t` - Tab mode
: Configure insertion of literal tabs by enabling **tab**(2m) mode. `t` enable tab, `T` disable tab.

`u` - Undo
: Configure the undo/redo history using **undo**(2m) mode. `u` enable undo, `U` disable undo.

`w` - Wrap
: Configure wrapping using **wrap**(2m) mode. `w` enable wrap, `W` disable wrap.

`x` - Time stamping
: Configure time stamping as defined by **time**(2m).

`1` - Reformatting hilighting
: A special internal flag indicating that hilighting is used for reformatting purposes, i.e. to make readable and hide control text.

`#` - Commenting
: Special flag used to indicate commenting is supported. The comment format should be defined with the *.fhook-XXX.comment* variable within the template.

**.fhook-XXX.setup** is the current set-up, this is normally defined to take settings from the registry providing a default when not defined. This is the only mask that is allowed to include the uppercase flags. All values defined in **.fhook-XXX.setup** are explicitly enabled or disabled by default. The setup set should contain the configuration flags that are required to make the template operate correctly, but should not include modes that the user may set up i.e. *exact* or *magic* etc.

**.fhook-XXX.setup-mask** defines the set of flags that are allowed with the template. The configuration defined in **buffer-setup**(3) is tested against the mask, if the mask allows it then the mode is enabled. **.fhook-XXX.setup** is generally a subset of **.fhook-XXX.setup-mask**.
**buffer-setup**(3) operates with the values defined by **.fhook-XXX.setup** and **.fhook-XXX.setup-mask**. The initialization of **.fhook-XXX.setup** is performed using the registry as follows:

```
set-variable .fhook-XXX.setup &reg "/history/fhook/XXX" "bdfghnopx"
```

This allows the user to configure over-ridding modes via **buffer-setup**(3). The user configuration is saved to the Registry and restored using **&reg**(4), when the variable is defined. The flags that are defined by **buffer-setup**(3) include `a`, `b`, `c`, `d`, `e`, `f`, `g`, `h`, `i`, `j`, `k`, `m`, `n`, `o`, `p`, `s`, `t`, `u`, `w`, and `x`.

### Tags

The buffer setup may be configured to add a tags generator to the language template, this is added to the menu. The tags generation is declared in the template by assigning the variable **.fhook-XXX.tags** with a command line to run tags utility. The command line string should commence with a `!` character to tell MicroEmacs to execute an external command, for example to use the GNU **ctags**(2) utility is defined as:

```
set-variable .fhook-c.tags "!ctags *.c *.h"
```

When the value starts with a '`!`' then rest of the line is spawned as a process, no additional interface is provided to the command.
Some language templates include a macro *tags* facility to generate a tags file (See **emftags**(3f) - `emftags.emf`, **ctags**(3f) - `ctags.emf`, **ntags**(3f) - `ntags.emf`). **.fhook-XXX.tags** may be defined to use an internal macro based tags generator rather than an external command. Without the '`!`' character the command line is assumed to be an MicroEmacs macro based and runs the command

```
me @<.tags-var-value> -v%tag-options=<options>
```

A macros based command line for the 'C' Programming Language is defined as:

```
set-variable .fhook-c.tags "ctags"
```

The presence of a tags generator is not flagged in the setup flags, declaration of the **.fhook-XXX.tags** variable is sufficient to declare that a tags generator exists.

### Private Configuration Extensions

The **.fhook-XXX.setup** may be extended to include local configuration items that are private to a specific buffer. The extensions are added to the **buffer-setup**(3) menu and provide an on/off check-box control to enable and disable specific buffer features. Feature examples include: Enable or disable Doxygen highlighting in a buffer, control hidden file names in a directory listing etc.

The configuration extensions are defined using two additional hook variables, these variables are normally omitted and only defined if the extensions are required.

**.fhook-XXX.setup-flags**
: Defines the alphabetic [a-z] setup characters that are used to enable or disable a feature. These are defined as MicroEmacs list variables (see **&lfind**(4), **&lget**(4), etc. for the full syntax definition). The characters that may be used are those that are not used by buffer setup i.e. "clqrvyz"

**.fhook-XXX.setup-labels**
: Defines the text labels that are added to the setup menu, the label is associated with the enable button.

To add a new feature switch then the the character flags to hold the state are are defined with the **.fhook-XXX.setup-flags** and **.fhook-XXX.setup-labels**. The characters defined with the **setup-flags** should be added to **.fhook-XXX.setup-mask**. If the option is to be enabled by default then the enabling character is added to **.fhook-XXX.setup**.

To enable/disable the features then **buffer-setup**(3) when invoked modifies the state of the variable **.fhook-XXX.setup** and then executes the buffer hook again causing the new state to be processed and applied. The modified **.fhook-XXX.setup** value is stored in the user registry and is applied thereafter when the specified buffer type is loaded.
Within the template definition then the additional flags are checked by testing the state of the **.fhook-XXX.setup** variable. An example is shown below.

```
; Add the extra variable to the buffer-init variables
; Option 1 is disabled and Option 2 is enabled by default here.
set-variable .fhook-XXX.setup &reg "/history/fhook/XXX" "bdfghnopxl"
set-variable .fhook-XXX.setup-mask "abdefghikmnoptuxcl"

; Define the additional options.
set-variable .fhook-XXX.setup-flags   "|c|l|"
set-variable .fhook-XXX.setup-labels  "|Option 1|Option 2|"

; Test for the Option 1 being enabled.
!if &sin "c" .fhook-XXX.setup
    ; Do Option 1 configuration.
!endif

; Test for the Option 2 being enabled.
!if &sin "l" .fhook-XXX.setup
    ; Do Option 2 configuration.
!endif
```

The templates hkc.emf, hkcpp.emf and hkjava.emf include private extensions which may be used as examples.

### Comments

The format of comments may be defined in the template using the **.fhook-XXX.comment** variable. Definition of the comment form allows the comment formatting to be automatically set up. Where the comment format is defined then the following key bindings are automatically added to the template:

C-c C-s **comment-start** - start a new comment.
C-c C-c **comment-line** - comment out the current line.
C-c C-d **uncomment-line** - remove commenting from current line.
C-c C-e **comment-to-end-of-line** - comment to end of line.
C-c C-r **comment-restyle** - format/re-style comment.

Help information is automatically added to the buffer to describe the key bindings if help information is enabled.

The default comment format is defined with the variable **.fhook-XXX.comment** with an argument string in the form "— <*comment-start*> — <*comment-end*> — <*comment-pad*> — <*mid-box-left*> — <*mid-box-right*> — <*comment-flags*> —". Where a language supports multiple comment types, or styles, then additional comment styles may be specified with a comment definition of the form **.fhook-XXX.comment-N** where **N** is an integer from `1` to `n`. Each definition defines an different comment style. Additional comment definitions should be ordered such that the longest or most definitive comment definition appears before other comment definitions which have less definitive features. The comment matching searches each of the comment definitions in the order `1` to `n` to determine the comment style, as soon as a positive match is located then the comment is used without searching the remaining comments.

The fields of the comment definition are defined as follows:

<*comment-start*>
: The format of the start of the comment or comment open syntax.

<*comment-end*>
: The format of the closing comment, where the comment exists to the end of the line then the field is defined as empty. Leading white space should be included where this affects the horizontal positioning of the comment, white leading space is ignored in any match i.e. " `*/`" for C type comments.

<*comment-pad*>
: The character to use to pad a comment between the <*comment-start*> and <*comment-end*> characters when creating a line or a box formation.

<*mid-box-left*>
: The start of a comment continuation line when creating a box comment.

<*mid-box-right*>
: The end of a comment continuation line when creating a box comment and the comment is not closed and continues on the next line.

<*comment-flags*>
: The comment flags define how the comment should be formatted when re-styled. The flags are defined as single characters that describe the attributes. Re-styling may include the following attributes:

  **b** Box format required, create a left and right edge for the comment.

  **f** A footer line is required, a new line is created following the comment text to contain the with a closing comment <*comment-end*> string. The end of comment is not padded.

  **F** As **f** except the footer line is left padded with <*comment-pad*> strings.

  **h** A header line is required, a new line is created before the comment text to contain the opening comment <*comment-start*> string. The end of comment is not padded.

  **H** As **h** except the header line is right padded with <*comment-pad*> strings.

  **j** Enable justification mode in the *comment* buffer and re-style.

  **r** A restyle key binding is added to the buffer "`esc o`" and "`esc q`" buffer key bindings are added and bound to **comment-restyle**(3). The **r** flag is only processed from the

  Examples of the comments format are shown below.

**'C' Programming language**

Format: `/* ... */`

```
set-variable .fhook-c.comment "|/*| */|*| * | * |f|"
```

The comment output is defined as:

```
/*************************************************************
 * This is a box comment                                     *
 *************************************************************/

/* Comment to end of line ************************************/
```

**C++ Programming language**

Format: `// ... <EOL>`

```
set-variable .fhook-cpp.comment "|//||/|// | //||"
```

The comment output is defined as:

```
////////////////////////////////////////////////////////////
// This is a box comment                                  //
////////////////////////////////////////////////////////////

// Comment to end of line /////////////////////////////////
```

C++ supports both C and C++ comment styles, both styles may be defined in the language template. For C++ then `//` is assumed to be the default style.

```
set-variable .fhook-cpp.comment   "|//||/|// | //||"
set-variable .fhook-cpp.comment-1 "|/*| */|*| * | * |f|"
```

**MicroEmacs macro language**

Format: `; ... <EOL>`

```
set-variable .fhook-emf.comment "|;||;|; | ;||"
```

The comment output is defined as:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
; Box Comment                                                      ;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

; Comment to end of line ;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
```

**Shell script**

Format: `# ... <EOL>`

```
set-variable .fhook-shell.comment "|#||#|# | #||"
```

The comment output is defined as:

```
################################################################
# Box Comment                                                  #
################################################################

# Comment to end of line ######################################
```

**MS-DOS Batch file**

Format: `rem ...  <EOL>`

```
set-variable .fhook-dos.comment "|rem |||rem |||"
```

The comment output is defined as:

```
rem
rem Box Comment
rem

rem Comment to end of line
```

**Commands and Key Bindings**

Local buffer commands and key bindings for over-riding key definitions used within the buffer are defined with the `fhook-XXX.command-*` variables. These are defined as MicroEmacs list variables (see **&lfind**(4), **&lget**(4), etc. for the full syntax definition) and define a list of keys, commands and bindings. Each command is typically separated by the '`|`' character. The field order in all of the `fhook-XXX.command-*` variables MUST be consistent, so if a command is defined in field 5, then all definitions associated with that command MUST be defined in field 5 for all variables.

To make the command definitions easier then the syntax `fhook-XXX.command-n` may be used where *n* is an integer from *0..n-1*. The `fhook-XXX.command-n` statements are used to construct the `fhook-XXX.command-*` lists, the advantage being that the command and key bindings etc. are defined together. Once the `command-n` variables are processed then they are deleted from the environment. The general syntax of the `command-n` variable is defined as a list:

```
|command-flag|command-nbind|command-name|command-kbind|command-desc|
```

The command variables are defined as follows:

*.fhook-c.command-flag*

A configuration flag that defines the components to use for the command. The flags are set as individual characters and are defined as follows:

`t` - Toolbar

Show the command in the toolbar. The `.fhook-XXX.command-desc` variable should contain a readable name for the command to display in the menu. An optional hot key for selection in any osd is specified by prefixing the character in the string with `\H`.

`h` - Help Screen

Show the command in the help screen. The `.fhook-XXX.command-desc` variable should contain a readable name for the command to display in the menu. An optional hot key for selection in any osd is specified by prefixing the character in the string with `\H`.

`b` - Key binding

Create a buffer key binding for the command, the key to bind is defined by the variable `fhook-XXX.command-kbind`, if the command requires a numerical argument then this is defined in the variable `fhook-XXX.command-nbind`. The binding is only created if the key buffer binding does not already exist, the binding may be forced by using the `o` flag. Where a key binding is only to be created if hilighting is enabled then the `b` flag should be qualified with the `H` flag.

`o` - Forced key binding
: Optionally used with the `b` and forces the binding to over-write any existing binding.

`i` - Emulated key binding
: Used with the `b` flag, ignores key bindings that are emulation mappings.

`H` - Hilighting key binding
: Used with the `b` flag, only performs the key binding if hilighting is enabled.

*.fhook-XXX.command-name*
: Specifies the name of the command to execute, this may be a built in command of a macro. Typically the command is a local macro defined in the hook file.

*.fhook-XXX.command-nbind*
: The value of the numeric argument to pass to the command defined in the *.fhook-XXX.command-name* field. This field may be empty if no numeric argument is required.

*.fhook-XXX.command-kbind*
: The local key binding to bind to the command, this field is only interpreted if the `b` is specified, otherwise the field may be empty. Where multiple key bindings are required to the same command then a field definition is required for each binding.

*.fhook-c.command-desc*
: A short description of the command that is used for the help page (`h`) and toolbar (`t`). The field may be empty if the field is not specified for insertion into the toolbar or help page.

As an example, two new commands or bindings may be defined for a template, the first field is bound to key `A-q` which executes macro `XXX-comp` with a numerical argument of 2. The second field binds a new key to **comment-restyle**(3).

```
.....

set-variable .fhook-XXX.command-flag  "|th|bo|"
set-variable .fhook-XXX.command-name  "|XXX-comp|comment-restyle|"
set-variable .fhook-XXX.command-nbind "|2||"
set-variable .fhook-XXX.command-kbind "|A-q|esc o|"
set-variable .fhook-XXX.command-desc  "|Complete command||"

.....

; Local macro definition to perform command completion
define-macro XXX-comp
    set-variable #l1 @#
    .....
!emacro
```

The alternative form of definition using the `command-n` syntax is defined as:

```
.....

set-variable .fhook-XXX.command-0 "|th|2|XXX-comp|A-q|Complete command|"
set-variable .fhook-XXX.command-1 "|bo||comment-restyle|esc o||"

.....

; Local macro definition to perform command completion
define-macro XXX-comp
    set-variable #l1 @#
    .....
!emacro
```

The key bindings sometimes need to be protected where they have been redefined to perform some other operation for example when emulation modes are used (i.e. **nedit**, **me3.8**). The buffer initialisation uses the command **buffer-bind-create** and the variables **.buffer-bind-create.map-from** and **.buffer-bind-create.map-to** which define replacement keys for bindings. As an example the **nedit** emulation mode uses key binding `C-c C-c` and provides a buffer binding mapping to translate any key buffer binding to a different key. i.e.

```
; reconfigure the buffer bindings
set-variable .buffer-bind-key.map-from "|C-c C-c|C-c C-d|"
set-variable .buffer-bind-key.map-to   "|C-b|C-u|"
```

### Information Hiding with Collapse

MicroEmacs provides a generic, albeit coarse, folding mechanism which is applied to some of the well known file modes. The folding mechanism allows parts of the buffer to be scrolled up and hidden, leaving a residue hilighting marker within the buffer indicating a folded region. A folded buffer typically allows a summary of the buffer contents to be viewed within several windows, hiding the detail of the buffer.

Note: **collapse** was called **fold** in earlier releases but the name was changed when real fold editing support was added, see `fold.emf` for more information.

The folding mechanism uses well defined *start* and *end* markers which form part of the syntax of the well known file mode. i.e. in 'C' this is the open and closed braces that appear on the left-hand margin ({ .. }). The intention is that the natural syntax of the text is used to determine the fold positions, requiring no additional text formating or special text tags to be inserted by the user.

The language template defines the syntax of blocks of information that may be collapsed and hidden with narrow. In order to utilize the **collapse-current**(3) and **collapse-all**(3) commands within a buffer, the *start* and *end* markers have to be initialized for the syntactical contents of the buffer. The buffer specific variables are defined as follows, where *XXX* is the file hook base name. The `.fhook-XXX.collapse-*` variables define the regular expression search pattern and line positioning.

**.fhook-*XXX*-collapse-open**
: A regular expression search string used to locate the start of the string. For speed the search string should include a regular expression start or end of line character whenever possible. i.e. in C the open is defined as "^{".

**.fhook-*XXX*-collapse-close**
: A regular expression search string used to locate the end of the string. For speed the search string should include a regular expression start or end line character whenever possible. i.e. in C the close is defined as "^}".

**.fhook-*XXX*-collapse-mopen**
: An integer value that denotes the number of lines to move in a forward or (negative n) backward direction from the *start* line located by the search string to the position in the buffer to be folded. If default value when **mopen** is omitted is 0, starting the fold from the search string line.

**.fhook-*XXX*-collapse-mclose**
: The relative displacement from the close fold line to the fold position, this is a positive or negative displacement depending on where the fold is to be positioned.

**.fhook-*XXX*-collapse-mnext**
: Specifies the number of lines to advance before the next search is continued on the fold operation. This is only used by collapse-all.

The following examples show how the collapse variables are set up in each of the buffer modes.

**C and C++**

**C** and **C++** collapse on the open and close brace appearing in the left-hand margin. The collapse variables are defined in `hkc/hkcpp.emf` as follows:

```
set-variable .fhook-c.collapse-open  "^{"
set-variable .fhook-c.collapse-close "^}"
set-variable .fhook-c.collapse-mopen  "-1"
set-variable .fhook-c.collapse-mclose "1"
set-variable .fhook-c.collapse-mnext "-1"
```

Given a 'C' function definition:

```
static void
myfunc (int a, int b)
{
    /* Function body */
}
```

the folded version appears as follows:

```
static void
myfunc (int a, int b) ... }
```

**emf**

MicroEmacs macro files **emf** support folding of macro definitions, the collapse variables are defined in `hkemf.emf` as follows:

```
set-variable .fhook-emf.collapse-open  "^0? ?define-macro[ \t]"
set-variable .fhook-emf.collapse-close "^!emacro"
set-variable .fhook-emf.collapse-mclose "1"
set-variable .fhook-emf.collapse-mnext "-1"
```

Given a macro definition:

```
0 define-macro mymacro
; This is the body of the macro
; ... and some more ...
!emacro
```

the collapsed version of the macro is defined as:

```
0 define-macro mymacro ... !emacro
```

**nroff**

**nroff** is configured for manual pages only and folds between `.SH` and `.SS` sections, the hook variables are defined as follows:

```
set-variable .fhook-nroff.collapse-open  "^\.S[SH]"
set-variable .fhook-nroff.collapse-close "^\.S[SH]\|\'"
set-variable .fhook-nroff.collapse-mnext "-1"
```

Given an nroff block of text defined as:

```
> .SH SYNOPSIS
> Some text
> Some more text
> .SH DESCRIPTION
```

Then the collapsed version appears as:

```
> .SH SYNOPSIS
> .SH DESCRIPTION
```

**tcl/tk**

**tcl/tk** is configured to collapse procedures. The collapse variables are defined as:

```
set-variable .fhook-tcl.collapse-open  "^proc "
set-variable .fhook-tcl.collapse-close "^}"
set-variable .fhook-tcl.collapse-mclose "1"
set-variable .fhook-tcl.collapse-mnext "-1"
```

Given a tcl procedure definition:

```
proc tixControl:InitWidgetRec {w} {
    upvar #0 $w data

    tixChainMethod $w InitWidgetRec

    set data(varInited)   0
    set data(serial)    0
}
```

The collapsed version of the same section appears as:

```
proc tixControl:InitWidgetRec {w} { ... }
```

Where folding is required using **fold-open**(3) then the collapse methods above are not enacted when folding is enabled. Folding may be enabled as defined by the File Hooks, alternatively the folding may be explicitly defined using the variable *.fhook-XXX.fold*. The *.fhook-XXX.fold* variable argument is defined as "— *<literal-open>* — *<literal-close>* — *<comment-termination>* — *<regex-find-open>* — *<regex-find-close>* —. Where the fields are defined as follows:

*<literal-open>*
: The literal fold open string.

*<literal-close>*
: The literal fold close string.

*<comment-termination>*
: The comment termination string.

*<regex-find-open>*
: The regular expression to find an open fold.

*<regex-find-close>*
: The regular expression to find a close string.

As an example, a language may have the following syntax for defining folds:

```
/* {{{ This is an example fold. */
This is
the body
of the fold
/* }}} */
```

The fold is defined as follows:

```
set-variable fhook-XXX.fold "|/\\* {{{|/\\* }}}|\\*/|/\\*[ \t]*{{{|/\\*[ \t]*}}}|"
```

### Item Lists

Item Lists perform a regular expression search of a buffer, presenting a list of the located text and associated types in a separate window which is presented to the left of the buffer window or more typically in the toolbar. The command **item-list**(3) is a generic command that interacts with the buffer environment variables to present abbreviated buffer information to the user.

The regular expression search strings are predefined in the language templates, using the variables **.fhook-***XXX***-item-list-s***Y* and **.fhook-***XXX***-item-list-r***Y* and are described more fully in **item-list**(3).

### Hilighting

Buffer hilighting is defined by the **hilight**(2) command. Within the context of **buffer-init** then the standard definitions and name space conventions should be adhered to:

```
!if &not &exist .hilight.XXX
    set-variable .hilight.XXX &pinc .hilight.next 1
!endif
!if &and &sin "h" .fhook-XXX.setup &band .hilight.flags 0x02
    ; High-light XXX Mode
    0 hilight    .hilight.XXX 0                    $global-scheme
    .....
    ; Variable hilighting
    !if &band .hilight.flags 0x08
        ; hilight constants, e.g. numbers
        hilight .hilight.XXX 1 "[[:digit:]]+"       .scheme.constant
        ......
    !endif
!endif
```

### Indentation

Buffer indentation is defined by the **indent**(2) command. Within the context of **buffer-init** then the standard definitions and name space conventions should be adhered to. The buffer indentation scheme uses the hilighting variable, if hilighting is not defined then the variable allocation of `.hilight.XXX` should be performed as defined in the previous Hilighting section.

```
!if &sin "d" .fhook-XXX.setup
    0 indent  .hilight.XXX 0 10
    indent .hilight.XXX n "{"  4
    indent .hilight.XXX o "}" -4
    .....
!endif
```

### HISTORY

The January 2005 version of MicroEmacs introduced the following changes:

- The `my-*hook-always` introduced allowing global file hooks to manipulate operate on any buffer.
- The Electric-C indentation previously flagged as > was removed, as was **cmode**(2m). Indentation configuration for C mode is now performed by the **indent**(2) command.
- Private template configuration using **.fhook-XXX.setup-flags** and **.fhook-XXX.setup-labels**.
- The `fhook-XXX.command-n` syntax was introduced in June 2005 to simplify the command list definitions.

## SEE ALSO

**$buffer-hilight**(5), **$buffer-indent**(5), **major-mode**(2), **buffer-abbrev-file**(2), **buffer-setup**(3), **collapse-all**(3), **collapse-current**(3), **ctags**(3f), **eaf**(8), **emftags**(3f), **etf**(8), **insert-template**(3), **execute-buffer**(2), **expand-abbrev**(2), **global-abbrev-file**(2), **hilight**(2), **indent**(2), **indent**(2m), **item-list**(3), **narrow-buffer**(2), **ntags**(3f), **restyle-buffer**(3), **restyle-region**(3), **scheme-editor**(3), **time**(2m),
File Hooks,
File Language Templates.

Ref: buffer-init(3) File: m3mac064.3 Date: 2026/04/16

## NAME

buffer-major-mode - Change the operating mode of a buffer

## SYNOPSIS

**buffer-major-mode** "*modename*"

## DESCRIPTION

**buffer-major-mode** changes the major mode of a buffer affecting the highlighting, indentation, key bindings and mode specific commands to the specified *modename*. This allows the buffer to be configured correctly for the type of file.

On invocation then the user is prompted for the *modename*, TAB performs completion.

Running **buffer-major-mode** causes the File Hook to be executed for the buffer for the new *modename*.

## NOTES

**buffer-setup**(3) may be used to make local changes to the major mode for the current buffer only. **major-mode-setup**(3) may be used to change the major mode settings across all buffers.

## SEE ALSO

**buffer-setup**(3), **major-mode-setup**(3), File Hooks

Ref: buffer-major-mode(3) File: m3mac087.3 Date: 2022/06/12

## NAME

c-hash-eval - Evaluate C/C++ #defines
c-hash-undo - Remove C/C++ #define evaluation
c-hash-define - Set a C/C++ #define
c-hash-undef - Unset a C/C++ #define

## SYNOPSIS

*n* **c-hash-eval**
**c-hash-undo**
**c-hash-define** "*variable*" "*value*"
**c-hash-undef** "*variable*"

## DESCRIPTION

**c-hash-eval** evaluates C/C++ '#' lines, hiding sections of code which have been 'hashed' out. **c-hash-eval** evaluates the following '#' lines:

#define <variable> ....
#ifdef <variable>
#if ...
#else
#endif

For `#defines` **c-hash-eval** creates a user variable "`%chd-<variable>`", setting it to the value found. For `#ifdef` a simple check for the existence of variable "`%chd-<variable>`" is made. If defined then code between the `#ifdef` and either its matching `#else` or `#endif` is displayed and code between the `#else` and `#endif` is hidden. If it is not defined then the reverse happens.

The state of `#if`'s are evaluated using **calc**(3), the following code is then displayed as for `#ifdef`.

Code is hidden by setting the **$line-scheme**(5) to a color similar to the back-ground. If an argument is given to the command the code is also narrowed out using **narrow-buffer**(2).

**c-hash-undo** undoes the effect of **c-hash-eval** by restores hidden code.

**c-hash-define** and **c-hash-undef** can be used to manually set and unset `#define` variables.

## NOTES

**c-hash-eval**, **c-hash-undo**, **c-hash-define** and **c-hash-undef** are macros defined in `cmacros.emf`.

Executing **c-hash-eval** in a project header file (h file) which contains all used `#define` definitions will set up all `#define` variables ready for the main C files.

## SEE ALSO

**calc**(3), **$line-scheme**(5), **narrow-buffer**(2).

Ref: c-hash-eval(3) File: m3mac017.3 Date: 2022/06/12

## NAME

c-slashify - C language, append backslash to end of lines of region
c-deslashify - C language, remove backslash to end of lines of region

## SYNOPSIS

**c-slashify**
**c-deslashify**

## DESCRIPTION

**c-slashify** and **c-deslashify** add and remove backslash characters to the ends of line(s) defined by a region, respectively.

The commands are used with the C programming language for `#define` macro editing, allowing backslashes to be added and removed quickly. Backslashes are typically removed with **c-deslashify** and the macro edited before the backslashes are re-applied using **c-slashify**.

## SEE ALSO

**c**(9), **cpp**(9).

Ref: c-slashify(3) File: m3mac069.3 Date: 2022/06/12

## NAME

c-to-cpp-convert - Convert a C file to C++

## SYNOPSIS

**c-to-cpp-convert**

## DESCRIPTION

**c-to-cpp-convert** changes the buffer type from C to C++ and loads the C++ language template **cpp**(9). The command is only available once a 'C' file is loaded as the command is defined within the C language template file `hkc.emf`.

The command is normally used when a new C++ file is being created, if the default file type is C rather than C++ then the new file is converted to C++ by changing the magic string from `-*- c -*-` to `-*- c++ -*-`, whenever the file is subsequently loaded then it will be presented in the correct mode.

## SEE ALSO

**c**(9), **cpp**(9).

Ref: c-to-cpp-convert(3) File: m3mac068.3 Date: 2022/06/12

## NAME

calc - Calculator
insert-calc - Calculator with answer insertion
calc-op - Low level calculator command

## SYNOPSIS

**calc** "*string*"
**insert-calc** "*string*"
**calc-op** "*string*"

## DESCRIPTION

**calc** can perform floating point, simple integer and logic based calculations given by "*string*", where the "*string*" takes the following form:

```
"[b]<s>"
```

Where '*b*' is an optional letter setting the required output base/format which can be one of the following:

`f` - Floating point
`b` - Binary
`o` - Octal
`d` - Decimal
`x` - Hexadecimal

Default when omitted is a hybrid of '*f*' (floating point) and '*d*' (decimal) whereby the calculation is performed using both floating point and integer maths and if the results differ then both are returned (floating point first with integer in brackets after). "*s*" is the sum to be calculated, which typically takes the form of a number followed by 1 or more operator & number pairs.

**Calc** supports numbers in the following formats:

`0b##` - Binary number where # can be 0-1
`0##` - Octal number where # can be 0-7
`##` - Decimal number where # can be 0-9, note it should not be prefixed with a 0
`0x##` - Hexadecimal number where # can be 0-9, A-F or a-f
`#.#` - Floating point number where # can be 0-9
`#e#` - Floating point number with exponent where # can be 0-9
`#.#e#` - Floating point number with exponent where # can be 0-9
`Ans` - Last calculation answer (case insensitive)
`$xxx` - MicroEmacs system variable
`%xxx` - MicroEmacs global variable
`.xxx` - MicroEmacs command variable
`"xx"` - String value (for logical evaluations)

Following is a list of supported operators in order of precedence (which is in compliance with the C programming language):

```
(..)   - Parentheses (contents calculated first)
!      - Bitwise-not, Logical-not (number prefix)
+ -    - Positive/Negative number (number prefix)
* / %  - Multiply, Divide, Modulus
+ -    - Addition, Subtraction
<< >>  - Bitwise rotate left/right
< > <= >= - Logical Less-than, Greater-than, Less-or-equal, Greater-or-equal
== != - Logical Equals, Not-equals
&      - Bitwise and
^      - Bitwise xor
|      - Bitwise or
&&     - Logical and
||     - Logical or
```

The answer of the last successful calculation is stored in **.calc.answer**(5), "Ans" in the following sum is replaced with this value. The result of the each calculation is stored in **.calc.result**(5), this can be the same as **.calc.answer**, the hybrid values when floating point and integer maths produce different answers and the required format is not specified, or, if there is an error in the string a MicroEmacs list with the first element being the character offset and the second being the error message.

The argument *n* is a bitwise flag where:

**0x01** Print out the result on the message-line.

**0x02** Insert the answer into the current buffer at the current location.

When omitted the default argument is 1.

**insert-calc** is a simple macro for executing **calc** with flag 2 set of the numerical argument so the answer is inserted.

**calc-op** is the low-level macro used by **calc** to perform the calculation, it is similar to executing calc with a numeric argument of 0 except that even errors are not printed to the message-line. This macro can be used by other macros to silently perform any computation, retrieve the result from variable **.calc.answer**(5).

## EXAMPLE

To calculate the number of hours in a year:

```
calc "365*24"
```

To then calculate the number of seconds in the year:

```
calc "Ans*60*60"
```

## NOTES

**calc** is a macro defined in `calc.emf`.

The command throws an error if an operator is given an incompatible argument, for example a floating point number cannot be directly given to a bitwise or boolean logic operator such as >> or ——, it can be given to a comparison operator, such as >=, the result of which is an integer boolean value (1 or 0) which can then be used by logical operators.

**calc-op** is used by **c-hash-eval**(3) to evaluate the state of any `#if` and `#ifdef` lines.

## SEE ALSO

**.calc.answer**(5), **.calc.result**(5).

Ref: calc(3) File: m3mac009.3 Date: 2026/02/17

## NAME

change-buffer-charset - Convert buffer; between two character sets
buffer-is-utf8 - Detect if the current buffer is UTF-8 encoded

## SYNOPSIS

**change-buffer-charset**
*n* **change-buffer-charset** "*from-charset*" "*fto-charset*"
*n* **buffer-is-utf8** *variable*

## DESCRIPTION

**change-buffer-charset** opens a dialog allowing the user to select a **From** and **To** character set. If the *Convert* button is selected the current buffer is converted to the destination character set. The command assumes that the current buffer is written in the **From** character set, no attempt is made to verify this.

The command also provides the ability to convert to and from UTF-8. MicroEmacs cannot directly support Unicode, UTF-8 or any multi-byte code pages, for various reasons (insert FAQ ref here) MicroEmacs only supports a single byte code page restricting the user to working with 255 characters at any one time. However, **change-buffer-charset** does provide the user with the ability to load a UTF-8 file, losslessly convert the loaded file to the current code-page, edit the file and then convert it back to UTF-8 before saving.

The lossless conversion makes use of 2 special prefix characters, **0x01** and **0x02**, followed by 3 or 5 hexadecimal digits respectively. It is important to understand that these 4 or 6 character strings are just that to the rest of the system, as a result operations like **kill-rectangle**(2) could split the character strings and invalidate them in any later conversion process. This is not an ideal solution but better than losing the characters entirely. The prefix characters are rendered as `[u]` and `[U]` if extended character rendering is enabled, see bit **0x10000** of **$system**(5) for more information.

**change-buffer-charset** also allows unsupported characters to be changed to the unsupported character (**0x07**) or the characters can be removed.

The current character set is configured using the **user-setup**(3) dialog (see Display Font Set). This in turn uses the command **set-char-mask**(2) to create the low level character conversion tables.

The **change-buffer-charset** dialog is not opened if both *from-charset* and *fto-charset* are given on the command-line, the argument *n* is used to set the options of the conversion, where the bits are defined as follows:

**0x01** When set the command prints the results of the conversion, when clear the command runs silently.

**0x02** When set, unsupported characters that cannot be preserved, or bit **0x04** is set, are simply removed. When this bit is clear they are changed to the unsupported character (**0x07**).

**0x04** When set, unsupported characters are not preserved, when clear they are preserved, see lossless conversion above.

**0x08** When set, unsupported characters are, where possible, converted to a character that

looks similar or may instil the same meaning. This is used in preference to converting to the unsupported char or removing, but is not an accurate translation.

A value of "`display`" can be given for either of the *from-charset* or *fto-charset* to indicate the user's current display character set, see **user-setup**(3).

The **buffer-is-utf8** command attempts to determine if the current buffer is UTF-8 encoded or not, setting the given *variable* to either:

**0** If the buffer contains no character whose ASCII number is greater or equal to `0x80`, i.e. there are no characters that would need encoding.

**1** If the buffer contains characters that would need encoding and all are encoded correctly for UTF-8. Note that this is not foolproof, the characters all being correctly encoded could be a fluke.

**-1** If the buffer contains characters that would need encoding and at least one is not encoded correctly for UTF-8.

## NOTES

**change-buffer-charset** and **buffer-is-utf8** are macros defined in `charsutl.emf`.

## SEE ALSO

**user-setup**(3), **set-char-mask**(2), **&uni**(4), Locale Support.

Ref: change-buffer-charset(3) File: m3mac041.3 Date: 2025/09/07

## NAME

change-company - Change current company profile”
%company-name - Name of company for template
%company-url - Company URL for template
%company-license - Company copyright license for template

## SYNOPSIS

*n* **change-company** ”*companyName*”
**%company-name** ”*string*” ; Default is ””
**%company-url** ”*string*” ; Default is ””
**%company-license** ”*string*” ; Default is ”*All rights reserved.*”

## DESCRIPTION

**change-company** changes the current company profile *companyName* by executing the macro file `companyName.emf`, this file is typically referred to as <*company*>.emf. If the file does not exist it is created with a base company template.

The <*company*>.emf macro file is used to define company related things such as company template information and supporting macros. The file will typically define values for the following variables:

**%company-name**
: Set to the name of the company, this value is used by **insert-template**(3) when replacing tag `$COMPANY_NAME$`.

**%company-url**
: Set to the main company URL, this value is used by **insert-template**(3) when replacing tag `$COMPANY_URL$` and to define the default value of `$COMPANY_REF$`.

**%company-license**
: Set to the text string of the company copyright notice, this value is used by **insert-template**(3) when replacing tag `$COMPANY_LICENSE$`. The value can contain any other tag, such as `$COMPANY_NAME$`, as well as newline ”`\n`” characters.

  When *companyName* is not supplied on the execution command-line, **change-company** prompts for the company name, using the value of **user-setup**(3)’s **Company File**, located on the Start-Up page, which can contain a space separated list of companies. The command also adds a ”*No company*” option which, if selected, unsets the **%company-name** variable and sets **%company-url** to the value of **%no-company-url** and **%company-license** to **%no-company-license**.

## NOTES

**change-company** is a macro defined in `tools.emf`.

## SEE ALSO

Company Profiles, **insert-template**(3), **user-setup**(3).

Ref: change-company(3) File: m3mac100.3 Date: 2024/12/16

## NAME

change-font-size - Incrementally change font size

## SYNOPSIS

*n* **change-font-size**

## DESCRIPTION

**change-font-size** incrementally increases or decreases the size of the font or resets it back to the original size. The numeric argument **n** is used to determine the action required, a **+ve** argument will increase the font size by the given amount, **-ve** will decrease and **0** or no argument will reset the size.

The size of **n** is important, a value of **+2** will increase the font size at twice the rate of **+1**.
When using a relatively small text size it is often desirable to use good bitmap based font rather than a scalable one as these typically offer greater clarity. For example ProggyFonts (*http://www.proggyfonts.net/*) are small yet very clear. However these fonts do not typically scale well. To address this issue the variable **.change-font.sz-font** can be set to a font name that should be used when changing the font size. For example, on UNIX this could be set to:

```
set-variable .change-font.sz-font "-*-fixed-medium-r-*-*-%d-*-*-*-*-*-iso8859-*"
```

On Windows the following could be used:

```
set-variable .change-font.sz-font "Consolas"
```

When this variable is used the first call to **change-font-size** often leads to a greater size change than desired as the new font sizing may differ, however this can then be adjusted via subsequent call to the command. When the font size is reset, by giving no numeric argument or an argument of **0**, the font as well as the font size is reset back to their original values.

## NOTES

The implementation of **change-font-size** is platform specific, on Windows the macro is defined in `windows.emf` and for UNIX systems it is defined in `unixterm.emf`. Both variant rely on the **change-font**(2) command.

## SEE ALSO

**change-font**(2).

Ref: change-font-size(3) File: m3mac097.3 Date: 2024/10/06

## NAME

change-scheme - Change the current highlight scheme

## SYNOPSIS

**change-scheme** *"schemeName"*

## DESCRIPTION

**change-scheme** changes the current highlight scheme used by the editor to the new scheme *schemeName* which is the ASCII name of the scheme. The change is immediate and the change in scheme is retained in the session. Note that the **tab** key provides a completion list from the command line prompt.

## NOTES

**change-scheme** is used by **read-session**(3) to restore the session scheme.

## SEE ALSO

**add-color-scheme**(2), **print-scheme**(2), **read-session**(3), **user-setup**(3).

Ref: change-scheme(3) File: m3mac084.3 Date: 2022/06/12

## NAME

check-line-length - Check the length of text lines are valid

## SYNOPSIS

**check-line-length**

## DESCRIPTION

**check-line-length** checks that the length of each line of the current buffer, starting with the current line, is less than or equal to **$buffer-fill-col**(5). The command aborts if a line too long is found, leaving the cursor on the offending line. If no invalid lines are found the command succeeds leaving the cursor at the end of the buffer.

## NOTES

**check-line-length** is a macro implemented in `misc.emf`.

## SEE ALSO

**$buffer-fill-col**(5).

Ref: check-line-length(3) File: m3mac052.3 Date: 2022/06/12

## NAME

clean - Remove redundant white spaces from the current buffer

## SYNOPSIS

*n* **clean**

## DESCRIPTION

**clean** removes redundant white spaces from the current buffer, there are three types this command remove:

1) Any space or tab character at the end of the line. All are removed until the last character is not a space or a tab, or the line is empty. Note that an empty line is not removed unless at the end of the buffer.

2) Space characters are removed when the next character is a tab, making the space redundant, e.g. the strings "`\tHello World`" and "` \tHello World`" will look identical because the tab character ('\t') indents the text to the 8th column with or without the space so the space can be removed.

3) Superfluous empty lines at the end of the buffer are removed, leaving only one empty line.

4) If argument *n* is given (value is not used) multiple blank lines are reduced to a single blank line.

## DIAGNOSTICS

`[Command illegal in view mode]`
: Caused by a redundant white space being found and the buffer being in view mode. Note that if clean completes while the buffer is in view mode then no superfluous white spaces where found.

## NOTES

**clean** is a macro defined in `format.emf`.

Most of this command's operation is performed by simple regex search and replace strings:

a) Search for: "[\t ]+$" Replace with: "\\0"

b) Search for: "[ ]+\t" Replace with: "\t"

c) Search for: "\n\n\n" Replace with: "\n\n"

It is useful to define a new command called **spotless** in the user macro file which cleans the file and removes blank lines. The new command is defined as follows:

```
; spotless; Perform a clean and remove any multi-blank lines.
define-macro spotless
    -1 clean
!emacro
```

## SEE ALSO

**replace-string**(2), **tab**(2m), **delete-blank-lines**(2), **tabs-to-spaces**(3).

Ref: clean(3) File: m3mac005.3 Date: 2026/02/17

## NAME

ClearCase - MicroEmacs ClearCase interface
cc-setup - MicroEmacs ClearCase interface setup
cc-add - MicroEmacs ClearCase interface - add file
cc-checkout - MicroEmacs ClearCase interface - checkout files and directories
cc-checkin - MicroEmacs ClearCase interface - checkin changes
cc-delete - MicroEmacs ClearCase interface - delete file
cc-diff - MicroEmacs ClearCase interface - diff changes
cc-state - MicroEmacs ClearCase interface - list state of directory files
cc-update - MicroEmacs ClearCase interface - update directory files

## SYNOPSIS

**ClearCase**
**cc-setup**
**cc-add**
**cc-checkout**
**cc-checkin**
**cc-delete**
**cc-diff**
**cc-state**
**cc-update**

## DESCRIPTION

The ClearCase and sub-commands provide MicroEmacs with an interface to **IBM Rational Software ClearCase**(1). **ClearCase** is a version control system; using it, you can record the history of your source file modifications. ClearCase is licensed from IBM Rational and is widely used in a commercial environment, see the documentation provided with ClearCase for more information on its features and use.

The **ClearCase** directory should be defined to MicroEmacs using **cc-setup**(3) which enables extensions in the **file-browser**(3) with an additional "`*cc-console*`" window. The "`*files*`" window includes additional columns showing the ClearCase state, revision and repository date. The functionality of the file-browser is the same as a non-ClearCase folder with the exception that additional ClearCase item controls are located in the mouse context menu (opened by clicking the right mouse button in the `*files*` buffer). This menu item opens another sub-menu providing access to the items below. When selecting commands than a pop-up check-box may be displayed which allows options to be selected for the command.

**Current State**

Checks the state of selected file(s) or directory against the repository. Differences detected are shown in the `*cc-state*` window. This provides a quick method of determining the working directory state.

The `*cc-state*` buffer supports some of the same features as a directory listing, the files can be loaded (using `return` or left click on the file name), selected (using space or left mouse button on the left hand edge) and ClearCase commands executed from the window operate either on the selected files or the listed files if no files are selected.

**Update Elements**

Updates the currently selected files, files are selected by clicking the left button to the left of the required file name. Multiple files may be selected by 'dragging' a hilight region over the required files.

**Clear CC console**

Clears the **cc-console** buffer.

The following sections describe some of the ClearCase commands that are available, a full list of commands may be found using **command-apropos**(2) searching for **cc** which provides a full list of all of the ClearCase related commands.

The **cc-add** command adds the current buffer file to the repository.

The **cc-checkout** command checks out a file or directory from the repository into the current directory, the read-only status of the file is changed to read-write, the user is prompted for a comment which will be used on a subsequent check-in operation. The user specifies the file on the message line.

The **cc-checkin** command commits any changes made to the currently buffer's file (including additions) to the repository. The user is prompted for a commit log message, the file state will change to read-only.

The **cc-diff** command opens a **cc-diff** window displaying the differences between the current buffer's local file and repository version. If the current buffer is a directory list it will list all the differences found in all files within the directory.

The **cc-delete** command removes the current buffer's file from the repository.

The **cc-state** command opens a **cc-state** window listing the state of any file in the current directory which is not up-to-date. Note that unlike most ClearCase sub commands this command executes over all files in the current buffer's file directory.

The **cc-update** command updates all files in the current directory, the output being reported to a new **cc-update** window. Note that unlike most ClearCase sub commands this command executes over all files in the current buffer's file directory.

## Setup

**cc-setup** provides a dialog interface to configure the MicroEmacs ClearCase interface if required. The VOB view location in the file system should be added for each view that is to be access by MicroEmacs.

A **Tools** tab exists to configure each of the **cleartool**(1) options, these may be configured to suite the users preferences.

**cc-setup** may also be accessed from **user-setup**(3) from the **Miscellaneous** configuration.

## ClearCase File Browser

The **ClearCase File Browser** is an extension to the **file-browser**(3) and shows the state of ClearCase files in a directory listing.

## NOTES

**ClearCase** and sub-commands are macros defined in file `cc.emf`.

## SEE ALSO

**git**(3), **svn**(3), **cvs**(3), **file-browser**(3), **$rcs-file**(5).

Ref: ClearCase(3) File: m3mac085.3 Date: 2026/07/20

## NAME

collapse-current - (un)Fold a region in the current buffer
collapse-all - (Un)Fold all regions in the current buffer

## SYNOPSIS

**collapse-current**
**collapse-all**

## DESCRIPTION

MicroEmacs provides a generic, albeit coarse, folding mechanism which is applied to some of the well known file modes. The folding mechanism allows parts of the buffer to be scrolled up and hidden, leaving a residue hilighting marker within the buffer indicating a folded region. A folded buffer typically allows a summary of the buffer contents to be viewed within several windows, hiding the detail of the buffer.

The folding mechanism uses well defined *start* and *end* markers which form part of the syntax of the well known file mode. i.e. in 'C' this is the open and closed braces that appear on the left-hand margin ({ .. }). The intention is that the natural syntax of the text is used to determine the fold positions, requiring no additional text formating or special text tags to be inserted by the user.

**collapse-current** opens and closes a folded region within the buffer. If the current cursor position lies between a *start* and *end* marker then the region between the start and end is folded out and hidden from view, leaving a highlight marker in the buffer. If the fold already exists then, moving the cursor to the folded line and invoking **collapse-current** removes the fold marker and reveals the text.

**collapse-all** opens and closes all folded regions within the buffer, if the current state is unfolded then all of the *start*/*end* markers are located and their regions folded. Conversely, if the buffer is currently folded and **collapse-all** is invoked, then all folds are removed and the associated text revealed.

## CONFIGURATION

In order to utilize the **collapse-current/all** commands within a buffer, the *start* and *end* markers have to be initialized for the syntactical contents of the buffer. This is performed within the hook function for the buffer, using the hook-name, the initialization is performed within **buffer-init**(3). Buffer specific variables are defined within the context of the buffer to configure that start and end fold handling. The buffer specific variables are defined as follows, where *XXX* is the file hook base name.

**.fhook-***XXX***-collapse-open**
: A regular expression search string used to locate the start of the string. For speed the search string should include a regular expression start or end of line character whenever possible. i.e. in C the open is defined as "^{".

**.fhook-***XXX***-collapse-close**
: A regular expression search string used to locate the end of the string. For speed the search string should include a regular expression start or end line character whenever possible. i.e. in C the close is defined as "^}".

**.fhook-*XXX*-collapse-mopen**
: An integer value that denotes the number of lines to move in a forward or (negative n) backward direction from the *start* line located by the search string to the position in the buffer to be folded. If default value when **mopen** is omitted is 0, starting the fold from the search string line.

**.fhook-*XXX*-collapse-mclose**
: The relative displacement from the close fold line to the fold position, this is a positive or negative displacement depending on where the fold is to be positioned.

**.fhook-*XXX*-collapse-mnext**
: Specifies the number of lines to advance before the next search is continued on the fold operation. This is only used by **collapse-all**.

## PRESENTATION

The presentation of a collapsed region takes the form:

*<open line string> ... <close line string>*

Where *<open line string>* and *<close line string>* are automatically extracted from the originating buffer using the search string definition. Existing **collapse** implementations used a simpler format:

*<open line string> ...*

In MicroEmacs '04 and later versions, then the *<close line string>* is always appended. Appending of the closing token is required to ensure that any indentation is maintained. This allows the indent processing to find both an *open* and *close* character sequence and not introduce an unwanted indentation. To ensure that indentation consistency is maintained then the **.fhook-XXX.collapse-close** definition should minimally match the corresponding indentation close regular expression.

## EXAMPLE

The following examples show how the collapse variables are set up in each of the buffer modes.

### C and C++

**C** and **C++** collapse on the open and close brace appearing in the left-hand margin. The collapse variables are defined in `hkc/hkcpp.emf` as follows:

```
set-variable .fhook-c.collapse-open  "^{"
set-variable .fhook-c.collapse-close "^}"
set-variable .fhook-c.collapse-mopen  "-1"
set-variable .fhook-c.collapse-mclose "1"
set-variable .fhook-c.collapse-mnext "-1"
```

Given a 'C' function definition:

```
static void
myfunc (int a, int b)
{
    /* Function body */
}
```

the folded version appears as follows:

```
static void
myfunc (int a, int b) ... }
```

### emf

MicroEmacs macro files **emf** support folding of macro definitions, the collapse variables are defined in `hkemf.emf` as follows:

```
set-variable .fhook-emf.collapse-open  "^0? ?define-macro[ \t]"
set-variable .fhook-emf.collapse-close "^!emacro"
set-variable .fhook-emf.collapse-mclose "1"
set-variable .fhook-emf.collapse-mnext "-1"
```

Given a macro definition:

```
0 define-macro mymacro
; This is the body of the macro
; ... and some more ...
!emacro
```

the collapsed version of the macro is defined as:

```
0 define-macro mymacro ... !emacro
```

### nroff

**nroff** is configured for manual pages only and folds between `.SH` and `.SS` sections, the hook variables are defined as follows:

```
set-variable .fhook-nroff.collapse-open  "^\.S[SH]"
set-variable .fhook-nroff.collapse-close "^\.S[SH]\|\'"
set-variable .fhook-nroff.collapse-mnext "-1"
```

Given an nroff block of text defined as:

```
> .SH SYNOPSIS
> Some text
> Some more text
> .SH DESCRIPTION
```

Then the collapsed version appears as:

```
> .SH SYNOPSIS
> .SH DESCRIPTION
```

### tcl/tk

**tcl/tk** is configured to collapse procedures. The collapse variables are defined as:

```
set-variable .fhook-tcl.collapse-open  "^proc "
set-variable .fhook-tcl.collapse-close "^}"
set-variable .fhook-tcl.collapse-mclose "1"
set-variable .fhook-tcl.collapse-mnext "-1"
```

Given a tcl procedure definition:

```
proc tixControl:InitWidgetRec {w} {
    upvar #0 $w data

    tixChainMethod $w InitWidgetRec

    set data(varInited)   0
    set data(serial)    0
}
```

The collapsed version of the same section appears as:

```
proc tixControl:InitWidgetRec {w} { ... }
```

## NOTES

**collapse-current** and **collapse-all** are macros implemented in `collapse.emf`. The collapsing is performed using the **narrow-buffer**(2) command.

**collapse-current** may also be bound to the mouse using the **user-setup**(3). The typical binding is `C-mouse-drop-1`.

## HISTORY

**collapse** is the new name for **fold** appearing in MicroEmacs 2004 version. The existing commands of **fold-current** and **fold-all** have been replaced by **collapse-current** and **collapse-all** respectively. **collapse** now uses the private command variables.

## SEE ALSO

File Hooks, **buffer-init**(3), **indent**(2), **user-setup**(3), **narrow-buffer**(2).

## NAME

comment-line - Comment out the current line
uncomment-line - Uncomment current line
comment-to-end-of-line - Extend comment to end of line
comment-restyle - Reformat the current comment
comment-start - Start a new comment
comment-end - End the current comment

## SYNOPSIS

*n* **comment-line**
*n* **uncomment-line**
**comment-to-end-of-line**
**comment-restyle**
**comment-start**
**comment-end**

## DESCRIPTION

The action of the **comment** commands are file type specific (comments in **C** are `/* ... to ... */` where as MicroEmacs macro file comments are `; ... to the end of line`) so the commands must be configured for each file type (see the NOTES section below). The configuration is automatically performed by almost all the standard file hooks released with MicroEmacs by the file hook so these commands should be fully functional.

**comment-line** comments out the current region, if active and *n* is omitted, or the current and next *n*-1 lines (default when *n* is omitted is to comment out just the current line). The cursor is then moved to the start of the next line.

**uncomment-line** behaves in a similar way, except that for a file type that has a start and end comment token, like **C**, it can effect lines outside the current region or line count if a comment goes beyond the start or end of the defined region. Note that **uncomment-line** also attempts to undo the effects of re-indenting a line that has been commented out, hopefully restoring the original indentation.

**comment-to-end-of-line** inserts *comment-pad*s (see NOTES) up-to the **$buffer-fill-col**(5) and then terminates the comment with the *comment-end* string. **comment-restyle** reformats the text within the comment, filling text lines to the **$buffer-fill-col** and regenerating any boxing and divide lines.

**comment-start** opens a new "***comment***" buffer which is configured for writing a text comment, the user can then type in the comment with all the benefits of MicroEmacs in a plain text editor. Once the comment is complete use the **comment-end** command to insert the comment into the source file, this comment is locally bounded to "**C-c C-c**". The styling of the comment is controlled by the *comment-flag* setting (see NOTES).

## NOTES

Consider the structure of a box comment to be as follows:

```
<comment-start><comment-pad><comment-pad><comment-pad><comment-box-right>
<comment-box-left> COMMENT TEXT ... COMMENT TEXT      <comment-box-right>
<comment-box-left> COMMENT TEXT ... COMMENT TEXT      <comment-box-right>
<comment-box-left><comment-pad><comment-pad><comment-pad><comment-end>
```

The comment commands are configured by the single file hook command variable **.fhook-**<*type*>**.comment** where <*type*> is the file type label. The structure of the variable is a list with the following format:

```
|<comment-start>|<comment-end>|<comment-pad>|<comment-box-left>|...
    ...<comment-box-right>|<comment-flags>|
```

Where | is the list divide character. The <*comment-flags*> are a list of character flags which are defined as follows:

| | |
|---|---|
| b | Box format required, i.e. create right edge using <*comment-box-right*>. |
| f | Footer line required. |
| F | Fill footer line with <*comment-pad*> strings. |
| h | Header line required. |
| H | Fill header line with <*comment-pad*> strings. |
| j | Enable Justify mode in *comment* buffer. |
| r | A restyle key binding is added to the buffer "esc o" and "esc q" buffer key bindings are added and bound to **comment-restyle**(3). |

## EXAMPLE

The following **comment** is the standard **C** setting:

```
set-variable .fhook-c.comment "|/*| */|*| * | * |f|"
```

This can be used to create comments of the form:

```
/* comment-line comments out the current and next n-1 lines (default
 * when n is omitted is to comment out just the current line). The
 * cursor is then moved to the start of the next line.
 *
 * uncomment-line behaves differently depending on whether the file
 * type terminates a comment with an end token or simply by the end
 * of the line.
 */
```

A box style comment can be generated by changing the <*comment-flags*> form f to bfFhH, producing:

```
/**********************************************************************
 * comment-line comments out the current and next n-1 lines (default *
 * when n is omitted is to comment out just the current line). The   *
 * cursor is then moved to the start of the next line.               *
 *                                                                   *
 * uncomment-line behaves differently depending on whether the file  *
 * type terminates a comment with an end token or simply by the end  *
 * of the line.                                                      *
 **********************************************************************/
```

## SEE ALSO

**buffer-init**(3), File Hooks.

Ref: comment-line(3) File: m3mac060.3 Date: 2025/07/17

## NAME

compile - Start a compilation process recompile - Rerun compilation process

## SYNOPSIS

*n* **compile** ["*auto-touch-file*"] "*compile-command*"
**recompile**

## DESCRIPTION

**compile** gets and executes the compile command using a pipe execution (incremental pipe on UNIX platforms), loading the output into a buffer called "***compile***", with go to error parsing using the command **get-next-line**(2). The default compile execution is set by variable **%compile-com**(5), the error parsing is setup using the command **add-next-line**(2).

Before the compile command is executed **save-some-buffers**(2) is executed to allow the user to ensure that all relevant buffers are saved.

Compile uses command **ipipe-shell-command**(2) to launch the given system *compile-command*. By default the output lines in the window are wrapped at 2046 characters as a balance between reliably searching for errors and usability. See flag **0x02** below or help on **ipipe-shell-command**(2) for details on how to change this behaviour.

The argument *n* can be used to change the default behavior of compile described above, *n* is a bit based flag where:

**0x01** Enables the user prompt before saving buffers, if this flag is not supplied then all modified visible buffers will be written.

**0x02** When set the compile output lines will be wrapped at the width of the current window.

**0x04** When set the previous *compile-command* is automatically used for this compile.

**0x08** When set the previous compile location (path) is automatically used for this compile.

**0x10** When set the compile prompts for an *auto-touch-file*, when set to a file name this file is automatically touched (modification timestamp updated without changing the files content) on this and any subsequent compiles. The feature is disabled if set to an empty string.

**recompile** reruns the previous compile process, it is implemented by executing compile with flags 0x04 and 0x08, the other flags of compile can also be passed to this command.

## NOTES

**compile** and **recompile** are macros defined in `tools.emf`.

## SEE ALSO

**add-next-line**(2), **%compile-com**(5), **get-next-line**(2), **save-some-buffers**(2), **grep**(3).

Ref: compile(3) File: m3mac000.3 Date: 2022/06/12

## NAME

copy-file - Copy or move files and directories
delete-file - Delete files and directories
delete-file - Delete file after a specified delay

## SYNOPSIS

*n* **copy-file** "*from*" "*to*"
*n* **delete-file** "*file*"
*n* **delete-file-delayed** "*file*" [ "delay" ]

## DESCRIPTION

**copy-file** copies or moves file or directory *from* to file name or directory *to*, particular attention should be paid to the last character of *to*, as a trailing '/' is assumed to be a destination directory as opposed to the destination name. For example:

```
copy-file "a/" "b"
```

will copy directory `a` to `b`, effectively renaming it, whereas:

```
copy-file "a/" "b/"
```

will copy directory `a` into directory `b/`. The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When set **copy-file** copies *from*, when clear **copy-file** moves *from* to *to*.

**0x02** When set the command will not prompt the user on a file read failure, the copy/move for that file will simply fail.

**0x04** When set the command will not prompt the user on a file create or write failure, the copy/move for that file/directory will simply fail.

**0x08** When set the command will not prompt the user on a file overwrite issue, 'no' is assumed for all, i.e. no files will be overwritten.

**0x10** When set the command will not prompt the user on a file overwrite issue, 'yes' is assumed for all, i.e. any file will be overwritten if required.

**0x20** When set all intermediate directories are create as required, i.e. if *to* evaluates to a directory that does not currently exist it will be created. If this bit is clear the command will fail if the destination directory does not yet exist.

**0x40** When set the file name part of *from* is treated as a regular expression, as this can match multiple files *to* must be a directory.

**0x80** When set an error is not thrown if *from* does not match a file.

**0x100** When set with bit **0x40** the given *from* regex mask is used when coping each matching sub-directory, for example if the *from* mask was ".*[^ ]", which excludes files ending with a ' ' (commonly backup files), the setting of this bit will ensure no backup files will be copied from the root directory or any of its sub-directories, whereas without this bit set backup files within sub-directories will be copied. When this bit is used,

the *from* mask should not have a path at the start, so the copy-file macro should be executed in the correct starting directory. This bit is ignored if bit **0x40** is not set.

**delete-file** deletes the given *file* or directory. The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When set the command will return an error if the given *file* does not exist or, if a regex mask (see bit **0x20** below), match at least one file or directory.

**0x02** When set the file is trashed first. While this is not guaranteed to put the file beyond recovery it should make it very much harder.

**0x04** When set the command will not prompt the user on a delete failure, the delete will simply fail and move on to the next.

**0x08** When set the command will not prompt the user before deleting a read-only file, '*no*' is assumed and all read-only files will be skipped.

**0x10** When set the command will not prompt the user before deleting a read-only file, '*yes*' is assumed and all read-only files will be deleted.

**0x20** When set the file name part of *file* is treated as a regular expression, this can be used to delete multiple files.

**0x40** When set messages, including errors, are not printed.

Both commands return 4 counts in **$result**(5) with the form "—<ds#>—<df#>—<fs#>—<ff#>-
a MicroEmacs list, where the counts are defined as follows:

`<ds#>`
The number of directories successfully copied, moved or deleted.

`<df#>`
The number of directories that failed to be copied, moved or deleted.

`<fs#>`
The number of files successfully copied, moved or deleted.

`<ff#>`
The number of files that failed to be copied, moved or deleted.

The **delete-file-delayed** command calls **delete-file** with the given numeric argument *n* and *file* after the given **delay**, which is specified in seconds. If **n** is not specified a default of `0x54` is used which suppresses error messages and will not prompt the user. If the **delay** is not specified a default of `20` seconds is used. This command can only be used via macro code, if called directly the macro will simply check if the time for the next file deletion has be reached, it will not prompt the user for a file to delete.

## NOTES

**copy-file**, **delete-file** and **delete-file-delayed** are macros defined in `hkdirlst.emf`.

## SEE ALSO

**file-op**(2), **$result**(5).

Ref: copy-file(3) File: m3mac098.3 Date: 2026/08/18

## NAME

copy-rectangle - Copy a rectangular region of text
yank-rectangle-overwrite - Insert a column of text with overwrite

## SYNOPSIS

**copy-rectangle**
**yank-rectangle-overwrite**

## DESCRIPTION

**copy-rectangle** copies a rectangle (or column) of text defined between the cursor and the set-mark position. The text between the mark column and the cursor column is copied from every line between the mark line and the cursor line inclusive and copied to the kill buffer. The copied text may then be extracted from the kill buffer using **yank**(2), **yank-rectangle**(2) or **yank-rectangle-overwrite**.

The mark position may be ahead or behind the current cursor position. If the rectangle column boundary divides a tab character which spans multiple columns, the tab character is replaced with the equivalent number of spaces. Similarly if the boundary divides an unprintable character which is displayed using multiple characters (e.g. '^A' for character 0x01) then spaces are inserted before the character to move it to the right of the boundary.

**yank-rectangle-overwrite** replaces a column of text with the current kill buffer, overwriting existing text with the contents of the kill buffer. The kill buffer should contain a rectangle region acquired with **copy-rectangle** or **kill-rectangle**(2). The current cursor position identifies the top left of the column to be written, the width of the column to be replaced is determined by the width of the kill buffer text being inserted.

Where an overwrite is not required then use **yank-rectangle**(2).

## SEE ALSO

**copy-region**(2), **kill-rectangle**(2), **kill-region**(2), **reyank**(2), **set-mark**(2), **string-rectangle**(3), **undo**(2), **yank**(2), **yank-rectangle**(2).

Ref: copy-rectangle(3) File: m3mac072.3 Date: 2022/06/12

## NAME

copy-registry - Recursive copy of registry subtree

## SYNOPSIS

**copy-registry** "*from*" "*to*"

## DESCRIPTION

**copy-registry** is a macro language construct only that performs a recursive copy of a registry subtree, copying all child nodes of *from* and recreating the subtree at *to*, where *from* and *to* defines the paths to the registry tree.

## NOTES

The **copy-registry** command is not publically available from the command line, but may be used within macros to perform the copy.

This command is implemented as a recursive macro in `utils.emf`.

## SEE ALSO

**delete-registry**(2), **find-registry**(2), **list-registry**(2), **mark-registry**(2), **read-registry**(2), **save-session**(3), **set-registry**(2), **&reg**(4), **erf**(8).

Ref: copy-registry(3) File: m3mac036.3 Date: 2022/06/12

## NAME

cvs - MicroEmacs CVS interface
cvs-setup - MicroEmacs CVS interface setup
cvs-add - MicroEmacs CVS interface - add file
cvs-annotate - MicroEmacs CVS interface - annotate file
cvs-cat-file - MicroEmacs CVS interface - output specific revision of a file
cvs-checkout - MicroEmacs CVS interface - checkout files and directories
cvs-commit - MicroEmacs CVS interface - commit changes
cvs-diff - MicroEmacs CVS interface - diff changes
cvs-file-status - MicroEmacs CVS interface - status of files
cvs-gdiff - MicroEmacs CVS interface - graphical diff changes
cvs-log-file - MicroEmacs CVS interface - log changes
cvs-remove - MicroEmacs CVS interface - remove file
cvs-resolve - MicroEmacs CVS interface - resolve conflicts
cvs-status - MicroEmacs CVS interface - list state of directory files
cvs-tag - MicroEmacs CVS interface - tag files
cvs-update - MicroEmacs CVS interface - update directory files

## SYNOPSIS

*n* **cvs**
**cvs-setup**
**cvs-add**
**cvs-annotate**
**cvs-cat-file**
**cvs-checkout**
**cvs-commit**
**cvs-diff**
**cvs-file-status**
**cvs-gdiff**
**cvs-log-file**
**cvs-remove**
**cvs-resolve**
**cvs-status**
**cvs-tag**
**cvs-update**

## DESCRIPTION

The cvs and sub-commands provide MicroEmacs with an interface to **cvs**(1). **CVS** is a version control system; using it, you can record the history of your source file modifications. CVS is licensed under the GNU General Public License and is freely available on the Internet, see the documentation provided with CVS for more information on its features and use.

The MicroEmacs **cvs** command opens up a modified **file-browser**(3) with an additional "`*cvs-console*`" window. The "`*files*`" window includes additional columns showing the CVS state, revision and repository date. The browser by default opens with 3 windows including a representation of the file tree. When **cvs** is provided with a numeric argument *n* of zero (0) then a single window displaying the directory is used only (i.e. **0 cvs**)

The functionality of the file-browser is the same as a non-CVS folder with the exception that additional CVS item controls are located in the mouse context menu (opened by clicking the right mouse button in the `*files*` buffer). This menu item opens another sub-menu providing access to the items below. When selecting commands than a pop-up check-box may be displayed which allows options to be selected for the command.

**Current State**

Checks the state of selected file(s) or directory against the repository. Differences detected are shown in the `*cvs-status*` window. This provides a quick method of determining the working directory state.

The `*cvs-status*` buffer supports some of the same features as a directory listing, the files can be loaded (using `return` or left click on the file name), selected (using space or left mouse button on the left hand edge) and CVS commands executed from the window operate either on the selected files or the listed files if no files are selected. This runs the command "`cvs -nq update <files>`".

**Checkout files**

Checks out a file or directory from the repository into the current directory. The file or directory is specified by typing the name into a dialog which is opened when this option is selected. This runs the command "`cvs checkout <file>`".

**Update files**

Updates the currently selected files, files are selected by clicking the left button to the left of the required file name. Multiple files may be selected by 'dragging' a hilight region over the required files. This runs the command "`cvs update <files>`".

**Commit files**

Commits any changes made to the selected files back to the CVS repository. This runs the command "`cvs commit <files>`". The user is prompted for the message to add on the commit.

**Diff files**

Displays any differences between the selected files and the CVS repository version in the **cvs-console** window. This runs the command "`cvs diff <files>`".

**Log files**

Displays the CVS logs for the selected files in the **cvs-console** window. This runs the command "`cvs log <files>`".

**Status files**

Displays the CVS status for each of the selected files in the **cvs-console** window. This runs the command "`cvs status -v <files>`".

**Tag files**

Adds, removes or moves a CVS tag on the selected file(s) or directories. A pop-up window is displayed which allows entry of the tag name and define the operation. This runs the command "`cvs tag <files>`".

**Add files**

Adds the selected files to the CVS repository. Note this command only performs the local add, a **CVS commit** is required to make the addition permanent. This runs the command "`cvs add <files>`". A pop-up is displayed to allow the user to change the mode of the add to binary or text.

**Remove files**
This command is deliberately not implemented as its far to dangerous! Instead it opens a dialog informing the user to use the **cvs-remove** command instead.

**Graphical diff**
This command opens a **gdiff**(3) window showing the differences between the currently selected file and the CVS repository version. Note this command only works with a single file.

**Cat file** This command opens a *<*file-name*>:<*revision*> window containing the selected file as it was at a given revision, the revision being selected from the command's dialog. Note this command only works with a single file.

**Resolve conflicts**
This command may be used to resolve merge conflicts created by a *CVS update* operation. The command opens a **gdiff**(3) window showing the areas of conflict allowing the user to select the correct version and saving the resultant version back to the local file. Note this command only works with a single file.

**Clear cvs console**
Clears the **cvs-console** buffer.

The **cvs-add** command adds the current buffers file to the repository. Note that this command only performs the local addition, a *CVS commit* is required to make the addition permanent.

The **cvs-annotate** command displays information about the most recent change for each line of the current buffer's file, very useful when trying to locate the cause of a problem.

The **cvs-cat-file** command opens a *<*file-name*>:<*revision*> window with the contents set to the specified revision of the current buffer's file. The revision is selected from the command's dialog and may be given as `HEAD`, an explicit revision or tag, or a date.

The **cvs-checkout** command checks out a file or directory from the repository into the current directory. The user specifies the file on the message line.

The **cvs-commit** command commits any changes made to the currently buffer's file (including additions) to the repository. The user is prompted for a commit log message.

The **cvs-diff** command opens a **cvs-diff** window displaying the differences between the current buffer's local file and repository version. If the current buffer is a directory list it will list all the differences found in all files within the directory.

The **cvs-file-status** command opens a **cvs-file-status** window and shows the state of the current buffer in the window including the modification state, repository etc.

The **cvs-gdiff** command opens a **gdiff**(3) window displaying the differences between the current buffer's local file and repository version.

The **cvs-log-file** command opens a **cvs-log-file** window displaying the CVS log of the current buffer's file.

The **cvs-remove** command removes the current buffer's file from the repository - PLEASE NOTE THIS CAN LEAD TO LOST DATA!!! This command only performs the local removal; as it deletes the buffer and file the **cvs-commit** command

cannot be used to commit the removal to the CVS repository. Instead the main **cvs** file-browser menu or **cvs**(1) itself must be used.

The **cvs-resolve** command may be used to resolve any conflicts created by CVS when the current buffer's file is updated. The command opens a gdiff window displaying the areas of conflict, the user may then select the correct version in each case and save the resultant new version over the local file.

The **cvs-status** command opens a **cvs-status** window listing the state of any file in the current directory which is not up-to-date. Note that unlike most cvs sub commands this command executes over all files in the current buffer's file directory.

The **cvs-tag** command opens a dialog box and allows the user to add, remove or delete a CVS tag.

The **cvs-update** command updates all files in the current directory, the output being reported to a new **cvs-update** window. Note that unlike most cvs sub commands this command executes over all files in the current buffer's file directory.

### Setup

**cvs-setup** provides a dialog interface to configure the MicroEmacs CVS interface if required. On a typical system installation then it is sufficient to have **$CVSROOT** defined in the environment and the CVS tools will operate from this environment variable without any further configuration of MicroEmacs. If MicroEmacs CVS configuration is provided then this will over-ride the environment variable.

**cvs-setup** may also be accessed from **user-setup**(3) from the **Miscellaneous** configuration. The the dialog provides the following configuration controls:

**CVS Command Setup**

`CVSROOT`
: If set, overrides the CVS **$CVSROOT** environment variable.

`CVS_RSH`
: If set, overrides the CVS **$CVS_RSH** environment variable. rsh.

`CVS Command`
: Defines the **cvs**(1) system command. The default when not defined is **cvs**.

`Compression`
: Sets the compression level to use with all CVS commands.

`Quieter CVS output`
: When enabled the `-q` CVS command-line option is used to suppress some of the more verbose print-out.

### CVS File Browser

The **CVS File Browser** is an extension to the **file-browser**(3) and shows the state of CVS files in a directory listing.

`Quick browser mode`
: When enabled the CVS file browser does not launch a cvs command to determine the current state of all files in the directory, instead it extracts as much information as it can from the local file system, inserting '`Q`'s for information it cannot determine. This is useful when the connection to the server is slow. The full CVS information can be obtained by executing the command **cvs-full-reread**, typically bound to `C-f5`.

### Tool Defaults

The following define the default settings for options used in various CVS commands:

`Recurse into sub-directories`

`Check-out missing directories`

`Prune empty directories`

### NOTES

**cvs** and sub-commands are macros defined in file `cvs.emf`.

When filtering is enabled in the file browser the filter used by the CVS interface depends on the file browser mode, if **Quick browser mode** is enabled the same filter is used, but when disabled all files ignored by **cvs**(1) are also removed by the filter. Use **buffer-setup**(3) in the buffer to enable/disable the filter.

The cvs-log, cvs-state, cvs-status and cvs-resolve-conflicts commands were renamed to cvs-log-file, cvs-status, cvs-file-status and cvs-resolve in August 2026 to align them with the equivalent SVN and VCS commands.

### SEE ALSO

**git**(3), **svn**(3), **ClearCase**(3), **file-browser**(3). **$rcs-file**(5).

Ref: cvs(3) File: m3mac057.3 Date: 2026/08/03

## NAME

cygwin - Open a Cygwin BASH window
%cygwin-path - Cygwin root directory
%cygwin-hilight - Cygwin shell hilight enable flag
%cygwin-prompt - Cygwin shell prompt

## PLATFORM

Windows '95/'98/NT/2K/XP/Vista/7/10/11 - windows ONLY

## SYNOPSIS

**cygwin**
***%*cygwin-path** "*path*"
***%*cygwin-hilight** [0—1]
***%*cygwin-prompt** "*hilightString*"

## DESCRIPTION

**cygwin** creates an interactive BASH shell window within a MicroEmacs buffer window, providing a UNIX command line facility within the Microsoft Windows environment. This is a preferable environment to the MS-DOS shell and is certainly far more comfortable for those familiar with UNIX.

Within the window BASH commands may be entered and executed, the results are shown in the window. Within the context of the BASH shell window then directory naming conforms to the **cygwin** standard conventions (as opposed to the Microsoft directory naming).

On running **cygwin** a new buffer is created called `*cygwin*` which contains the shell. Executing the command again creates a new shell window called `*cygwin1*`, and so on. If a cygwin window is killed off then the available window is used next time the command is run.

Additional controls are available within the shell window to control the editors interaction with the window. The operating mode is shown as a digit on the buffer mode line (2/3/4) which is defined in the following sections. The operational mode is changed with the **F3** function key.

**Mode 2 - Browse Mode**
: Locks the window and allows local editing to be performed. All commands entered into the window are interpreted by the editor. Mode 2 is typically entered to cut and paste from the window, search for text strings etc. In mode 2, a **2** is shown on the mode line.

**Mode 3 - Semi-immersion**
: The normal operating mode, text typed into the window is presented to the shell window. Translation of MicroEmacs commands (i.e. beginning-of-word) are translated and passed to the shell. For interactive use this is the default mode. In mode 3, a **3** is shown on the mode line.

**Mode 4 - Full Immersion**
: All input is passed to the shell, no MicroEmacs commands are interpreted and keys are passed straight to the shell window. This mode is used where none of the keys to be entered are to be interpreted by MicroEmacs. Note that you have to un-toggle mode 4 before you can swap buffers as this effectively locks the editor into the window.

**F5** Clears the buffer contents. This simply erases all of the historical information in the buffer. The operation of the shell is unaffected.

To exit the shell then end the shell session using "`exit`" or "`C-d`" as normal and then close the buffer. A short cut "`C-c C-k`" is available to kill off the pipe. However, it is not recommended that this method is used as it effectively performs a hard kill of the buffer and attached process

**%cygwin-path** is a user defined variable that defines the file system location of the *cygwin* directory, this is typically `c:/cygwin`. If *cygwin* is located at a different location then the variable MUST be defined within the user start up script in order for the **cygwin** command to start the shell. With a default installation of *cygwin* then the settings are typically defined as:
**Current Release**

```
set-variable %cygwin-path "c:/cygwin"
```

**%cygwin-hilight** is a boolean flag which controls how the cygwin command shell window is hilighted. This value MUST be defined within the user start up script prior to executing cygwin if hilighting is to be disabled; by default hilighting is enabled. A value of 0 disables shell hilighting i.e.

```
set-variable %cygwin-hilight 0
```

**%cygwin-prompt** is an optional variable that is used in conjunction with **%cygwin-hilight**, it defines the hilighting string identifying the prompt. This allows the prompt to be rendered with a different color. The default prompt is `bash-x.yyr$` and may be hilighted using a definition:

```
set-variable %cygwin-prompt "bash-2.01$"
```

The user typically overrides the prompt definition within the BASH startup file, a more appropriate definition of the prompt may be:

```
set-variable %cygwin-prompt "^[a-z]*@[^>]*>"
```

## NOTES

The **cygwin** command uses the **ipipe-shell-command**(2) to manage the pipe between the editor and the **bash** shell. The window is controlled by the macro file `hkcygwin.emf` which controls the interaction with the shell.

The macro **cygwin** in `hkcygwin.emf` defines the parameter setup to connect to the cygwin bash shell, installed in the default location `c:/cygwin`. If your installation of cygwin is in a different location then correct the macro to match your install location, preferably correct by creating a *mycygwin.emf* file in your user directory simply containing a re-defined **cygwin** macro.

If you have exported some of the cygwin environment variables in your `autoexec.bat` then you will have to figure out for yourself what variables macro *cygwin* needs to export - the current configuration is for a vanilla install.

The **bash** shell is executed with options **-i**, for interactive shell and **-m** to enable job control.

As of the 2004 release then the **Cygwin/Cygnus** support has been revised in light of the changes in **cygwin**

- Renamed the command from **cygnus** to **cygwin**.
- Following a change in the **bash**(1) command line operation, which now no longer echos user input, then the user input is processed by MicroEmacs. This does cause some problems with completion as the Cygwin environment has a different name space. The completion still requires some more work, but sufficient for smaller tasks.

## TESTED CONFIGURATIONS

This configuration has only been tested on a Windows 2000 installation, whether this works on Windows 95/98/NT is unknown while XP is very likely to work.

We have only been running ”make” operations in the shell and do not know how the likes of ”more”, ”man” or anything other terminal interaction works.

**Tested Configurations**

Windows 2K (Pentium-4) and Windows 2K (SunPCi-III)

cygwin latest version as of February 2004.

## BUGS

**Break Key**

A break in a bash shell is `C-c`, the macros define the key `C-c C-c` to perform the break. There does not appear to be a way to prevent this from happening as trapping the interrupt does not appear to be supported in Cygwin Bash i.e. ”`trap '' INT`”.

**Completion**

Completion processing needs to be improved. The directory location is retrieved from **bash**(1) on each command line prompt, this is used for file completion. There is no command completion. Room for some improvement here.

## SEE ALSO

**ipipe-shell-command**(2), **ishell**(3).
Cygwin home site **www.cygwin.com**

Ref: cygwin(3) File: m3mac033.3 Date: 2024/07/16

## NAME

delete-hilight-lines - Delete hilight lines

## SYNOPSIS

**delete-hilight-lines**

## DESCRIPTION

**delete-hilight-lines** deletes all lines in the buffer that are hilighted with a line color scheme applied by **osd-search**(3) or **line-scheme**(3). Deletion is only performed on the user hilight lines, **collapse-current**(3) annotation lines that use a line based hilight scheme are not removed.

The line hilighting may be used to mark up lines that are not required, typically in debugging logs, and then delete those lines using **delete-hilight-lines**.

## SEE ALSO

**collapse-all**(3), **collapse-current**(3), **hilight**(2), **osd-search**(3), **line-scheme**(3), **$line-scheme**(5).

Ref: delete-hilight-lines(3) File: m3mac070.3 Date: 2022/06/12

## NAME

delete-indentation - Join 2 lines deleting white spaces

## SYNOPSIS

*n* **delete-indentation**

## DESCRIPTION

**delete-indentation** deletes all white characters between the beginning of the current line and the end of the previous line, including the line-feed. If the current line is not empty then a space is inserted to divide the two lines now joined.

If a positive argument *n* is given then the process is repeated *n* times. Note that the deleted characters are not added to a kill buffer.

If a negative argument *n* is given then lines are joined without a separating space.

## NOTES

**delete-indentation** is a macro defined in `format.emf`.

## SEE ALSO

**delete-blank-lines**(2), **clean**(3), **kill-line**(2).

Ref: delete-indentation(3) File: m3mac021.3 Date: 2025/03/30

## NAME

diff - Difference files or directories
diff-changes - Find the differences from a previous edit session
rdiff - Recursively compare files or directories
%diff-com - Diff command line

## SYNOPSIS

**diff** "*old-version*" "*new-version*"
*n* **diff-changes**
**rdiff** "*old-version*" "*new-version*"
`%diff-com` "*string*"; Default is "`diff`"

## DESCRIPTION

**diff** executes the **diff**(1) command with the command line set by the **%diff-com**(5) variable and the user supplied *old-version* and *new-version*. The output of the command is piped into the ***diff*** buffer and is hilighted to show the changes (GNU diff only).

Your version of **diff**(1) will determine whether it is possible to difference directories.

**diff-changes** is a simple macro that diff's the current buffer with the last backup of the associated file. It is a quick way to determine what has been modified recently. This macro only works if a backup file exists. The numerical argument *n* can be used to diff against older backups if multiple backups are enabled (see **$kept-versions**(5)), for example if *n* is `2` then the diff will with the file's `.2` backup file. Note that this is a zero based index as the first backup has the extension `.0`.

**rdiff** operates in the same way as **diff** except it launches the **diff**(1) command with an extra *-r* option which enables recursive compare (i.e. compares the contents of sub-directories as well). Your version of **diff** must support this option for **rdiff** to operate correctly.

**%diff-com** is the command line that is used to execute a **diff**(1) system command.
For GNU diff then the following command line setting is recommended:

```
diff --context --minimal --ignore-space-change \
    --report-identical-files --recursive
```

which should be defined in your personal user configuration. This is the default for Linux.

## NOTES

**diff**, **diff-changes** and **rdiff** are macros defined in `tools.emf`.

**diff**(1) must be executable on the system before diff or diff-changes can function.

**diff**(1) is a standard utility on UNIX systems. For Windows 95/NT a version of GNU **diff** may be found at:

*<ftp.winsite.com/ftp/pub/pc/winnt/misc/gnudiff.zip>*

For MS-DOS users, a DJGPP port of **diff** is also available on the net. A commercial version of **diff** is also available from MKS.

## SEE ALSO

**compare-windows**(2), **compile**(3), **gdiff**(3), **xgrep**(3), **%xgrep-com**(5), **xdiff**(3), **%xdiff-com**(5).

Ref: diff(3) File: m3mac035.3 Date: 2025/07/24

## NAME

diff-lines - Highlight the differences between two lines

## SYNOPSIS

*n* **diff-lines**

## DESCRIPTION

**diff-lines** opens a window showing the differences found between two lines. If there is an active region the two lines compare are the lines containing the mark with the line containing the cursor. If the is no active region the current line is compared with the next.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When set the command copies the two lines into a new `*diff-lines*` buffer and displays it in popup window. When clear the command operates in the current buffer directory, in which case it is the caller's responsibility to ensure the buffer's highlighting has been setup correctly, see notes below.

**0x02** If set the mark position is always used to locate 2nd line, regardless of whether the region is active.

**0x04** If set the comparison is made from the current column of the cursor (and mark if region is used) to the end of the line, this can be useful when the start of the lines have an ignorable difference such a +/- diff line markers.

**0x08** If set then the two lines are not wrapped, if clear the lines are wrapped at the width of the display window.

## NOTES

**diff-lines** inserts the following **hilight**(2) tags into the two lines:

`\CCM` Start of a region of common change.

`\CCN` Start of a region of added text.

`\CCO` Start of a region of removed text.

`\CCA` End of the current region.

**diff-lines** is a macro defined in `diffline.emf`.

## SEE ALSO

**compare-windows**(2), **diff**(3).

Ref: diff-lines(3) File: m3mac104.3 Date: 2026/03/26

## NAME

display-matching-fence - Display the matching bracket

## SYNOPSIS

*n* **display-matching-fence**

## DESCRIPTION

**display-matching-fence** draws the fence (or bracket) pairing the one the cursor is currently over. A fence is considered to be one of the following:

```
{...}   (...)   [...]
```

If the matching fence is currently being drawn (i.e. it is visible) both fences are drawn in the '*Normal*' Matching Fence scheme (see **scheme-editor**(3)). If the matching fence is not currently visible the cursor is temporarily moved to the matching fence for **$pause-time**(5) milliseconds before returning to the starting position, the fences are hilighted using the Matching Fence 'Current' scheme. The matching fence delay can be interrupted by pressing any key. If the fence cannot be matched the fence is hilighted using the 'Select' scheme which is usually a bold red color.

The numeric argument **n** passed to the command is a bitwise flag where each bit is defined as follows:

**0x01** Display fence (if not set nothing is done).

**0x02** Use set-position id '`\x85`' instead of '`\x84`' (for internal use).

**0x04** Don't Jump when matching fence is off screen.

**0x08** Jump when closing a fence and its pair is off screen (for internal use).

**0x10** Always jump to matching fence when closing a fence (for internal use).

**0x20** Give preference to closing fence to left of cursor rather than character under the cursor (for internal use).

## NOTES

This macro is used by the **Fence Display** setting of **user-setup**(3), the macro is bound to the `idle-pick` event using some of the more obscure numeric argument flags.

## SEE ALSO

**goto-matching-fence**(2), **user-setup**(3), **scheme-editor**(3), **$pause-time**(5).

Ref: display-matching-fence(3) File: m3mac061.3 Date: 2022/06/12

## NAME

display-whitespaces - Toggle the displaying of white characters

## SYNOPSIS

**display-whitespaces**

## DESCRIPTION

**display-whitespaces** toggles the displaying of white characters in the main display. By default white characters, space tab and new-lines, are represented with invisible characters such as one or more ’ ’s for spaces and tabs and text moving to the next line for new-lines. The user can make this characters become ’visible’ using this function.

When this function is first called it toggle enables the displaying of these characters, other characters are drawn in their place to make them visible. A subsequent call will disable the displaying of them.

## NOTES

**display-whitespaces** is a macro implemented in `misc.emf` and uses bit `0x80000` of the **$system**(5) variable.

The displaying of white characters can be enabled or disabled at start-up using **user-setup**(3).

This feature may be more confusing on some terminals due to the lack of characters available for displaying the white characters. The characters used when displaying white characters are defined in the variable **$window-chars**(5).

The macro **display-white-chars** was renamed to **display-whitespaces** in September 2006.

## SEE ALSO

**$system**(5), **user-setup**(3), **$window-chars**(5).

Ref: display-whitespaces(3) File: m3mac051.3 Date: 2022/06/12

## NAME

draw - Simple line drawing utility

## SYNOPSIS

**draw**

## DESCRIPTION

**draw** provides a simple way of drawing lines into the current buffer, this has a variety of uses such as drawing tables. **draw** copies the current buffer into a temporary buffer and then allows the user to draw using simple commands until the user either aborts, discarding any changes, or exits insert the changes back into the buffer.

The keys for **draw** are defined as follows:

**esc h** Display a help dialog.

**up**, **down**, **left**, **right**
The cursor keys (or any other keys bound the the same commands) will move the cursor, drawing in the current mode.

**d** Change the current mode to **d**raw (default), cursor movement will result in drawing in the current style.

**e** Change the current mode to **e**rase, cursor movement will result in erasing to spaces.

**m** Change the current mode to **m**ove, no drawing is performed with cursor movement.

**u** Change the current mode to **u**ndo, cursor movement will result in undoing the character to the original or a space.

**-** Sets the current horizontal line drawing style to use '−'s (default).

**=** Sets the current horizontal line drawing style to use '='s.

**C-g** Abort - changes are lost.

**return** Exit, inserting any changes into the current buffer.

## NOTES

**draw** is a macro defined in `draw.emf`.

Ref: draw(3) File: m3mac032.3 Date: 2022/06/12

## NAME

edit-dictionary - Insert a dictionary in a buffer
restore-dictionary - Save dictionary user changes

## SYNOPSIS

**edit-dictionary** "*dictionary*"
**restore-dictionary**

## DESCRIPTION

**edit-dictionary** dumps the contents of "*dictionary*" into the temporary buffer "**dictionary**", if this buffer already exists then **edit-dictionary** simply swaps to this buffer. This enables the user to correct and prune the words in any dictionary. The given dictionary must have already been added as a main dictionary using **add-dictionary**(2).

The format of the created buffer is one word on each line, each word takes one of the following 3 forms:

`xxxx` - Good word `xxxx` with no spell rules allowed
`xxxx/abc` - Good word `xxxx` with spell rules `abc` allowed
`xxxx>yyyy` - Erroneous word with an auto-replace to `yyyy`

Executing **restore-dictionary** in a buffer created by **edit-dictionary** will first call **delete-dictionary**(2) to remove the original dictionary from memory. It then uses **add-dictionary**(2) to create a new dictionary with the same name and then uses **spell-add-word**(3) to add all the words in the current buffer into the new dictionary.

**restore-dictionary** does not save the new dictionary.

## NOTES

**edit-dictionary** and **restore-dictionary** are macros defined in file `spellutl.emf`. They are not defined by default so *spellutl.emf* must be executed first using **execute-file**(2).

## SEE ALSO

**spell-add-word**(3), **add-dictionary**(2), **save-dictionary**(2), **delete-dictionary**(2).

Ref: edit-dictionary(3) File: m3mac022.3 Date: 2022/06/12

## NAME

encase - Encase a region with start/end string

## SYNOPSIS

*n* **encase** "*start-string*" "*end-string*"

## DESCRIPTION

**encase** provides a short cut to encapsulate a selected region of text with a *start-string* and *end-string*. The command is modal and operates in two different operating modes as follows.

First run **encase** without a current region section (i.e. no selected highlighted text) and the user is prompted for the *start-string* and *end-string*. The user is also prompted if the command has not been run before and the *start-string* and *end-string* are undefined.

With the strings defined then select a region with the mouse or keyboard (**esc space** and **move** or **S-move**) to define the start and end point of the region. With a selected region then execute the **encase** command using **esc x encase**. This is most effective when bound to a simple key, see **global-bind-key**(2).

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When set the start and end location is determined by the position in the buffer, the start being first in the buffer. When this bit is clear the start position in always the start of the region regardless of where it is in the buffer. When defining the *start* and *end* strings, i.e. no region is defined or bit **4** is set, the value of this bit is used as a default for subsequent calls when no argument is given.

**0x02** When set the current mark and cursor positions are used as the start and end position rather than the current region, this removes the need for a current region.

**0x04** The *start* and *end* strings are always prompted for when this bit is set.

## NOTES

**encase** is a macro defined in `misc.emf`.

## SEE ALSO

**global-bind-key**(2).

Ref: encase(3) File: m3mac091.3 Date: 2025/08/31

## NAME

execute-tool - Execute a user defined shell tool

## SYNOPSIS

*n* **execute-tool** "*tool-name*"

## DESCRIPTION

**execute-tool** launches a predefined shell tool, the tools are typically defined by the **user-setup**(3) Tools page and executed using the MicroEmacs main Tools menu. See help on **user-setup**(3) for more information on the basic facilities given by execute-tool.

If the numeric argument *n* is supplied it is used as the tool name to be executed, otherwise the argument "*tool-name*" must be given.
A tool with a numeric name can be executed via a key binding, for example, to execute tool **3** (as defined by **user-setup**) to '`C-3`' add the following line to the user setup file:

```
3 global-bind-key execute-tool "C-3"
```

## NOTES

The registry entries for a tool must be located in registry directory "`/history/`**$platform/tool/tool-name**" where **$platform** is the current setting of variable **$platform**(5) and **tool-name** is the name of the tool as given to the command. The following registry entries are used:

**name** The name of the tool as displayed in the user-setup Tools dialog and the Main Tools menu. This is only used for tools `0` to `9`.

**command**
The command-line to be launched when the tool is executed, the following special tokens may be used in the command-line which are substituted at execution:

- **%ff** The current buffer's full file name, including the path.
- **%fp** The current buffer's file path.
- **%fn** The current buffer's file name without the path.
- **%fb** The current buffer's file base name, i.e. the file name without the path or the extension.
- **%fe** The current buffer's file extension with the '.' (e.g. "*.emf*"), set to the empty string if the file name does not have an extension.

Note that "**%ff**" is always the same as "**%fp%fn**" and "**%fp%fb%fe**". If any of these tokens are used, the tool will fail to execute if the current buffer does not have a file name.

**flag** A bit based flag setting the tool characteristics, where:

- **0x01** Enable current buffer saving.
- **0x02** Enable prompt before saving current buffer.
- **0x04** Enable all edited buffers saving.
- **0x08** Enable prompt before saving an edited buffer.

**0x10** Enable output capturing.

**0x20** Enable concurrent running, not available on all platforms, see variable **$system**(5).

**bname** The name of the buffer to be used if the output is captured. The following special tokens may be used in the buffer name which are substituted at execution:

**%fn** The current buffer's file name without the path, set to the buffer name if the current buffer does not have a file name.

**%fb** The current buffer's file base name, i.e. the file name without the path or the extension. Set to the buffer name if the current buffer does not have a file name.

**%fe** The current buffer's file extension with the '.' (e.g. "*.emf*"), set to the empty string if the current buffer does not have a file name or it does not have an extension.

Note that "**%fn**" is always the same as "**%fb%fe**". Default buffer name when this field is left empty is "**command*", or "**icommand*" if `Run Concurrently` is enabled.

If more than 10 tools are required (maximum number definable by **user-setup**) or names are preferred, it is recommended that the **user-setup** dialog is used to define the tool and then use the registry copy utility bound to '`c`' in a **list-registry**(2) buffer.

## SEE ALSO

**user-setup**(3), **ipipe-shell-command**(2), **pipe-shell-command**(2), **shell-command**(2), **system**(5).

## NAME

expand-abbrev-handle - Expand an abbreviation handler

## SYNOPSIS

**expand-abbrev-handle** (**esc esc**)

## DESCRIPTION

**expand-abbrev-handle** pulls together all forms of abbreviation expansion into a single command so that it can be bound to a single key. The abbreviation must be an alpha-numeric string and the cursor must be one position to the right of the abbreviation (which must not be alpha-numeric) when this command is called. The command attempts to expand the abbreviation using the following commands in turn:

**expand-abbrev**(2)

> Uses a buffer specific and global abbreviation files, if set, to look up the abbreviation. The use of the abbreviation file can be disabled using **buffer-setup**(3).
>
> **expand-iso-accents**(3)
>
> Expands ISO accent letter if the expansion mode is enabled via either the **user-setup**(3) General Page or by using the **iso-accents-mode**(3) command.
>
> **expand-look-back**(3)
>
> Looks for a word starting the same in the current buffer's last 100 lines, this can be enabled in the **user-setup**(3) General page.
>
> **Buffer specific expansion**
>
> Executes a buffer specific abbreviation expansion if the current buffer's file hook supports abbreviation expansion.
>
> **Word expansion**
>
> If the current buffer does not support file type specific expansion and Word Expansion is enabled via the **user-setup**(3) General page (`Dict'n` setting) expansion is attempted using the **expand-word**(3) command which expands the current partial word using the dictionary of the user's current language; warning - this can be slow!
>
> The command exits after first command to successfully expand or if none expand the command fails. See the help in the individual expansion commands for more help.

## SEE ALSO

**user-setup**(3), **expand-abbrev**(2), **expand-iso-accents**(3), **expand-look-back**(3), **expand-word**(3).

## NAME

expand-iso-accents - Expand an ISO accent
iso-accents-mode - Enable/disable ISO accent expansion short-cut mode

## SYNOPSIS

**expand-iso-accents**
*n* **iso-accents-mode**

## DESCRIPTION

**expand-iso-accents** provides a facility to enter a plain text representation of an ISO accent and then to expand it into a proper ISO accented character. For example:

`` `a `` => small a, grave accent
`^a` => small a, circumflex accent
`'a` => small a, acute accent
`"a` => small a, umlaut
`a` => small a, tilde
`.a` => small a, ring
`14` => fraction, one-quater
`12` => fraction, one-half
`34` => fraction, three-quaters
`ae` => ae ligature
`sz` => small sz ligature, German.
`+-` => plus or minus (math.)
`co` => copyright
`rg` => registered trademark
`tm` => trade mark
`oe` => small oe ligature
`/o` => small o, slash

The **expand-iso-accents** can be called directly to expands the ISO abbreviated character sequence into it's ISO ASCII character equivalent. The command looks at the 2 characters to the left of the cursor and tries to find a matching abbreviation, if found the 2 characters are removed, replaced by the single ISO character.

The more typical way of using this feature is by enabling its use in the abbreviation handler which is usually bound to "`esc esc`". It can be enabled by either by using the **iso-accents-mode** command or, for a more permanent installation, from the **user-setup**(3) => General => Abbrev Expansion settings.

When using the **iso-accents-mode** command, if a numeric argument *n* is given (the value is not used) then the ISO accent expansion is installed locally into the current buffer. If *n* is omitted then expansion is enabled/disabled globally (across all buffers).

**expand-iso-accents** is the macro command that This is by default bound to . If an ISO character is not located then **expand-abbrev**(2) is invoked to try a standard abbreviation.

## NOTES

**iso-accents-mode** and **expand-iso-accents** are implemented as macros in the file `abbrev.emf`, the repertoire of expansions may be enhanced by editing this file.

Unlike the general **expand-abbrev**(2) command which attempts to expand the current word, **expand-iso-accents** only considers the last two characters regardless of whether they are word characters or start a word. Therefore the general **expand-abbrev** command cannot be used to implement a similar feature.

## SEE ALSO

**expand-abbrev-handle**(3), **buffer-abbrev-file**(2), **expand-abbrev**(2), **expand-look-back**(3), **expand-word**(3).

Ref: expand-iso-accents(3) File: m3mac045.3 Date: 2022/06/12

## NAME

expand-look-back - Complete a word by looking back for a similar word

## SYNOPSIS

**expand-look-back**

## DESCRIPTION

**expand-look-back** attempts to complete the word at the current position by looking backward for another word which starts the same. If such a word is found within 100 lines of the current cursor position the current partial word is replaced with the word found.

**expand-look-back** is automatically invoked from the **expand-abbrev-handle**(3) macro in response to an expansion command, it is only invoked if enabled in the **user-setup**(3) => General => Abbrev Expansion => Lookbk setting is enabled.

## NOTES

**expand-look-back** is a macro implemented in `abbrev.emf`.
The **user-setup** configuration simply sets the macro variable `.expand-look-back.on` to TRUE, i.e.:

```
set-variable .expand-look-back.on 1
```

It may be subsequently disabled by setting the variable back to 0.

## SEE ALSO

**expand-abbrev-handle**(3), **user-setup**(3).

Ref: expand-look-back(3) File: m3mac062.3 Date: 2022/06/12

## NAME

expand-word - Complete a word by invocation of the speller

## SYNOPSIS

**expand-word**

## DESCRIPTION

**expand-word** attempts to complete the word at the current position through the use of the current language dictionary. The user is presented with a list of endings for the given word portion. These may be selected with the cursor or mouse.

**expand-word** is automatically invoked from the **expand-abbrev-handle**(3) macro in response to an expansion command, it is only invoked if enabled in the **user-setup**(3) => General => Abbrev Expansion => Dict'n setting is enabled.

## NOTES

**expand-word** is a macro implemented in `abbrev.emf`.
The **user-setup** configuration simply sets the macro variable `.expand-word.on` to TRUE, i.e.:

```
set-variable .expand-word.on 1
```

It may be subsequently disabled by setting the variable back to 0.

## SEE ALSO

**expand-abbrev-handle**(3), **spell-buffer**(3), **find-word**(3).

Ref: expand-word(3) File: m3mac048.3 Date: 2022/06/12

## NAME

file-browser - Browse the file system file-browser-close - Close the file-browser file-browser-swap-buffers - Swap between file-browser windows

## SYNOPSIS

*n* **file-browser** (**f10**)
**file-browser-close**
**file-browser-swap-buffers**

## DESCRIPTION

**file-browser** can be used to browse around the file system. When first executed **file-browser** creates 2 buffers, "`*directory*`" displaying the directory structure and "`*files*`" listing the files in the current directory with information on each file. **file-browser** displays these buffers side by side, splitting the current window horizontally if required.

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When not set **file-browser** only creates a single "`*files*`" window.

**0x02** When set **file-browser** will attempt to identify if the current directory is being managed by a source code revision system (such as **cvs**(3) or **svn**(3)) and if so modifies the directory listing to include addition revision information. Note that the control system's support macros must already be loaded, this can typically be done by enabling the relevant 'support on start-up' option.

Once open the user can browse through the system using the following keys in the `*directory*` buffer, use **esc-h** for help within the buffer:

`space` Selects the directory on the current line and up-dates the `*files*` buffer with the information on this directory. This can also be done by clicking the left mouse button on the directory name.

`return`
Selects the directory on the current line, if open (sub-directories displayed) then closes it or if closed it is opened. The `*files*` buffer is up-dated with the information on the directory. This can also be done by clicking the left mouse button on the '+' or '−' symbol just before the directory name.

`C-return`
As with `return` expect sub-directories are recursively opened or closed, note that this could take some time on large file systems. This can also be done by clicking the right mouse button on the '+' or '−' symbol just before the directory name.

`tab` Move to the `*files*` buffer.

`f10` Closes file-browser.

The following keys can be used in the `*files*` buffer:

`return`
If the current line is a directory, this because the current directory, updating both the `*directory*` and `*files*` buffers. If the line is a file then it is opened using **find-file**(2). This can also be done by clicking the left mouse button on the file name.

Where a **file-tool** open action has been defined using **file-tool-setup**(3) then the selected tool or utility is run to process the file. This may be used to launch viewers for PDF, image files etc.

`space` Toggles the tag state of the file on the current line, see `x` command. This can also be done by clicking the left mouse button anywhere before the file name, or for multiple files drag a region with the left mouse button.

`X` or `x` Executes a **shell-command**(2) on all tagged files. The user is prompted for the command line which can contain the following special tokens:

`%p` Full file name, including path.
`%f` The file name without the path.

As the **shell-command** is executed in the directory `%f` is safe to use in a command such as "`del %f`".

`D` or `d` or `delete`
Delete selected files or current file/directory.

`tab` Move to the `*directory*` buffer.

`backspace`
Move to the parent directory.

`l` Refresh the listing from the file system.

`f` Toggle the filter, enables/disables the display of back-up, object files etc.

`S` Initiates a search for files, at the command prompt then enter the leading characters to search for use **C-s** and **C-r** to search forwards or backwards respectively. The `s` version of this command is an incremental search and is easier to use.

`s` Launch an incremental search by name, at the prompt enter the search characters, use **C-s** and **C-r** to search forwards or backwards respectively.

`1` Sort list by selected state.

`2` Sort list by attribute.

`3` Sort list by size.

`4` Sort list by modified time.

`5` Sort list by name.

`6` Sort list by extension.

`space` or `+`
Select file.

`-` Unselect file.

`*` Invert selection.

`a` Select all.

`u` Unselect all.

`e` Edit file, opens the file in the editor for modification.

`i` Edit selected file, opens all of the selected file in the editor for modification.

`v` View file, opens file for viewing only (write protected).

`r` Rename file or directory, command line prompt for the new name.

`c` Copy file(s) and/or directory(s), command line prompt for directory to copy file(s).

`m` Move file(s) and/or directory(s), command line prompt for directory to move file(s).

`n` Create a new directory (or folder), command line prompt for the name of the new directory to create.

`t` Invokes the file tools menu to process the file, see **file-tool-setup**(3).

**file-browser-swap-buffers** swaps between the `*directory*` and `*file*` windows, making the other the current window, this is usually locally bound to the `tab` key.

**file-browser-close** hides both the `*directory*` and `*file*` windows, closing the file-browser, this is usually locally bound to the `f10` key. Note that this command is hidden and will not complete on the command line, but may be bound or input explicitly.

## HISTORY

The command line mapping was revised in the 2006 release for a more logical key to file operation mapping.

## SEE ALSO

**directory-tree**(2), **file-tool-setup**(3), **find-file**(2), **find-zfile**(3), **shell-command**(2), **cc-setup**(3), **cvs-setup**(3), **svn-setup**(3).

Ref: file-browser(3) File: m3mac027.3 Date: 2023/06/29

## NAME

file-exec - Executes the current buffer using a compiler or interpreter
file-format - Format the current buffer using a dedicated formatter
file-lint - check the current buffer for problems using a dedicated linter
major-mode-tools-setup - File major-mode tool setup

## SYNOPSIS

**file-exec** [ "*command-line-arguments*" ]
**file-format**
**file-lint**
**major-mode-tools-setup**

## DESCRIPTION

The **file-exec** macro for interpreted languages executes the current buffer using the command line interpreter defined for this language. In case of programming languages which needs to be compiled, first the code is compiled to machine code and then executed. Before the program is executed **save-some-buffers**(2) is executed to allow the user to ensure that all relevant buffers are saved. If bit `0x01` of the numeric argument *n* is clear then all modified buffers are automatically saved, when set (the default) the user is prompted for each buffer. If bit 0x02 of the numeric argument is clear (the default) the user is prompted for any additional *command-line-arguments* which are passed to application or the script command. The arguments given are stored in a buffer variable, and used as the default arguments next time the script is executed.

After executing the program, an `*FHOOK-exec*` buffer is opened with the output of the script or program. The script executes the current buffer if required using the current MicroEmacs character encodings.

The **file-format** macro formats the current buffer using a dedicated formatting tool for the buffer's programming language. For R for example, the package 'formatR' must be pre-installed by the user, this is typically done by running 'install.packages('formatR')' within an R console.

The **file-lint** macro checks the currently edited file using a language specific linter, for example for R, the package 'lintr' must be installed by the user for instance using the usual 'install.packages('lintr)' syntax from within a R console.

The command opens an `*FHOOK-lint*` buffer with the output from the linter, warnings or styling issues raised are typically high-lighted. the buffer is configured to for **get-next-line**(2) to make traverse through the issues as easy as possible.

## SETUP

The **major-mode-tools-setup** command can be used to configure the three tools for the current file's major mode, this dialog can be accessed by clicking on the bottom left **Tools** button in the **major-mode-setup**(3) dialog. Note, however, that as the tool configurations should work with all files of the same major-mode, the command-lines should use tags (`%`*l*...) rather than literal file names. Of particular importance are the **%f**, **%p** and **%"** tags, for further details see the documentation for the **Command** field of **file-tool-setup**(3), which uses the same format.

The **exec-file** command has 3 additional configuration checkboxes; when enabled, **Raw output** suppresses the 3 line header, which details the location and command-line being run, and stops

long lines in the command output from being wrapped. **No line wrap** also stops the wrapping of long lines, but does not suppress the header. When enabled, **Supports buffers** indicates that, if the current buffer does not have a file name, the **exec-file** command can save the buffer contents to a temporary file and then run the given utility with the temp file. This typically requires the execution process to not be dependent on any other local content. Note that the generated temp file is removed 20 seconds after the start of the execution so this option should not be used if the execution requires the file to be present for longer.

For creators of MicroEmacs file hooks, the file tools can also support the use of multiple interpreters/tools, e.g. `clang` and `gcc` to compile a C file. The configuration of each tool can include tests to verify the availability of the tool on the current computer as well as the required command-line to run.
To get the **file-exec**, **file-format** and **file-lint** commands properly working, configuration variables are typically required for the current buffer's major-mode. For example, the following configuration is taken from hkpython.emf:

```
set-variable .fhook-python.coms "\bpython3\bpython2\bpython\b"
set-variable .fhook-python.coms-windows "\b~/AppData/Local/Programs/Python/Python\\d+\\(-\\d+\\)?/python
set-variable .fhook-python.exec1   "\b%v[com]\b\b\b\b%\"%v[com]%\" \"%p%f\" %v[args|20|Additional commar
set-variable .fhook-python.format1 "\bruff\bruff -h\b0\b\bruff format %\"%f%\"\bRuff\b"
set-variable .fhook-python.format2 "\b%v[com]\b%\"%v[com]%\" -m yapf -h\b0\b\b%\"%v[com]%\" -m yapf -i %
set-variable .fhook-python.format3 "\b%v[com]\b%\"%v[com]%\" -m black -h\b0\b\b%\"%v[com]%\" -m black -q
set-variable .fhook-python.lint1   "\bruff\bruff -h\b0\b\bruff check %\"%f%\"\bRuff\b"
set-variable .fhook-python.lint2   "\b%v[com]\b%\"%v[com]%\" -m pylint -h\b0\b\b%\"%v[com]%\" -m pylint
```

Where the `.exec#`, `.format#` and `.lint#` variables configure a tool to use the specified utility, python hhas been configured to check for and potentially use 3 different utilities for formatting, **file-format** will use the first utility found. The format of these variables is as follows:

```
|<program>|<test>|<test-success-exit-code>|<path>|<command-line>|<ID>|<err-regex>|<wrn-regex>|<inf-regex
```

Only the `<program>` and `<command-line>` parameters are required, `<test>` and `<test-success-ex` parameters should only be used when the mere presence of the `<program>` is not a guarantee that the utility is fully installed and correctly running. For example, **Yapf** is a python module that must be installed in addition to python itself, to the presence of a python script engine, i.e. the **python3** program, is no guarantee that the yapf module has also been installed. Instead the command "<python> -m yapf -h" and the exit code is tested, **0** being a success.

## NOTES

**file-exec**, **file-format** and **file-lint** are macros defined in `hktools.emf`. For programming lanuages like C++, Fortran and R there are already useful variable defaults defined in the hkHOOK.emf files, the user can overwrite these settings using **major-mode-tools-setup**.

The executables or the interpreters will need to be installed in the system path for the commands to function correctly, otherwise the full path will need to be given in the settings.
When setting up tools it can be useful to add "`%v[<tool>-opts|]`" in places where standard options, such as user name or server, may need to be supplied. Users can then define a **.fhook-<major-mode>.<tools>-opts** variable in their `my<major-mode>.emf` file so they do not have to type these values every time they restart MicroEmacs. For example, to execute an SQL statement in MySql the base server, access and database details are required, rather than require the user to supply this every time the main command can be defined as:

```
mysql %v[exec-opts|] %v[exec-args|20|Additional command-line arguments|--host=localhost --user=<user> --
```

The user can then add the following line to their `mysql.emf` file:

```
set-variable .fhook-sql.exec-opts "--host=localhost --user=JDOE --password=12345"
```

So the base DB access is pre-defined and only the database name is required, a better default can also be supplied by adding:

```
set-variable .fhook-sql.exec-args "MYDB"
```

The user will still be prompted, but the default is now likely to be correct. The user can avoid being prompted by defining the variable `var-name`**-np** ('np' = no-prompt), i.e. adding:

```
set-variable .fhook-sql.exec-args-np "MYDB"
```

Would define the database name and stop the prompting.

## SEE ALSO

**get-next-line**(2), **file-tool-setup**(3), **major-mode-setup**(3) **c**(9), **cpp**(9), **f90**(9), **go**(9), **pas**(9), **perl**(9), **py**(9), **r**(9), **tcl**(9).

Ref: file-exec(3) File: m3mac099.3 Date: 2026/03/24

## NAME

file-list - Generate list of matching file names

## SYNOPSIS

**file-list** "*file-path-regex*"

## DESCRIPTION

**file-list** is a macro language construct only that performs a recursive search for all files and directories that match the given *file-path-regex*. Unlike **$file-names**(5), the *file-path-regex* argument can have a regex at any point in the path, enabling the command to locate matching files in multiple locations.

The one restriction is that regex classes and special characters cannot and will not include the path separator character, i.e. giving the command a mask of "`/foo/[a-z/]*/bar`" to find any `bar` within `/foo` is not supported and the '`.*`' in "`/foo/.*/bar`" will only match files and directories in `/foo`. The only exception to this rule is for a trailing class which can contain a path character, e.g. the mask can end with "`[^/]`" to filter out any directories from the result.

On completion, the macro sets the command variable **.file-list.file-count** to the number of matches found, and the variables **.file-list.file0**, **.file-list.file1**, ..., **.file-list.file***n* to each match, where *n* is the number of matches minus one, i.e. **.file-list.file-count**`-1`.

## EXAMPLE

The following example can be used to locate the **Rscript**(1) interpreter on Windows, which is typically installed into `C:\Program Files\R\R-<version>\bin\` folder:

```
file-list "C:/Program Files/R/R-\\d+\\.\\d+\\.\\d.*/bin/Rscript.exe"
!if &not .file-list.file-count
  ml-write "ERROR: Failed to locate the Rscript interpreter"
  !abort
!endif
set-variable #l0 .file-list.file0
!iif &gre .file-count 1  ml-write &cat "Warning: Multiple Rscript interpreters found, using " #l0
```

## NOTES

The **file-list** command is not publically available from the command line, but may be used within macros to perform the copy.

This command is implemented as a recursive macro in `utils.emf`.

## SEE ALSO

**str-to-regex**(3), **filemask-to-regex**(3), **$file-names**(5), **&xsequal**(4).
Regular Expressions

Ref: file-list(3) File: m3mac108.3 Date: 2025/12/01

## NAME

find - Execute find command

## SYNOPSIS

**find** ["*find-options*"] "*file-mask*" ["*file-mask*"...] ["*starting-path*"]
**find-setup**

## DESCRIPTION

**find** performs a search for files in the current directory (or from *starting-path*) using built in MicroEmacs constructs rather than relying on any external executables such as **find**(1). **find** operates in exactly the same way as a traditional *find* command line utility and uses a command line option sequence from the MicroEmacs command line. The *file-options* allow recursive and regular expression searches, searching against the file modification time etc. The command line takes the user supplied *file-options*, *file-mask* and *starting-path*. The output of the command is piped into the ***find*** buffer.

*find-options*

Defines the *options* that are specified on the command line, the default options may also be defined with the **file-setup**(3) command invoked from the command line or from the the **Tools** menu. The *find-options* are defined as follows:

**–** Last option, the following string must be the first *file-mask* which may start with a '-'.

**-2** List the file and directory attributes in the output (directory list sort key)

**-3** List file and directory size in the output.

**-4** List file and directory modified time in the output.

**-a** Re-read all out of date buffers, prevents a manual prompt, equivalent of Yes to all. This is the inverse of **-o**

**-b** Ignore binary files.

**-B** Do not ignore binary files.

**-c** Command return the number of matchs found/replaced rather than the default, which is simply 0 for 1 or more matches or 1 if no matches were found/replaced.

**-C** Command returns 0 (non-error for command-line tool) regardless of if it finds any matches.

**-d** List directories or grep in directory listings.

**-D** Do not list directories or grep in directory listings.

**-f** List/search files.

**-F** Do not list/search files.

**-g** File masks are regex (i.e. .*.emf)

**-G** File masks are not regex (i.e. *.emf)

**-i** Ignore case (case insensitive) when searching for the **-w** <*search-pattern*>.

**-I** Do not ignore case (case sensitive) when searching for the **-w** <*search-pattern*>.

**-l** Use existing buffer of loaded files.

**-L** Do not use existing buffer of loaded files.

**-m** Search pattern is a regular expression.

**-M** Search pattern is not a regular expression.

**-o** Do not re-read all out of date buffers, prevents a manual prompt, equivalent of No to all. This is the inverse of **-a**.

**-p** Starting path specified.

**-P** Starting path not specified.

**-q** Enables quiet mode for reduced print-out.

**-r** Recurse into sub-directories

**-R** Do not recurse into sub-directories

**-S** <*size*>
File size - less than or equal, the following size formats are supported:

**-S** <*number*> File size specified bytes. **-S** <*number*>**k** File size specified in Kilobytes.

**-s** <*size*>
File size greater than; supported size formats are the same as **-S**.

**-T** <*time*>
Modification time - older than, the following time formats are supported:

**-T** *yy-mm-dd* **-T** *yyyy-mm-dd* **-T** *yyyy-mm-dd hh:mm:ss* **-T** *days* Offset time in days from midnight this morning, e.g. **-T 0** would match all files not modified today. **-T** *days.hours[.minutes[.seconds]]* Offset time in , e.g. **-T 1.1.30** matches all files 25 hours and 30 minutes old.

Note that **-T 1** is not the same as **-T 1.0**, except at midnight.

**-t** <*time*>
Modification time - newer than or equal, supported time formats are the same as **-T**.

**-u** When used with **-w**, allows the file size limit of 100MB to be disabled and strings are searched in files of any size.

**-U** When used with **-U**, files are limited in size 100MB otherwise they will not be searched.

**-w** <*search-pattern*>
File must contain a string matching the given *search-pattern*.

**-W** <*search-pattern*>
File must not contain a string matching the given *search-pattern*.

**-x** Use MicroEmacs **$file-ignore**(5) to exclude unimportant files

**-X** Do not use MicroEmacs **$file-ignore**(5).

**-&** Launch find using a separate process, this stops the current MicroEmacs hanging during the find process which can be useful for time consuming searches.

*file-mask*
: The files to search for e.g. ***.txt**

*starting-path*
: The directory location where the search should commence. The default is the location of the current buffer.

  **find** sets **$result**(5) to the return of the command, which is `0` if matches were found or 1 of no match was found, unless options **-c** or **-C** are used.

  **file-setup** invokes a dialogue and allows the default *file-options* to be specified when the **find** command is invoked.

## NOTES

- Directories are not considered binary files, so **-b** will not exclude them.
- The default when no options are given will be **-XGPrdfMIBL**, default options can be set via the find-setup command (sets the **/history/find-com** registry entry).
- Use of **-w** on directories searches for <*search-pattern*> within the MicroEmacs directory listing, the command is then the same as **grep**(3) except for the output buffer format.
- String searches of **-w** are not performed on files that exceed 100MB unless the **-u** option is specified on the command line. This prevents MicroEmacs from loading exceptionally large files unless explicitly required.

## SEE ALSO

**%xgrep-com**(5), **find**(1), **grep**(3), **get-next-line**(2), **grep**(1), **xfind**(3), **xgrep**(3), **rgrep**(3).

Ref: find(3) File: m3mac088.3 Date: 2026/01/22

## NAME

find-bfile - Load a file as binary data
find-rfile - Load a file as binary data (reduced)

## SYNOPSIS

*n* **find-bfile** "*file-name*" (**C-x 9**)
*n* **find-rfile** "*file-name*"

## DESCRIPTION

**find-bfile** and **find-rfile** both load a file as binary data, providing binary editing capabilities. The commands operate as command **find-file**(2) except that they load the given file with **binary**(2m) or **rbin**(2m) mode set.

**find-bfile**, which uses binary mode, loads any file and converts every 16 bytes into a line of text giving the hex address of the current position in the file, the bytes as hexadecimal numbers and all printable characters at the end of the line (all non-printable characters are displayed as a '.'). However, This format makes it memory hungry in that every 16 bytes of the file requires a 78 character line (approximately 5 times more memory!).

**find-rfile**, which uses rbin mode, loads any file and converts every 256 bytes into a line of text containing only 2 hexadecimal numbers per byte leading to a line length of 512 characters. This format greatly reduces the memory required (approximately 2 times more memory) making it ideal for large binary file editing.

The extension and content of a file loaded using binary or rbin mode is ignored with respect to major-mode assignment (See **major-mode**(2)). Instead the **binary** or **rbin** major-mode is always applied if they exists.

Each line of text may be edited, providing that the basic format of every line remains the same, the altered file may be written out, providing full binary editing capabilities.

## EXAMPLE

Given a single line MSDOS file:

```
Live long and prosper.
```

When loaded in using **find-bfile** will produce the following 2 line buffer:

```
00000000: 4C 69 76 65 20 6C 6F 6E 67 20 61 6E 64 20 70 72  |  Live long and pr
00000010: 6F 73 70 65 72 2E 0D 0A 1A                       |  osper....
```

Note the "0D 0A 1A" at the end, this is due to MSDOS's "\n\r" carriage returns and ^Z file termination, these characters are unprintable and are shown as '.' at the end of the line. The basic form of a binary file is:

```
AAAAAAAA: XX XX XX XX ....  |
```

When saving a binary file, only the text between the ':' and '|' is considered and it may contain any number of hexadecimal numbers. The given file could be made UNIX compatible by editing the buffer to:

```
00000000: 4C 69 76 65 20 6C 6F 6E 67 20 61 6E 64 20 70 72  |  Live long and pr
00000010: 6F 73 70 65 72 2E 0D                             |  osper....
```

If the word "**long**" is required to be removed, the following would not work:

```
00000000: 4C 69 76 65 20 6C 6F 6E 67 20 61 6E 64 20 70 72  |  Live and pr
00000010: 6F 73 70 65 72 2E 0D 0A 1A                       |  osper....
```

The ASCII end letters are ignored, instead the following operation must be performed which removes the characters from the binary representation:

```
00000000: 4C 69 76 65 20 61 6E 64 20 70 72  |  Live long and pr
00000010: 6F 73 70 65 72 2E 0D 0A 1A                    |  osper....
```

One could be nasty by doing the following:

```
00000000: 4C 69 76 65 20 73 68 6F 72 74 20 61 6E 64 20 |
00000010: 64 6F 6E 27 74 20 70 72 6F 73 70 65 72 2E 0D 0A 1A           |
```

("Live short and don't prosper"). If this file was then loaded using **find-rfile**, the following single line buffer would be created:

```
4C6976652073686F727420616E6420646F6E27742070726F737065722E0D0A1A
```

## SEE ALSO

**find-hfile**(3), **find-lfile**(3), **find-file**(2), **binary**(2m), **rbin**(2m).

Ref: find-bfile(3) File: m3mac095.3 Date: 2026/02/25

## NAME

find-hfile - Load a file using selected handler
find-cfile - Load an encrypted file
find-lfile - Load a file lossless
find-cfile - Load a Unicode file

## SYNOPSIS

*n* **find-hfile** "*h*" "*file-name*" (**C-x C-h**)
*n* **find-cfile** "*file-name*"
*n* **find-lfile** "*file-name*"
*n* **find-ufile** "*file-name*"

## DESCRIPTION

**find-hfile** first prompts the user for the read handler *h* required to load the file into MicroEmacs, where *h* can be:

**C** As with **c** below except that the user is prompted for a second handler which is combined with crypt. Supported secondary handlers are **b**, **l**, **r**, **t** and **u**. When used this handler must be given first.

**b** Standard binary file handler, see **binary**(2m) mode for more information.

**c** Encrypted file handler, see **crypt**(2m) mode for more information.

**l** Lossless text file handler, allows loading of any file, including binary files, as if a normal text file while preserving every byte. This allows easy searching for text strings within any file, but care needs to be taken if editing as binary files are easily corrupted. See command **translate-buffer**(2) for more information on the lossless text format.

**r** Reduced binary file handler, see **rbin**(2m) mode for more information.

**t** Normal text file, equivalent to standard use of **find-file**(2).

**u** Unicode text file handler, see **find-ufile** below for more information.

**z** Zip file handler, see **find-zfile**(3) for more information. This handler cannot currently be combined with **C**.

The macro then calls the appropriate underlying file handler command with the required numerical argument to load *file-name* correctly.

**find-cfile** provides a simple interface to loading files in **crypt**(2m) mode. The numeric argument has the same effect as with the **find-file**(2) command except the **crypt** mode is always enabled. See documentation on the mode and **find-file** command for more information.

The **find-lfile** macro uses **find-file**(2) to load the given file in **rbin**(2m) mode, then calls **translate-buffer**(2) with a numerical argument of 0 to convert the buffer to text without loss of any character. When using this mode care should be taken not to change the **cr**(2m), **ctrlz**(2m) or **lf**(2m) modes as these will affect the output.

**find-ufile** allows loading of any UTF-8, UTF-16 or UTF-32 encoded Unicode file as if a normal text file, while preserving every character (except those in the last

`0x100000` Unicode plane). This allows for easy reading and editing of all languages supported by the current charset of MicroEmacs. Care must be taken to avoid corrupting the byte sequences of characters not supported by the current charset, see command **translate-buffer**(2) for more information on the Unicode preserving text format used.

## NOTES

**find-hfile**, **find-cfile**, **find-lfile** and **find-ufile** are macros defined in file `tools.emf`.

Buffers containing directory listings generated by **find-file** and zip content listings created by **find-zfile** provide a similar interface to **find-hfile** bound to the letter **h**.

The command **find-file**(2) is bound to key "`C-x 9`" with a numeric argument of 3, this is equivalent to executing **find-bfile** with no argument.

## SEE ALSO

**find-bfile**(3), **find-rfile**(3), **find-zfile**(3), **find-file**(2), **translate-buffer**(2), **binary**(2m), **crypt**(2m), **rbin**(2m).

Ref: find-hfile(3) File: m3mac030.3 Date: 2026/02/25

## NAME

find-zfile - Compressed file support
zfile-setup - Compressed file support setup
zfile-reset - Compressed file support reset

## SYNOPSIS

**find-zfile** "*file-name*"
**zfile-setup** "*zfile-id*" "*extension-regex*" "*list-command*"

"*top-cut-to-regex*" "*bottom-cut-to-regex*"
[ "*body-search-regex1*" "*body-replace-regex1*" "*body-search-regex2*"
"*body-replace-regex2*" ... ]
"*change-slashes*" "*bottom-summary-search-regex*" "*bottom-summary-replace-regex*"
"*list-header*" "*extract-command*"

**zfile-setup** "*zfile-id*" "*extension-regex*" "" "*piped-extract-command*"
**zfile-reset**

## DESCRIPTION

**find-zfile** provides generic support for listing and extracting the contents of compressed files, **find-zfile** also supports the extraction and loading of the internal files into a buffer.

**find-zfile** must be configured for each compression utility using **zfile-setup**. It relies on command-line programs to generate content lists which are used to generate the main file listing, and subsequently, the ability to extract individual files for file extraction support.

For basic content listing support the first 3 arguments must be given to zfile-setup. The first argument is a tool ID, if an existing tool's ID is entered **zfile-setup** will reconfigure the tool, otherwise a new tool will be created. The second argument, "*extension-regex*" is used to match a file to this utility, the file's full name is compared with ".*\.<*extension-regex*>" and if matched the tool will be used. Comparisons are always case insensitive for improved cross-platform behaviour.

The compressed file contents list is generated from executing the user supplied "*list-command*" and dumping the output into the list buffer. The command is run from the directory containing the compressed file and the following special tags may be used within the "*list-command*" which get substituted as follows:

**%%** The token is replaced with a single '%'.

**%"**<str>**%"**
The **%"** tokens are replaced with '"'s if <str> contains a space, if it doesn't they are simply removed.

**%zb** The token is replaced with the compressed file's base name, i.e. the file name without the path or the extension.

**%ze** The token is replaced with the compressed file's extension, i.e. the string matched by the *extension-regex*, note this can be multiple extensions if, for example, *extension-regex* match 'tar.gz'.

**%zf** The token is replaced with the compressed file's file name, i.e. the file name without the path but with the extension.

**%zp** The token is replaced with the compressed file's absolute path, including a trailing '/'. All directories will be separated by a '/' character. extension.

**%zP** The token is replaced with the compressed file's absolute path, including a trailing '/'. However, unlike **%zp** the character used to separate directories will be the OS preferred character, see index 0 of **%os-chars**(5).

The head of the list output is often unwanted verbose printout, this can be automatically be removed by the use of the "*top-cut-to-regex*" argument. The argument, if supplied (not an empty string), must be a regex search string matching the end of the text to be removed. If found, all text from the end of the match and before is removed.

To improve usability, the list typically needs reformatting so that items can be selected (first column must be a space) and the list can be sorted on the main column headings. This is achieved by supplying *body* regex-based search and replace strings, as many can be supplied as required. Note that the result should ensure the information given for each item aligns with the column heading otherwise the sort features are unlike to work. Of particular important is the item name, if this is miss-aligned the item extraction is unlikely to work.

The bottom of the output often has summary information, such as the total number of files a size etc. MicroEmacs puts the summary information for directory listing on the second line, and for consistency zfile does too. This can be best achieved by supplying the *bottom-summary* regex-based search & replace strings, if the search matched the replace string is inserted into the second line of the buffer. If no summary search is supplied zfile creates a basic file and directory count summary.

For extraction to work, the given *list-header* line must include the string '**Name**' and this sting must be aligned to the start of each item's file or directory name. For copying, viewing or editing files, the same key bindings used in directory listing are also supported by zfile.

The selected files are then extracted by executing the supplied "*extract-command*" to the selected destination, if copying, or to a temporary directory which is automatically removed later. As with the *list-command*, the extract command is run from the directory containing the compressed file and the same **%z?** tokens can be used. The command line must contain a `%p` or `%P` token to specify the output location, it can also use the %% & %" tokens, %f for the item name and **%*[...]** for multiple file support, see help on **file-tool-setup**(3) for more information on this last token.

When first used, if there are currently no compression utilities configured MicroEmacs will test for the availability of **7zip**(1), **tar**(1) and **gunzip**(1), and automatically configure the tool appropriately. If these tools, particularly **7zip** are available then use of **zfile-setup** may not be necessary. However, the system setup or the default configuration of MicroEmacs may change, in which case **zfile-reset** should be run, this will remove any existing zfile configuration for the current platform, including any manual configuration, and regenerate the bast default setup available.

## EXAMPLE

For zip file support the freely available **unzip**(1) command can be used, running "`unzip -l /tmp/Jasspa_MicroEmacs_macros.zip`" produces an output like the following:

```
Archive:  Jasspa_MicroEmacs_macros.zip
  Length      Date    Time    Name
 --------    ----    ----    ----
        0  18-09-24 01:18   macros/
     3552  18-09-24 01:18   macros/dmf.emf
     1413  18-09-24 01:18   macros/langutl.emf
    22193  18-09-24 01:18   macros/hkcpp.emf
        .
        .
        .
     2314  18-09-24 01:18   macros/hkrd.emf
    18910  18-09-24 01:18   COPYING.txt
     4759  18-09-24 01:18   readme.txt
 --------                   -------
  3461536                   364 files
```

The following configuration can be used:

```
zfile-id=zip
extension-regex=\(zip\|epub\|apk\)
list-command=unzip -l %zf
top-cut-to-regex=---\n
bottom-cut-to-regex=^ ---
body-search-regex1=        0  \([-\d: ]\{14\}\)   \(.*/\)$
body-replace-regex1= d...          0  \1 \2
body-search-regex2=^\([ \d]\{9\}\)  \([-\d: ]\{14\}\)   \(.*\)$
body-replace-regex2= ....  \1  \2 \3
change-slashes=0
bottom-summary-search-regex=^ *\(\d+\) +\(\d+ files\)
bottom-summary-replace-regex=    \1 used in \2
list-header= Attr     Length        Modified Name
extract-command=unzip -o %zf %*[ %"%f%"] -d %"%p%"
```

With this setup, running find-zfile on this zip file will produce the following listing:

```
ZFile-zip: /tmp/Jasspa_MicroEmacs_macros.zip
    3461536 used in 364 files

 Attr     Length        Modified Name
 d...          0  18-09-24 01:18 macros/
 ....       3552  18-09-24 01:18 macros/dmf.emf
 ....       1413  18-09-24 01:18 macros/langutl.emf
 ....      22193  18-09-24 01:18 macros/hkcpp.emf
    .
    .
    .
 ....       2314  18-09-24 01:18 macros/hkrd.emf
 ....      18910  18-09-24 01:18 COPYING.txt
 ....       4759  18-09-24 01:18 readme.txt
```

The top 3 lines are removed as they are not required due to the reformatting requiring a new header line. Each directory line is reformatted first and flags with a 'd' Attribute, the remaining file lines are then reformatted. The bottom summary information is matched with the search regex so the useful information it contains can be pulled out and inserted onto the second line.

As the names of all items now align with the 'Name' string in the new header, the extraction process will get the correct names. Note that files search as `dmf.emf` will be extracted into a `macros` sub-directory.

## NOTES

**find-zfile**, **zfile-setup**, and **zfile-reset** are macros defined in `zfile.emf`.

The order in which the tools are defined can be important and priority is given to the last tool defined. So if the tool currently being used to support a compressed file type could be improved, another tool can be added for that extension and will be used in preference.

## SEE ALSO

**find-file**(2), **find-hfile**(3).

Ref: find-zfile(3) File: m3mac040.3 Date: 2026/01/30

## NAME

fold-create - Create a new fold (**C-f2**)
fold-close - Close an open fold (in-line) (**C-f2**)
fold-enter - Enter fold (**f3**)
fold-exit - Exit a fold (**esc f3**)
fold-exit-all - Exit all folds (**esc C-f3**)
fold-kill - Delete the fold and all its contents (**C-x C-k**)
fold-open - Open fold (in-line) (**f2**)
fold-remove - Remove the fold, maintaining content (**esc C-f3**)

## SYNOPSIS

**fold-create** (**C-f2**)
**fold-close** (**C-f2**)
**fold-enter** (**f3**)
**fold-exit** (**esc f3**)
**fold-exit-all** (**esc C-f3**)
**fold-kill** (**C-x C-k**)
**fold-open** (**f2**)
**fold-remove** (**esc C-f3**)

## INTRODUCTION

MicroEmacs supports folding editor type features that allow folds to be created, deleted and traversed. Folded regions of the file may be entered and edited whilst hiding information in other folds. Folds may also be opened in-line such that more than one fold is visible at any time.

The **fold** operation has different semantics than collapse which simply hides information with a single narrow and does not provide features such as entering the fold.

## DESCRIPTION

**fold-create** creates a new fold from the text defined between the cursor and the set-mark position. The text between the mark line and the cursor line inclusive are then folded and fold markers are inserted into the buffer. The folded line may be commented with a fold comment. The fold may be subsequently removed with **fold-remove** which removes the fold under the cursor, deleting any fold annotations.

**fold-open** opens the fold under the cursor, the fold is expanded in line such that the contents of the fold are visible. The fold may be subsequently closed using **fold-close** when the body of the text is hidden underneath a fold marker.

**fold-enter** enters a fold under the cursor and the contents of the fold are reveled the rest of the buffer is narrowed out. The previous fold level may be restored with **fold-exit**. Where the fold nesting is deep then **fold-exit-all** may be used to return to the top level.

**fold-kill** deletes the fold under the cursor and all of the information contained within the fold. **undo**(2) will restore the fold if this was not the required action.

## DEFINITION

Any file may be folded provided that the opening and closing fold strings are defined. Folding may be defined and initialized for a file type using **buffer-init**(3) by defining the **fhopok-XXX.fold** variable to define the opening and closing fold markers. i.e.

```
set-variable .fhook-XXX.fold "<open> <close>"
```

Where <*open*> and <*close*> are the opening and closing fold strings.

Folding may also be introduced on a file basis by defining the sold markers in the magic string by assignment of the open and close fold strings through the variables **fold-open:** and **fold-close:**. The macro file `fold.emf` does this to fold a MicroEmacs macro file as follows:

```
; -!- emf: fold-open: ";OPEN" ; fold-close: ";CLOSE" ; -!-

;OPEN fold-idle
0 define-macro fold-idle
    !if &lfind "|yank|reyank|" @cl
        fold-region
        2 screen-update
    !endif
    !if &exi :fold-idle
        execute-line :fold-idle
    !endif
!emacro
;CLOSE
```

## SEE ALSO

**$line-flags**(5), **buffer-init**(3), **collapse-all**(3), **collapse-current**(3), **narrow**(2m), **narrow-buffer**(2), **set-mark**(2).

Ref: fold-create(3) File: m3mac073.3 Date: 2022/06/12

## NAME

ftp - Initiate an FTP connection

## SYNOPSIS

**ftp**

## DESCRIPTION

**ftp** initiates a File Transfer Protocol (FTP) connection to a remote host on the network. Using FTP, editing of files may be performed in much the same way as on the local file system. Directory listings may be retrieved and traversed using the mouse or cursor keys. Using the directory listing, files may be transfered to/from the remote host to the local machine.

On issuing the command then a dialog is presented to the user which is used to open the connection. The dialog entries are defined as follows:

**Registry File**
: The name of a MicroEmacs registry file which is used to store the FTP information. If a registry name is provided then all FTP address information is stored in the registry file and saved for later sessions. Be aware that password information is saved in this file as plain text if a password is entered into the site information.

  If the registry information is omitted then the information is not saved between sessions.

**Site Name**
: An ASCII pseudo name for the remote host. The pull-down menu may be used to select existing sites that have been previously entered.

**Host Address**
: The address of the host, this may be an IP address (`111.222.333.444`) or a DNS name (i.e. `ftp.mysite.com`).

**User Name**
: The login name for the site. If this is omitted then **$user-name**(5) is used by default.

**Password**
: The password used to enter the site for the given login name. If the password is NOT supplied then the user is prompted for the password when a transaction takes place. If the password is omitted and left to promt then it is not stored in the registry.

  Take note of the comments provided above regarding the password information.

**Initial Host Path**
: The starting directory at the remote host. If this is omitted then the root directory ('/') is used by default.

  On selecting **Connect** then a FTP connection is opened and the initial directory appears as a directory listing, if the initial path is a file then the file is loaded into the editor.

  Thereafter the file may be edited within the editor as normal, on a write operation then the file is written back to the host, via FTP.

On opening a FTP connection the progress of the transfer, and the FTP commands issued, may be viewed in the `*ftp-console*` buffer. This buffer may automatically appear depending upon the value of the **%ftp-flags**(5) variable.

## NOTES

**ftp** is a macro implemented in `ftp.emf`. This uses the underlying command **find-file**(2) to implement the FTP transfer.

FTP files can be directly loaded and edited using the standard file commands such as **find-file**(2).

The FTP addresses are retained in a registry file (see **erf**(8)). The registry file is automatically loaded when MicroEmacs starts up each session. The current site information may be viewed using **list-registry**(2) and is located at the following registry addresses:

**/url** Data value is file system location of the FTP registry file.

**/url/ftp/**<*hostName*>
The name of the host to which the connection is to be made.

**/url/ftp/**<*hostName*>**/host**
The name or IP address of the remote host

**/url/ftp/**<*hostName*>**/user**
The user name used to log into the remote host.

**/url/ftp/**<*hostName*>**/pass**
The user password to the remote host. If this entry is empty then the user is always prompted for the password when the connection is made.

**/url/ftp/**<*hostName*>**/path**
The initial path at the remote site.

When a FTP connection is initiated then the connection (socket) remains open for a period of approximately 4 minutes from the last transfer time, after that the connection is automatically closed and is re-initiated if required again.

**NOTE:** For windows platforms then the resultant executable must be built with URL support enabled, for UNIX platforms socket support is automatically enabled.

## BUGS

Directory completion is not available when the current working directory is an FTP address. To work around this from the command line, select <RETURN> to get a directory listing of the current directory and select the file(s) from the directory to load.

## SEE ALSO

**%ftp-flags**(5), **$user-name**(5), **erf**(8), **find-file**(2), **file-op**(2), **list-registry**(2).

## NAME

gdb - GNU Debugger
dbx - UNIX Debugger

## SYNOPSIS

**gdb** "*program-name*" "*additional-args*"
**dbx** "*program-name*" "*additional-args*"

## DESCRIPTION

**gdb** and **dbx** provide an editor interface to the GNU and native system debuggers, respectively. On running either command then an interactive shell window is opened to the debugger command line interface. MicroEmacs then interprets the information from the debugger interface and opens files and hilights the current line as required. The current line is maintained while single stepping through the code.

Buffers opened and referenced by the debugger have the key `F9` bound to setting a break point.

The input mode in the shell window may be toggled using key **f3**

## NOTES

**gdb** and **dbx** are macros defined in file `hkipipe.emf`.

## SEE ALSO

**perldb**(3), **ipipe-shell-command**(2).

Ref: gdb(3) File: m3mac046.3 Date: 2022/06/12

## NAME

gdiff - Graphical file difference
%gdiff-com - Gdiff diff(1) command line

## SYNOPSIS

**gdiff** "*version1*" "*version2*"
`%gdiff-com` "*string*"; Default is "`diff -c -w`"

## DESCRIPTION

**gdiff** is a macro utility that facilitates the merging of two files (typically with different modification revisions). The changes between the revisions are hilighted with color, allowing modification regions and lines to be selected for the generation of a newer revision file, which might encompass selected modifications from each of the base revisions.

**gdiff** executes the **diff**(1) command with the command line set by the **%gdiff-com**(5) variable and the user supplied *version1* and *version2*. The output is displayed in two buffer windows, side by side, and the differences in the lines are hilighted to show the changes. In addition the content of the two buffers is *normalized* such that both windows are aligned at the same line position, allowing the changes in the text to be viewed in both windows at the same time.

Whilst in **gdiff** view mode then both scroll bars (if visible) are *locked*, such that either scrolls BOTH windows at the same time. Other key commands are disabled, as are the menu interactions. The short cut keys are defined as follows:

`esc h/A-h` - View the help page.
Invokes the display of a OSD help box, summarizing the interaction commands

`C-up` - Move to previous difference
Moves to the previous changed region above the current cursor position.

`C-down` - Move to next difference
Moves to the next changed region below the current cursor position.

`left mouse button`
`space`
`enter`
`r` - Select difference version

Selects the difference version of the currently selected window. The region is hilighted as the required region to be incorporated into the new revision.

`R` - Select neither version.
Marks both regions as not required.

`l` - Line select current version
Selects the current line from the region as being included, without including ALL of the region modifications.

`L` - Line select neither version
Discards lines from both revisions of the file.

`g` - Globally selects the current version.
: Shortcut allows ALL modifications to the current side to be accepted. This is typically the fastest method to select all changes, minor region adjustment may then be performed on those regions which are inappropriately included by the selection.

`G` - Globally selects neither version.
: Marks all regions as not being acceptable.

`C-x C-s` - Save current side
: Saves the current window to the specified file, merging the selected changes between the two revisions. Note that the save only operates iff all hilighted changes have been selected.

`C-x C-w` - Save current side as
: Same as **Save current side** except the user is prompted to enter a new filename to which the modifications are written.

`C-x k` - Exit graphical diff
: Exits the **gdiff** utility.

### Hilighting

The hilighting within the windows is dependent upon the color scheme selected, in general the following hilights apply:

normal text
: No change

cyan/grey
: Addition/removal of line(s)/region(s) between files.

yellow Modification in line(s)/region(s).

green/red
: Selected region, red or green is attributed to a selection for each window.

## NOTES

**gdiff** is a macro defined in `gdiff.emf`, inspired by the GNU utility of the same name **gdiff**(1)

**diff**(1) must be executable on the system before **gdiff** can function. The **diff**(1) invocation must include the *context* difference, which annotates the differences with a +, − or ! markers. **diff**(1) is typically invoked with the options **-c -w**.

**diff**(1) is a standard utility on UNIX systems. For Windows 95/NT a version of GNU **diff** may be found at:

*<ftp.winsite.com/ftp/pub/pc/winnt/misc/gnudiff.zip>*

For MS-DOS users, a DJGPP port of GNU **diff** is also available on the net. A commercial version of **diff** is also available from MKS.

## SEE ALSO

**compare-windows**(2), **compile**(3), **diff**(1), **gdiff**(3f), **xgrep**(3), **%xgrep-com**(5).

Ref: gdiff(3) File: m3mac044.3 Date: 2022/06/12

## NAME

generate-password - Generate a password with given criteria
insert-password - Create and insert a new password string
password-to-phonic - Convert a password to phonic string

## SYNOPSIS

*n* **generate-password**
*n* **generate-password** "*flags*"
**generate-password** "*length*" "*flags*"
*n* **insert-password** "*length*" "*flags*"
**password-to-phonic** "*password*"

## DESCRIPTION

The password commands are typically used by a system administrator to generate new random passwords or to phonically unambiguously describe a password.

**generate-password** is a low level macro for setting the **$result**(5) variable to a new password meeting the criteria defined by combinations of argument *n* and optionally passed arguments "*length*" and "*flags*".

If argument *n* is given then the lower byte (0x0ff) defines the password length, otherwise the "*length*" must be given. When a word base password is generated the length is the number of words, if a character based password is to be generated the length is the number of characters. The maximum length of the password is 255 bytes/words.

If the argument *n* is given and its value is greater than 255 then the bits above 255 are used to determine the flags, see list below. Otherwise the *flags* must get given and are defined as follows:

**w** Generate a random word(s) based password, the given length is used to define the total number of letters to be used by one or more words selected randomly using the spelling dictionaries. If flag **u** is also given then the first or last letter of each word may randomly be turned to uppercase, if flag **!** and/or **b** are given then each word will be separated by a 'special' character (see flags below) otherwise each word is separated by a '-' for clarity. Finally if flag **d** is given then a random two digit number is appended to the password. This can be specified by bit 0x0100 in argument **n**.

**h** Generate a random password using hexadecimal characters, i.e. 0-9 and a-f. If the **u** flag is given then uppercase letters are used instead. No other flag apart from **N** can be used. This can be specified by bit 0x0200 in argument **n**.

**d** The 10 digit characters, 0-9, are added to the character class used when not generating a word or hex based password. This can be specified by bit 0x0400 in argument **n**.

**l** The 26 lowercase letters, a-z, are added to the character class used when not generating a word or hex based password. This can be specified by bit 0x0800 in argument **n**.

**u** The 26 uppercase letters, A-Z, are added to the character class used when not generating a word or hex based password. This can be specified by bit 0x1000 in argument **n**.

**!** 24 'special' characters are added to the character class used when not generating a word or hex based password, these characters include !, $, % & ... but not brackets or white spaces. This can be specified by bit 0x2000 in argument **n**.

**b** 8 bracket characters ([, ], {, }, (, ), <, >) are added to the character class used when not generating a word or hex based password. This can be specified by bit 0x4000 in argument **n**.

**s** The space character is added to the character class used when not generating a word or hex based password. This can be specified by bit 0x8000 in argument **n**.

**e** When the **e** flag is given then at least one character from each of the main requested character classes, i.e. digits, lowercase letters, uppercase letters and specials, must be present in the generated password, if not another password is generated until it does. This can be specified by bit 0x10000 in argument **n**.

**D** When generating a word based password and this flag is given the current spelling dictionary is used, when not given a fixed dictionary of 7776 English words is used. Caution should be used when using a general MicroEmacs dictionary as the words can be rude, hard to spell and have accents, the alternative password word list contains only relatively unambiguous, easy to spell non-expletive words. This can be specified by bit 0x20000 in argument **n**.

**N** When given, flag **N** does not update the user's last or default password criteria, this flag can be used by macros to avoid affecting the user experience. This can be specified by bit 0x40000 in argument **n**.

**insert-password** uses **generate-password** to generates a new random password and insert it into the current buffer with the phonic form. The *length* and *flags* arguments are the same as **generate-password**, see above, however argument **n** can be used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** When set the user is prompted for the *length* and the *flags*, however when not set the default or last values will be reused.

**0x02** When set the phonic form will not be inserted, however when not set a phonic form will be inserted in brackets after the password if the password is not word of hex based and contains something other than digits.

**password-to-phonic** converts a string of characters to the phonic alphabet, the command requires a single argument *password* which is converted to a phonic string. The command **password-to-phonic** "*y5Ck*" inserts the phonic string into the buffer:

```
yankee - Five - CHARLIE - kilo
```

## NOTES

The password commands are macros defined in `password.emf`.
generate-password can be used to generate a large random number sequence by reducing the character class to just **d**, for example:

```
insert-password 30 "d"
```

Will insert a 30 digit random sequence. Similarly the **h** flag can be used instead to create a random byte sequence in hex format.

## SEE ALSO

**$random**(5), **$result**(5), **insert-string**(2).

Ref: generate-password(3) File: m3mac086.3 Date: 2024/11/23

## NAME

generate-tags-file - Generate a tags file

## SYNOPSIS

*n* **generate-tags-file** [ "*tag-command*" ]

## DESCRIPTION

The **generate-tags-file** command provides an interface to tag file generation. Typically the "*tag-command*" argument will not be required as the current buffer will automatically configure **generate-tags-file** on how tags are generated for the current buffer's file type. See the notes below for more information on configuration.

**generate-tags-file** supports two different methods of tag generation, firstly via a MicroEmacs macro file and secondly by an external shell command (such as **ctags**(1)). It is generally configured in the current buffer's setup hook.

If a macro file is used a setup dialog is opened if an argument of 0 is given to **generate-tags**. This dialog can be used to configure which type of tags are required and the starting directory (useful when using recursive tags over a source tree). Note that not all tag types are available for all file types.

The generated tags file can then be used by the **find-tag**(2) command.

## NOTES

**generate-tags-file** is a macro defined in file `gentags.emf`.

**generate-tags-file** can be configured in one of 2 ways:

When a MicroEmacs macro file (such as `ctags.emf`) is to be used, simply give the name of the macro file to be run as the "*tag-command*" argument. Alternatively set the variable **.**<*$buffer-fhook*>**.tags** to this name, e.g. for C files

```
set-variable .fhook-c.tags "ctags"
```

Note the "`.emf`" extension is assumed.

When an external shell command is to be used, set the *tag-command* to the shell command-line prefixed with a '`!`' character, for example to use **ctags**(1) try the following:

```
set-variable .fhook-c.tags "!ctags *.c *h"
```

Note that the generate-tags dialog is not available in this mode of execution.

## SEE ALSO

**find-tag**(2).

Ref: generate-tags-file(3) File: m3mac056.3 Date: 2026/01/18

## NAME

git - MicroEmacs GIT interface
git-setup - MicroEmacs GIT interface setup
git-add - MicroEmacs GIT interface - add items to the index
git-blame - MicroEmacs GIT interface - annotate a file
git-cat-file - MicroEmacs GIT interface - output specific commit of a file
git-commit - MicroEmacs GIT interface - commit changes to repository
git-diff - MicroEmacs GIT interface - diff changes
git-fetch - MicroEmacs GIT interface - download from another repository
git-gdiff - MicroEmacs GIT interface - graphical diff changes
git-log-commit - MicroEmacs GIT interface - show information about a specific commit
git-log-file - MicroEmacs GIT interface - list log of changes
git-move - MicroEmacs GIT interface - move an item
git-pull - MicroEmacs GIT interface - download and integrate another repository
git-push - MicroEmacs GIT interface - update remote repository
git-remove - MicroEmacs GIT interface - remove items
git-revert - MicroEmacs GIT interface - revert local changes
git-status - MicroEmacs GIT interface - status of items
git-switch - MicroEmacs GIT interface - switch or create branches
git-unstage - MicroEmacs GIT interface - unstage items

## SYNOPSIS

*n* **git**
**git-setup**
**git-add**
**git-blame**
**git-cat-file**
**git-commit**
**git-diff**
**git-fetch**
**git-gdiff**
**git-log-commit**
**git-log-file**
**git-move**
**git-pull**
**git-push**
**git-remove**
**git-revert**
**git-status**
**git-switch**
**git-unstage**

## DESCRIPTION

The git and sub-commands provide MicroEmacs with an interface to the **GIT**(1) version control system, see the documentation provided with GIT for more information on its features and use.

The MicroEmacs **git** command opens up a modified **file-browser**(3) with an additional "`*git-console*`"

window. The "`*files*`" window includes additional flags within the Attributes column showing the GIT state. The browser by default opens with 3 windows including a representation of the file tree. When **git** is provided with a numeric argument *n* of zero (0) then a single window displaying the directory is used only (i.e. **0 git**)

The functionality of the file-browser is the same as a non-GIT folder with the exception that additional GIT item controls are located in the mouse context menu (opened by clicking the right mouse button in the `*files*` buffer). This menu item opens another sub-menu providing access to the items below. When selecting commands than a pop-up check-box may be displayed which allows options to be selected for the command.

**Current State**
: Executes the git-status command.

**Commit files**
: Executes the git-commit command.

**Diff files**
: Executes the git-diff command.

**Graphical Diff**
: Executes the git-gdiff command, one file must be selected.

**Item Log**
: Executes the git-log-file command, one or more items must be selected.

**Item Blame**
: Executes the git-blame command, one file must be selected.

**Move items**
: Executes the git-move command, one or more items must be selected.

**Add items**
: Executes the git-add command, one or more items must be selected.

**Remove items**
: Executes the git-remove command, one or more items must be selected.

**Revert items**
: Executes the git-revert command, one or more items must be selected.

**Unstage items**
: Executes the git-unstage command, one or more items must be selected.

**Update files**
: Executes the git-pull command.

**Clear Git console**
: Clears the **git-console** buffer.

**GIT Setup**
: Executes the git-setup command.

### Sub-Commands

The **git-add** command adds the current buffer or selected items to the index (also known as the staging area). Note that a *GIT commit* is required to make the addition permanent.

The **git-blame** command opens a **git-blame** window annotating the commit and author of each line in the current buffer or single selected file.

The **git-cat-file** command opens a *<*file-name*>:<*commit*> window with the contents set to the specified commit of a selected file.

The **git-commit** command commits any changes made to the index (staging area) back to the repository, local changes that have not been added to the index are ignored. The user is prompted for a commit log message. While the git commit command can commit changes to local files at the same time, this feature is not supported as it has been deemed confusing and dangerous as it will always commit *ALL* index changes back, not just selected ones. As a result all modified and new files must first be added to the index using **git-add**.

The **git-diff** command opens a **git-diff** window displaying the differences between the current buffer or selected local files and the repository version.

The **git-fetch** command downloads branches and changes from the external repository, the command does not change the current index or local files.

The **git-gdiff** command opens a **gdiff**(3) window displaying the differences between two versions of the current buffer's file, the two versions being selected from the command's dialog. The command operates on a single file only. Where the right hand side is the local file any changes made may be saved back over it, repository versions cannot be saved.

The **git-log-commit** command opens a **git-log-commit** windows displaying information about a specified GIT commit.

The **git-log-file** command opens a **git-log-file** window displaying the GIT commit log history of the current buffer or a combined log of the selected files.

The **git-move** command moves (i.e. same as rename when the destination directory is the same) the current buffer's file or selected items, any required changes to the buffer should be saved first and the new file loaded once moved. Note that unlike copy, GIT does not support a direct local to reposition move. The advantage of using move over simply deleting and adding a new item is that the item's change log history will be maintained.

The **git-pull** command performs a **git-fetch** first followed by a forward only integration. The command can optionally perform a `git stash` first and, if the integration is successful it can optionally perform a `git stash pop`. The integration can fail when there are any changes to the local repository (i.e. commits), index changes (i.e. local modifications that have been added but not committed) or to local files. If there have been local commits then these changes will need to be merged, **git-pull**'s use of forward only will successfully integrate trivial merges, but more complicated merges will need to be done manually using git merge or rebase and is beyond the scope of this documentation. To merging changes any index changes will need to be committed first, local changes can be committed or handled by use of the Stash options. Note that if the pull requires a manual merge, any local changes that have been stashed will need to be manually popped once the merge has been completed.

The **git-push** command updates the remote repository with all changes made to the local repository. This command can fail if the local repository is out of date, i.e. other commits have been made or pushed to the remote repository since the last fetch. When this happens the **git-fetch** command should be called and changes integrated, see notes on **git-pull** above.

The **git-remove** command removes the file of the current buffer or selected files and flags them for removal in the index (staging area), the files will be permanently removed from the repository when the change is committed back. PLEASE NOTE THIS CAN LEAD TO LOST DATA!!! As the local file will be removed the file-browser will no longer list the file, use the **git-status** command to list change as this list includes all items flagged for removal.

The **git-revert** command reverts local modifications made to the current buffer's file or selected items back to the last checked out version, discarding the local changes; unlike **git-remove** this

does not affect the repository. By default the file is restored from the index (staging area), so any changes already staged via **git-add** are kept; enabling the `Revert staged changes` option instead restores the file from `HEAD`, discarding both staged and unstaged local modifications.

The **git-status** command opens a **git-state** window and shows the state of all the items within the current directory, including the modification state, and optionally whether items are out-of-date (i.e. have been modified in the repository).

The **git-switch** command switches to an existing branch, or creates and switches to a new one, taking over the branch-switching role formerly played by **git-checkout** (file restoration duties have moved to **git-revert** and **git-unstage** instead). Unlike **git-fetch** and **git-pull** the branch name cannot be left blank, since there is no configured default branch to fall back on; the special name `-` (a single hyphen) switches back to the previously checked out branch. Enabling `Create a new branch` creates the named branch from the current `HEAD` before switching to it; enabling `Track the branch` additionally sets up an upstream tracking relationship for it. If the switch would otherwise be blocked by local modifications, enabling `Stash local changes first` stashes them beforehand. Unlike **git-pull**, which offers an option to pop the stash back once its operation has completed, **git-switch** does not offer this, since immediately after a branch switch the stash would be applied against the wrong branch's content; instead recover it later with `git stash pop` once back on the original branch (use `git stash list` first to identify the right entry if more than one stash is outstanding).

The **git-unstage** command removes the current buffer or selected items from the index (staging area); any local modifications already made to the working tree files are left untouched. This is the opposite of **git-add**.

### Setup

**git-setup** provides a dialog interface to configure the MicroEmacs GIT interface if required. On a typical system installation then it is sufficient to have the **git** program in the **$PATH** and the GIT tools will operate from a working copy of a repository.

**git-setup** may also be accessed from **user-setup**(3) from the **Miscellaneous** configuration. The the dialog provides the following configuration tabs and options:

**General**

`GIT Command`
: Sets the git command-line program name, this can be an absolute path.

`Global opts`
: Sets command-line arguments which are to be always given to the git command, the default of `--no-pager` informs GIT that the program is being run in the background so must not wait for user input.

`Default Rep`
: Defines the default GIT repository, this is used by commands like **git-fetch** when a branch is specified and is typically `origin`.

`Default Branch`
: Defines the default GIT branch which is typically `main` or **master**, many commands dynamically get the current branch for GIT when required.

`DEnable Support on Start-Up`
: When enabled the git support macros are loaded on start-up enabling the file-browser to detect GIT controlled directories. If this is not enabled then the file-browser will only identify GIT controlled directories once a GIT command has been executed.

**Tools** In this tab each sub-command can be selected and their default option settings changed as required.

## NOTES

**git** and sub-commands are macros defined in file `git.emf`.

The git-show, git-show-file and git-log commands were renamed to git-log-commit, git-cat-file and git-log-file in August 2026 to align them with the equivalent SVN and VCS commands.

## SEE ALSO

**svn**(3), **cvs**(3), **ClearCase**(3), **gdiff**(3), **file-browser**(3), **$rcs-file**(5).

Ref: git(3) File: m3mac106.3 Date: 2026/08/03

## NAME

grep - Search a file for a pattern
grep-for-region - Grep current directory for current region

## SYNOPSIS

**grep** "[*grep-options*] <*search-pattern*> [*file-mask*] [*file-mask ...*]" ["*starting-path*"]
**grep-for-region** [*search-pattern*]

## DESCRIPTION

**grep** performs a search for a pattern and displays all lines that contain that pattern within files in the current directory (or from *starting-path*) using built in MicroEmacs constructs rather than relying on any external executables such as **grep**(1). **grep** operates in exactly the same way as a traditional *grep* command line utility and uses a command line option sequence from the MicroEmacs command line. The *grep-options* allow recursive and regular expression searches, searching against the file modification time etc. The command line takes the user supplied *grep-options*, *search-pattern*, *file-mask* and *starting-path*. The output of the command is piped into the ***grep*** buffer.

*grep-options*

Defines the *options* that are specified on the command line, the default options may also be defined with the **find-setup**(3) command invoked from the command line or from the the **Tools** menu. The *grep-options* are defined as follows:

**–** Last option, the following string must be the *search-pattern* which may start with a '-'.

**-a** Re-read all out of date buffers, prevents a manual prompt, equivalent of Yes to all. This is the inverse of **-o**

**-b** Ignore binary files.

**-B** Do not ignore binary files.

**-c** Command return the number of matchs found rather than the default, which is simply 0 for one or more matches or 1 if no matches were found.

**-C** Command returns 0 (non-error for command-line tool) regardless of if it finds any matches.

**-d** List directories or grep in directory listings.

**-D** Do not list directories or grep in directory listings.

**-e** Only include the matching string in the `*grep*` output buffer.

**-f** List/search files.

**-F** Do not list/search files.

**-g** File masks are regex (i.e. .*.emf)

**-G** File masks are not regex (i.e. *.emf)

**-i** Ignore search pattern case (case insensitive).

**-I** Do not ignore search pattern case (case sensitive).

**-l** Use existing buffer of loaded files.

**-L** Do not use existing buffer of loaded files.

**-m** Search pattern is a regular expression.

**-M** Search pattern is not a regular expression.

**-o** Do not re-read all out of date buffers, prevents a manual prompt, equivalent of No to all. This is the inverse of **-a**.

**-p** Starting path specified

**-P** Starting path not specified

**-q** Enables quiet mode for reduced print-out.

**-r** Recurse into sub-directories

**-R** Do not recurse into sub-directories

**-S** <*size*>
File size - less than or equal, the following size formats are supported:

**-S** <*number*> File size specified bytes. **-S** <*number*>**k** File size specified in Kilobytes.

**-s** <*size*>
File size greater than; supported size formats are the same as **-S**.

**-T** <*time*>
Modification time - older than, the following time formats are supported:

**-T** *yy-mm-dd* **-T** *yyyy-mm-dd* **-T** *yyyy-mm-dd hh:mm:ss* **-T** *days* Offset time in days from midnight this morning, e.g. **-T 0** would match all files not modified today. **-T** *days.hours[.minutes[.seconds]]* Offset time in , e.g. **-T 1.1.30** matches all files 25 hours and 30 minutes old.

Note that **-T 1** is not the same as **-T 1.0**, except at midnight.

**-t** <*time*>
Modification time - newer than or equal, supported time formats are the same as **-T**.

**-u** Removes the default file size limit of 100MB allowing files of any size to be search. Note that MicroEmacs uses **page-file**(3) on files larger than 100MB to avoid using too much memory.

**-U** Enforces the 100MB file size limit.

**-x** Use MicroEmacs **$file-ignore**(5) to exclude unimportant files

**-X** Do not use MicroEmacs **$file-ignore**(5).

**-&** Launch grep using a separate process, this stops the current MicroEmacs hanging during the grep process which can be useful for time consuming searches.

*search-pattern*
The search pattern to look for.

*file-mask*
The files to search for e.g. ***.txt**, if no file mask is given then **grep** attempts to get a file list from the current buffer, this could either be a `*grep*` output buffer, in which case each file listed will be selected for the next grep, or some kind of file listing (this could be a loaded directory, an output from **find** or a source revision control system such as cvs or svn) in which case the files to be searched must be selected (line starts with a "*".

*starting-path*
: The directory location where the search should commence when the **-p** option is used. The default when **-p** is omitted is the location of the current buffer.

  **grep** sets **$result**(5) to the return of the command, which is `0` if matches were found or 1 of no match was found, unless options **-c** or **-C** are used.

  **grep-for-region** uses the **grep** command to search for the string defined by the current region within all files of the current directory. If no region is defined within the current buffer the the string is prompted for.

  **find-setup** invokes a dialogue and allows the default *file-options* to be specified when the **find** command is invoked.

## NOTES

- Trying to make the command-line look like the standard UNIX **grep**(1) command may be more of a hindrance than a help because its not grep. No support is provided for options like **-w**, **-l** etc.
- The default when no options are given is **-XGPRDfMIBL**, default options may be set via the **find-setup**(3) command (sets the */history/grep-com* registry entry).
- Use of **-d** searches for <*search-pattern*> within the MicroEmacs directory listing.
- It may be advantageous to enable the **-l** option default using **find-setup**(3).

The other options of the underlying **find**(3) command are allowed but provide no benefit to these commands.

## SEE ALSO

**%xgrep-com**(5), **find**(1), **find**(3), **find-setup**(3), **get-next-line**(2), **grep**(1), **replace-string-in-files**(3), **xfind**(3), **xgrep**(3), **rgrep**(3).

Ref: grep(3) File: m3mac089.3 Date: 2025/05/23

## NAME

Hearts - MicroEmacs version of the card game Hearts

## SYNOPSIS

**Hearts**

## DESCRIPTION

**Hearts** is a trick-taking card game for four players, played with a standard pack of 52 cards. The user plays the south hand against three computer opponents named West, North and East. The object of the game is to score as few points as possible.

Each player is dealt 13 cards. Before play begins each player passes 3 cards to another player, and receives 3 in return. The queen of spades may never be passed. The three cards received are held up out of the hand for a moment before they settle into it, so that they can be seen. The player to the dealer's left leads the first trick and may lead any card. Each player in turn must follow the suit that was led if they are able, otherwise they may play any card. The highest card of the suit led wins the trick, and the winner leads the next one.

By default there are no restrictions on what may be led or played - any card may be led to the first trick, hearts may be led at any time and points may be taken on the very first trick. Each of the three traditional restrictions may be turned on individually from the options page.

## SCORING

Under the default house scoring a hand is worth 33 points:

**Queen of spades**
: 10 points

**Ace of hearts**
: 5 points

**King of hearts**
: 4 points

**Queen of hearts**
: 3 points

**Jack of hearts**
: 2 points

**Two to ten of hearts**
: 1 point each

The classic scoring, in which every heart is worth 1 point and the queen of spades is worth 13 for a total of 26, may be selected from the options page.

If a single player takes every point in a hand they have **shot the moon**. That player scores nothing and every other player is given the full value of the hand instead.

The game ends when any player reaches the target score, 100 points by default. The player with the lowest score at that point is the winner.

## CONTROLS

The game may be played with the mouse or the keyboard. To play a card, or to select a card to be passed, click on it with the left mouse button. Cards selected for passing are raised out of the hand.

Using the keyboard:

*left* and *right*
: Move the highlight along the hand. The highlight wraps round at either end, so the quickest way to a card on the far side of the hand is often to run off the near edge.

*space* or *return*
: Play the highlighted card, or select it to be passed.

*d*
: Deal the next hand.

*p*
: Pass the selected cards.

*c*
: Clear the current pass selection.

**Continue**
: While the cards just received are on show, any key or click puts them into the hand and starts play without waiting for the pause to run out.

*o*
: Show the options page.

*s*
: Save the options and leave the options page.

*x*
: Quit the game. On the options page it cancels instead, so the game cannot be left by accident from there.

*esc h*
: This help page.

To the right of the board are the control buttons. These are shown only when they may be used, so **Deal** and **Options** appear between hands, and **Pass** and **Clear** while cards are being passed.

Note that the screen may be updated at any time using "*C-l*".

## OPTIONS

The options page is reached with the **Options** button between hands. Press *1* to *9* to change a setting, then **Save** ("*s*") to keep the changes and remember them between sessions, or **Cancel** ("*c*") to put everything back as it was. Saved changes take effect from the next hand dealt.

**Difficulty**
: Five levels from **Beginner** to **Expert**. The lower three play by judgement alone; the upper two also search, playing the hand out many times over against guessed distributions of the cards they cannot see before choosing. The weaker levels are weaker because they miss things - they do not always find the best card, and they do not keep track of who has shown out of a suit - not because they play at random.

Each level was measured against the one below it over several thousand hands, so every step up is a real increase in strength rather than simply a slower one.

**Passing** Pass left every hand, or rotate left, right and across. There is no hold hand.

**Scoring** House scoring for a 33 point hand, or classic scoring for a 26 point hand.

**Jack of Diamonds**
: The Omnibus Hearts variant, off by default. When enabled the jack of diamonds is worth -10 points to whoever captures it. It is not needed to shoot the moon, and the bonus still applies if the player shooting the moon takes it.

**First lead**
: When enabled the holder of the two of clubs must lead it to the first trick. Off by default, in which case the player to the dealer's left leads any card.

**Leading hearts**
: When enabled a heart may not be led until a heart has been played to some earlier trick, unless the hand holds nothing else. Off by default.

**First trick**
: When enabled no card carrying points may be played to the first trick unless there is no alternative. Off by default.

**Game ends at**
: The target score, 50, 100 or 200 points.

**Play speed**
: How quickly the computer players play their cards.

## NOTES

**Hearts** is a macro defined in `hearts.emf`.

The computer players use no information that is not available to the user. They see their own cards, the cards that have been played, the points taken and which players have shown out of a suit - all of it on the table for anyone to follow. At the higher levels they also count how many cards each player still holds, and deal the unseen cards out at random among them to search; the identity of those cards is never consulted.

## SEE ALSO

**Patience**(3), **Triangle**(3).

## NAME

ifill-paragraph - Format a paragraph

## SYNOPSIS

*n* **ifill-paragraph** (**esc q**)

## DESCRIPTION

**ifill-paragraph**, like **fill-paragraph**, fills the current paragraph from the left margin to the current fill column. In addition ifill-paragraph also recognizes joined bullet lists and fills each bullet paragraph separately.

See **fill-paragraph**(2) for more information on the process of filling paragraphs.

## EXAMPLE

Following are 2 copies of the same paragraph, the first has been filled using **ifill-paragraph**:

```
This  is the  main  paragraph  which  can be as long as  required,
following is a list of bullets, some with a sub-bullet  list. Here
is the list:
    a) The bullet paragraph can also be as long as required and it
       also can have a bullet  list  following  (sub-bullet  list)
       which will also be filled correctly. Here is the sub-bullet
       list:
       1. First  sub-bullet - again no length  restrictions,  this
          will be filled correctly.
       2. second sub-bullet - no problems.
       3. Third sub-bullet - again no length restrictions, this is
          getting boring.
    b) This is the second  major bullet and this can just carry on
       for ever, but all things must come to an
```

The following version has been filled using the normal **fill-paragraph**:

```
This  is the  main  paragraph  which  can be as long as  required,
following is a list of bullets, some with a sub-bullet  list. Here
is the  list:  a) The  bullet  paragraph  can  also  be as long as
required and it also can have a bullet list following  (sub-bullet
list) which will also be filled  correctly. Here is the sub-bullet
list: 1. First  sub-bullet  - again no length  restrictions,  this
will be filled  correctly. 2. second  sub-bullet - no problems. 3.
Third sub-bullet - again no length  restrictions,  this is getting
boring. b) This is the second major bullet and this can just carry
on for ever, but all things must come to an
```

## NOTES

**ifill-paragraph** is a macro defined in `format.emf`.

## SEE ALSO

**fill-paragraph**(2), **paragraph-to-line**(3).

Ref: ifill-paragraph(3) File: m3mac026.3 Date: 2022/06/12

## NAME

indent-increase - Increase the line indentation
indent-decrease - Decrease the line indentation

## SYNOPSIS

*indent-increase n* **indent-decrease**

## DESCRIPTION

**indent-increase** increases the indentation of a line or selection region by the numeric argument *n* tab stops defined by **$buffer-indent-width**(5). If the argument is negative then the indentation is reduced by the specified number of tab stops. The default when omitted is a single indentation level.

**indent-decrease** operates in the same way as **indent-increase** except the indentation is decreased.

## NOTES

**indent-increase** and **indent-decrease** are defined in `format.emf`.

## SEE ALSO

**$buffer-indent-width**(5).

Ref: indent-increase(3) File: m3mac082.3 Date: 2022/06/12

## NAME

info - Display a GNU Info database
info-on - Display Info on a given topic
info-goto-link - Display Info on a given link
$INFOPATH - GNU info files base directory
.info.path - Cached info search path

## SYNOPSIS

**info**

**info-on** *topic-str*

**info-goto-link** *link-str*

**$INFOPATH** *string*

**.info.path** *string*

## DESCRIPTION

**info** interprets the GNU *info* pages, and presents the info file information within a buffer window called `*info XXXXX`, where `XXXXX` is the name of the info file. The root of the info page is displayed and may be traversed by selecting the links with the mouse, or by using the standard *info* traversal keys.

The root of the *info* tree is, by default, a file called **dir**, which points to the other information sources. The default search paths for the *info* directories are:

`c:/info` - MS-DOS and MS-Windows (all).
`/usr/local/info` - All UNIX platforms.

The root directory may also be specified with the `$INFOPATH` environment variable. This is a colon (`:`) or semi-colon (`;`) separated list of directory paths which specify the locations of the info files, for UNIX and Microsoft DOS/Windows environment's, respectively.

**info-on** gets info on a user specified top level topic, e.g. "`gcc`", the info file "*topic-str*`.info`" must be found in the info search path.

**info-goto-link** gets and displays info on a user specified link or subject. The link may be within the currently displayed topic (the *link-str* need only specify the subject node name) or a subject within another topic (in which case the *link-str* takes the following form "`(`*topic*`)` *subject*").

## NOTES

**info** is a macro implemented in file `info.emf`.

When an **info** command is run for the first time, the info search path is constructed and stored locally in the command variable **.info.path**. This variable must be directly changed by the user if changes to the info search path are required.

## SEE ALSO

**info**(9), **info**(3f).

Ref: info(3) File: m3mac043.3 Date: 2026/08/01

## NAME

insert-template - Insert template file into current buffer
%company-ref - Company reference template

## SYNOPSIS

*n* **insert-template** [ "*template*" ]
**%company-ref** "*string*" ; See $COMPANY_REF$ below for default when undefined

## DESCRIPTION

**insert-template** is generally called by file hooks when the new buffer has been created as opposed to loaded from a file (see **major-mode**(2) and **$buffer-fhook**(5)).

**insert-template** uses **&find**(4) to locate and insert the required "*template*.etf" file. If successful, **insert-template** then replaces the following strings in the template:

`$ASCII_TIME$`
: To the current time. Inserts the output of **ascii-time**(3). The default format of the time string may be changed by setting variable **%ascii-time**(5).

`$COMPANY_NAME$`
: To the value of **%company-name**(5), or, if not defined, to the value used for `$USER_NAME$`. **%company-name** is usually set up in the company setup file defined in User setup.

`$COMPANY_REF$`
: This is set to either "`$COMPANY_NAME$ ($COMPANY_URL$)`" if the **%company-url** is defined, or simply "`$COMPANY_NAME$`" if not, the tags are then replaced as defined in this help. The value can be altered by setting the **%company-ref** variable.

`$COMPANY_URL$`
: To the value of **%company-url**(5), or if not defined to an empty string (i.e. ""). **%company-url** is usually set up in the company setup file defined in User setup.

`$FILE_NAME$`
: To the buffer name which is typically the buffer's file name without a path. A trailing "<##>" string (used to make the buffer name unique) is removed.

`$FILE_BASENAME$`
: To the buffer name, which is typically the buffer's file name without a path, and without the file extension. A trailing "<##>" string is removed before the extension is removed.

`$FILE_NAME_UD$`
: Same as **$FILE_NAME$**, but with the name capitalized and '.'s are replaced with '_'.

`$USER_NAME$`
: To the value of the registry entry "`/history/user-name`", or if not defined to the value "`<unknown>`". The user name is usually set up in the User setup dialog.

`$YEAR$`
: To the current year (4 digit number).

`$CURSOR$`
: To leave the cursor at this point, only one of these tokens should be used in the template and the token is removed.

The numeric argument *n* is used to modify the default behavior of the command, where the values are defined as follows:

*0* **insert-template**
: Inserts the default template for the current file hook, for example if the current file is a MicroEmacs macros file the hook will be `fhook-emf` so this will insert "`emf.etf`".

*0* **insert-template** '`<template-name>`'
: In this form "`<template-name>`" must be given in the execution line, i.e. will not be prompted for, and only the **&find**(4) macro function is used to locate the template file, this will include the **$user-path**(5) and the other paths on the **$search-path**(5) but will not include directory "`$user-path/<fhook>/`", see next option.

**insert-template** '`<template-name>`'
: When no argument is supplied, the file of the given template is first search for within the directory "`$user-path/<fhook>/`" and only if not found will it then use the **&find**(4) macro function to locate the file.

*#* **insert-template**
: When **n** is non-zero **insert-template** inserts the template matching "`$user-path/<fhook>/#_<na`" where '`#`' can be prefixed with zeros, i.e. '`3 insert-template`' would insert templates `3_test.etf` or `003_hello.etf` if located in the correct directory. If multiple files match this base criteria the first file will be inserted.

## EXAMPLE

The following is taken from hkmake.emf and inserts the "*makefile.etf*" template if the buffer has been created.

```
define-macro fhook-make
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "makefile"
    !endif
    set-variable $buffer-hilight .hilight.make
    -1 buffer-mode "tab"                ; Normal tabs please !!!
    1 buffer-mode "indent"
    1 buffer-mode "time"
!emacro
```

## NOTES

**insert-template** is a macro defined in `etfinsrt.emf`.

**magic**(2m) mode is always used to perform the the search/replace so the replace strings should be appropriate for **magic**.

## SEE ALSO

**major-mode**(2) **%ascii-time**(5), **$buffer-fhook**(5), **%company-name**(5), **%company-url**(5), **&find**(4), **ascii-time**(3).

## NAME

ishell - Open a interactive shell window
%ishell - User defined ishell shell

## PLATFORM

Windows '95/'98/NT/2K/XP/Vista/7/10/11 - windows
Cygwin - cygwin
Unix - All variants.

## SYNOPSIS

*n* **ishell**
***%*ishell** = <*shell*>

## DESCRIPTION

**ishell** creates an interactive shell window within the a MicroEmacs buffer window, providing access to the native operating systems command shell. Within the window commands may be entered and executed, the results are shown in the window.

On running **ishell** a new buffer is created called `*shell*` which contains the shell. Executing the command again while that shell is still running simply pops up the existing `*shell*` window, a new shell is started in the same buffer once the current shell has been exited.

### Numeric Argument

The numeric argument *n* is a bit based flag where:

**0x0f** Selects the shell to be executed from the shell list, see "Selecting the Shell" below. When no argument is given a value of `1` is used, selecting the first shell in the list.

**0x10** Disables the interpretation of ANSI color escape sequences, see bit **0x1000** of **ipipe-shell-command**(2). Color is enabled by default, so `17 ishell` runs the first shell in the list without color.

If a -ve argument is given then an existing `*shell*` window is popped up as usual, but no new shell is created if one is not already running.

### Selecting the Shell

The shell that is executed is selected from a platform dependent list of shells using the numeric argument, the list being as follows

**Unix** `$SHELL`

**Windows**
`$COMSPEC` (typically `cmd.exe`), `powershell`

**Cygwin** `$COMSPEC` (converted to a cygwin path), `powershell`, `$SHELL`

When the list contains only one shell that shell is always used and the numeric argument is ignored. Otherwise, if the argument does not select a valid entry in the list

(e.g. `0 ishell`) then the user is prompted to "Select shell" from the list, a shell which is not in the list may also be entered.

An alternative shell may be added to the list by setting the user variable **%ishell** to the command used to launch it. The shell is inserted at the head of the list and therefore becomes the default, the standard platform shells remain available via the numeric argument. **%ishell** is typically set in the user or company setup macro file, see **user-setup**(3), for example:

```
set-variable %ishell "C:/Program Files/Git/bin/bash.exe"
```

Additional controls are available within the shell window to control the editors interaction with the window. The operating mode is shown as a digit on the buffer mode line (2/3/4) which is defined in the following sections. The operational mode is changed with the **F3** function key.

**Mode 2 - Browse Mode**
Locks the window and allows local editing to be performed. All commands entered into the window are interpreted by the editor. Mode 2 is typically entered to cut and paste from the window, search for text strings etc. In mode 2, a **2** is shown on the mode line.

**Mode 3 - Semi-immersion**
The normal operating mode, text typed into the window is presented to the shell window. Translation of MicroEmacs commands (i.e. beginning-of-word) are translated and passed to the shell. For interactive use this is the default mode. In mode 3, a **3** is shown on the mode line.

**Mode 4 - Full Immersion**
All input is passed to the shell, no MicroEmacs commands are interpreted and keys are passed straight to the shell window. This mode is used where none of the keys to be entered are to be interpreted by MicroEmacs. Note that you have to un-toggle mode 4 before you can swap buffers as this effectively locks the editor into the window.

To exit the shell then end the shell session using the normal exit command i.e. "`exit`" or "`C-d`" as normal and then close the buffer. A short cut "`C-c C-k`" is available to kill off the pipe. However, it is not recommended that this method is used as it effectively performs a hard kill of the buffer and attached process.

### UNIX

The UNIX environment uses the native **pty** support of the operating system. The shell that is opened is determined by the conventional `$SHELL` environment variable, see "Selecting the Shell" above.

The shell window assumes that the user is running some sort of Emacs emulation on the command line (i.e. `VISUAL=emacs` for **ksh**(1), **zsh**(1), **bash**(1), **tsch**(1)) and passes Emacs controls for command line editing.

The shell window understands re-size operations and provides a limited decoding of the *termio* characters for a VT100 screen. From within the shell window it is possible to run the likes of **top**(1) correctly. It is even possible to run another MicroEmacs terminal session !!

### WINDOWS

The Windows environment provides a very poor command shell facility, this is more of a fundamental problem with the operating system than anything else. Unfortunately NT/2K/XP/7/10/11

are no better than Windows '95/'98, stemming from the fact that Microsoft Windows is actually a huge Window Manager (hindered by legacy issues of MS-DOS).

For those familiar with the UNIX command shell then it is strongly recommended that the **cygwin**(3) BASH shell is used as an alternative. This is a far more responsive shell window and provides the familiar Emacs editing of the command line.

The command shell under Microsoft Windows is slow and very unresponsive, this would appear to be a problem with the *command.com* as the same problems are not apparent with the cygwin environment. However, the shell window is good for kicking off command line utilities (such as *make*) or any command line processes that generates output on *stdout* as all of the output is captured in the buffer window which can be scrolled backwards for post analysis. For this very reason it is more preferable to a standard MS-DOS box.

Where a ConPTY pseudo terminal can be created, see bit **0x08** of **ipipe-shell-command**(2), utilities which use embedded screen control characters may be run as the editor interprets them. If a pseudo terminal cannot be created a warning is issued as such utilities will not then work.

### Changing the Shell

The shells offered are the command processor defined by the environment variable `$COMSPEC` (typically `cmd.exe`) and `powershell`, selected by the numeric argument as described in "Selecting the Shell" above. Where the user is using a different command shell (i.e. 4-DOS), then problems may arise if this is an old 16-bit executable, an alternative shell may be made the default by setting **%ishell**.

Note that **%ishell** replaces the **$ME_ISHELL** environment variable used by earlier versions of MicroEmacs, a **$ME_ISHELL** setting is no longer used and must be converted to **%ishell**.

## Bugs

### WinOldAp

**Winoldap** is created by the Microsoft environment whenever a shell is created. On occasions where processes have terminated badly the user may be prompted to kill these off; this is the normal behaviour of Windows. It is strongly advised that the shell is always exited correctly (i.e. `exit`) before leaving the editor. The Windows operating system for '95/'98 is not particularly resilient to erroneous processes and can bring the whole system down. NT/2K/XP do not appear to suffer from these problems (so much).

### Locked Input

There are occasions after killing a process the editor appears to lock up. This is typically a case that the old application has not shut down correctly. Kill off the erroneous task (`Alt-Ctrl-Del` - *End Task*) then bring the editor under control using a few `C-g` **abort-command**(2) sequences.

## NOTES

The **ishell** command uses the **ipipe-shell-command**(2) to manage the pipe between the editor and the shell. The window is controlled by the macro file `hkipipe.emf` which controls the interaction with the shell.

## SEE ALSO

**ipipe-shell-command**(2), **cygwin**(3), **user-setup**(3).

Ref: ishell(3) File: m3mac034.3 Date: 2026/08/18

## NAME

item-list - Abbreviated search and list buffer contents.
item-list-find - Find the selected item in the item list
item-list-close - ”Close the item list”

## SYNOPSIS

**item-list** (**F7**)
**item-list-find**
**item-list-close** (**esc F7**)

## DESCRIPTION

**item-list** performs a regular expression search of a buffer, presenting a list of the located text and associated types in a separate window which is presented to the left of the buffer window. **item-list** is a generic function that interacts with the buffer environment variables to present abbreviated buffer information to the user.

The regular expression search strings are predefined in the language templates. To add support for a new buffer type a list of search/replace strings must be created. The search strings must use regex (magic mode) and groups `\(..\)` to place the located object string into the replace string. Within the template buffer search strings (**s**) and replace (**r**) are defined with the following syntax:

set-variable *.hookname*.item-list-s*x* ”*regexp*”
set-variable *.hookname*.item-list-r*x* ”*replace*”

Where:

*hookname*
: The name of the file hook i.e. `fhook-c` for ANSI ’C’.

*x*
: The search number, this is valid in the range 1..9, commencing from 1. The search is processed in the order of the numeric identity.

*regexp*
: The regular expression to search for. One of the arguments must include a groups `\(..\)` definition to allow the string to be moved to the replace.

*replace*
: The replace string, this typically includes a *type* and part of the search string.

On invocation of **item-list** the buffer is searched and the results are presented in the `*item-list*` window appearing at the left-hand side of the window. If there is no item list set up for the file type then an error message is displayed.

The user may interact with the `*item-list*` buffer using the mouse or <RETURN>, on selecting a line then the user is moved to the corresponding line in the original buffer.

**item-list-find** finds the current item list item and searches for the text in the original buffer. This is typically bound to a mouse or key stroke action.

**item-list-close** closes the item list buffer.

## EXAMPLE

The following example works through the **item-list** definition for the ME macros e.g. given that the ME macro definition is:

```
define-macro macro-name
```

Searching for "`define-macro \([a-z-]+\)`" and replacing with "`Macro \1`" will work most of the time. The space between `define-macro` and the name does not have to be a single space and the *name* does not have to contain just lower case letters, so these search strings should be a flexible as possible, try:

```
"define-macro\s +\(\w+\)"
```

This however is not as optimal as it could be and if you have large files this could become slow. Performance can be greatly increased if it can be anchored to the start of the line, e.g.

```
"^define-macro\s +\(\w+\)"
```

but to allow for initial white spaces and the optional numeric argument, you really need:

```
"^\s *[0-9]*\s *define-macro\s +\(\w+\)"
```

To hilight the function name you need to encase the name in a magic hilighting string:

```
"\CCbmacro-name\CCA"
```

where `\CC` is a `Ctrl-C` (`\x03`) char, so the replace string should be:

```
"Macro \CCb\1\CCA"
```

Now all thats needed is to set these variables as fhook command variables, for macro files, the file hook command is `fhook-emf`, therefore the following is required:

```
set-variable .fhook-emf.item-list-s1 "^\\s *[0-9]*\\s *define-macro\\s +\\(\\w+\\)"
set-variable .fhook-emf.item-list-r1 "Macro \CCb\\1\CCA"
```

Note that you can have as many of these search and replace variables as you require, i.e. `.item-list-s1`, `.item-list-s2`, `.item-list-s3`, ... ; but the more you have the slower it will be, often a good regex can do the job of 2 or 3.

## SEE ALSO

**occur**(3), **search-forward**(2), Regular Expressions

## NAME

keep-buffer - Rename the current buffer

## SYNOPSIS

**keep-buffer**

## DESCRIPTION

**keep-buffer** renames the current buffer by appending a number to the end of the current buffer name. This ensures that, if the command used to generate the buffer (such as **compile**(3) and **grep**(3)) is rerun, the current buffer will not be deleted.

This command is useful when the current buffer is time consuming to generate and another compile or grep is needed before it has been fully processed.

It can also be used as a quick way to create multiple ishells, by executing this macro on the first `*shell*` buffer, which will rename it to `*shell*1`, executing **ishell** will create a new shell.

## NOTES

**keep-buffer** is a macro defined in `tools.emf`.

## SEE ALSO

**compile**(3), **grep**(3), **ishell**(3).

Ref: keep-buffer(3) File: m3mac102.3 Date: 2026/08/04

## NAME

Mahjongg - MicroEmacs version of the solitaire Mah Jongg game

## SYNOPSIS

**Mahjongg**

## DESCRIPTION

Mah Jongg is an ancient Chinese game usually played by four players with tiles similar to dominos. This is a MicroEmacs version which was inspired by the X-Windows version of the same game. The X-Windows version for the solitaire game originally seen on the PC and later ported to SunView.

### Theory Of Play

The object of the game is to remove all the tiles from the board. Tiles are removed by matching two identical tiles which have either an open left edge or open right edge. The only exception to this rule is that any open "*flower*" tile (bamboo `[BAMB]`, orchid `[ORCH]`, plum `[PLUM]`, or chrysanthemum `[CHRY]`) matches any other open "*flower*" tile and any open "*season*" tile (spring, summer, autumn, or winter) matches any other open "*season*" tile.

Tiles are stacked on the board, the height of the tile is indicated by the color coding as follows:

`Level 5` - White
`Level 4` - Red
`Level 3` - Yellow
`Level 2` - Green
`Level 1` - Cyan

To remove a pair of tiles, click the left mouse button on a tile (which will show in the selection color) and then click the left mouse button on the matching tile. At this point, both tiles will disappear from the board. If after selecting the first tile, you decide that you don't wish to play that tile, simply reclick the left button on the selected tile, alternatively click the right button to deselect any selected tile.

To the right of the board are a number of control buttons. To select an option, click the left mouse button on it.

**NEW** Start a new game (keyboard n).

**SAME** Start the same game again (keyboard s).

**QUIT** Exit the game (keyboard q).

**HELP** This help page (keyboard esc h).

The counter shows the number of remaining tiles on the board, at the start of the game there are 144 tiles.

## NOTES

Mahjongg is a macro defined in `mahjongg.emf`.

Mah Jongg may only be played with a mouse, there is no keyboard support, with the exception of the re-start keys.

## ACKNOWLEDGEMENT

Thanks to Jeff S. Young who (I think) wrote the original X-Windows version, and whose manual page formed the basis of this page.

The tile patterns were inspired from the X-Windows tile patterns. The X-Windows tile patterns themselves are copyright 1988 by Mark A. Holm <tektronix!tessi!exc!markh>.

## SEE ALSO

Games, **Match-It**(3), **Patience**(3).

## NAME

man - UNIX manual page viewer.
man-clean - Clean UNIX manual page.
%man-com - Man Command line

## SYNOPSIS

**man**
**man-clean**
**%man-com**

## DESCRIPTION

**man** provides a mechanism to display a UNIX manual page within the MicroEmacs window. On invoking **man** the user is prompted for the name of the manual page to display:

```
Man on ?
```

The name of the manual page (and any options) are entered on the command line. The macro invokes the UNIX utility **man**(1) to generate the page and displays the results in a window. The man page executable may be defined with the variable **%man-com** this is defined as `man` by default.

Another manual page can be selected by either moving the cursor to the link and pressing return or double clicking on it with the left mouse button. MicroEmacs will then attempt to load and display the selected manual page.

**man-clean** removes any man-page formatting codes from the current buffer reducing a manual page to plain text. The formatting codes are used to create the bold and underline fonts. This allows the page to be treated as a normal buffer, i.e. string searches and other similar command will work as expected.

## NOTES

**man** and **man-clean** are macros defined in `hkman.emf`.
**man** is only made available within UNIX environments, the UNIX start up file `unixterm.emf` links in the macro. If the **man** utility is required on other platforms then the following definition is required in a start-up file.

```
define-macro-file hkman man
```

On Microsoft Windows environments, if the **cygwin** directory is located then **man** is enabled and uses the **cygwin man**(1) utility and manual pages.

## SEE ALSO

**man**(1), **man**(9), **cygwin**(3), **user-setup**(3), **spell-buffer**(3).

Ref: man(3) File: m3mac031.3 Date: 2022/06/12

## NAME

Match-It - MicroEmacs version of the Match-It game

## SYNOPSIS

**Match-It**

## DESCRIPTION

The object of the game is to score the largest number of points, to do this the player must complete as many sheets as possible. A sheet is completed when all the tiles are removed from the board within the given time limit - ALL sheet are possible. If the player fails to remove all the tiles before the time runs out a life is lost, if all lives have been lost then the game is over.

Tiles are removed from the board by matching two identical tiles which have an 'extraction' path between them. The only exception to this rule is that any open "*flower*" tile (bamboo `[BAMB]`, orchid `[ORCH]`, plum `[PLUM]`, or chrysanthemum `[CHRY]`) matches any other open "*flower*" tile and any open "*season*" tile (spring, summer, autumn, or winter) matches any other open "*season*" tile.
An 'extraction' path is a straight line which uses 2 or less right angles, the following are legal extraction paths, '*'s denote the right angles:

```
            A---*       *------*     A----*
A----A          A       AXXXXXA     XXXXX|
                                    A----*
```

The following are illegal paths:

```
*----*          *---*
AXXXX|          |XXXA
XXXXA*      A---*XXXX
```

2 points are added to the score whenever a pair is successfully removed, a point is deducted whenever a pair is selected which can not be removed because there is no valid extraction path. There are 2 aids, pressing the right button on a tile when no other tile is selected will hilight all tiles of matching type, this costs 4 points. The other help is activated by a button at the top right of the screen and it removes a random removable pair (or informs the user that there are no removable pairs), there are a limited number of these helps.

At the end of a successful sheet the score is increased be the time left, the number of lives and helps remaining and by the Pedigree and Internal bonuses if they were achieved.

The Pedigree bonus is obtained when only identical tiles are paired, i.e. no differing flowers or seasons were paired, 50 points are awarded when achieved. Its status is indicated by a 'P' to the left of the 'Help' button and the top of the window.

The internal bonus is obtained when the outer 4 margins are not used. If the left or right margins are not used then 10 points are awarded for each, if the top or bottom are not used then 20 points are awarded for each and if none are used then 400 points are awarded! The status on the Internal bonus is indicated by an '`I`' surrounded by '*'s, one for each margin. This can be found next to the Pedigree bonus '`P`'.

## GAME CONTROLS

To the right of the high score table on the main menu there are a number of control buttons. To select an option, click the left mouse button on it.

**NEW** Start a new game.

**QUIT** Exit Match-It.

**HELP** This help page (keyboard esc h).

During a sheet, to remove a pair of tiles, click the left mouse button on a tile (which will show in the selection color) and then click the left mouse button on the matching tile. At this point, if the tiles can be removed, the extraction path is drawn and both tiles will disappear from the board. If after selecting the first tile, you decide that you don't wish to play that tile, simply reclick the left button on the selected tile, alternatively click the right button to deselect any selected tile.

To the top right of the sheet there are a number of control buttons:

**HELP** Removes a tile pair.

**QUIT** Exit the game.

**BOSS** Hides Match-It, also acts as a pause. Execute Match-It again to return to the game.

The top left shows the number of remaining lives, the current sheet level, the current score, time remaining for the current sheet and the status of the Internal and Pedigree bonuses.

## NOTES

Match-It is a macro defined in `matchit.emf`.

Match-It may only be played with a mouse, there is no keyboard support, with the exception of the re-start keys.

The sheet database file matchit.edf must be accessible for Match-It to work.

## SEE ALSO

Games, **Mahjongg**(3), **Metris**(3).

Ref: Match-It(3) File: m3mac013.3 Date: 2022/06/12

## NAME

Metris - MicroEmacs version of the falling blocks game

## SYNOPSIS

**Metris**

## DESCRIPTION

Traditional falling blocks game, make solid horizontal lines out of the falling blocks. The blocks can be rotated and moved left or right by the user as they fall. Once a horizontal line is completely solid it will disappear and everything above it will drop down. A bonus is given if 3 solid rows are made at the same time, i.e. using one block.

Every line you make the game speeds up until it gets too fast!! The game ends when there is no more room to put a block.

The keys used to control Metris are:

**left** or **j** Move the block left one character.

**right** or **l**
Move the block right one character.

**down** or **k**
Rotate the block counter-clockwise 90 degrees.

**space** Drop the current block.

**p** Pause the current game.

**q** Quit the current game.

**C-l** Redraw the display.

**return** Start a new game.

**esc h** View this help page.

## NOTES

**Metris** is a macro defined in `metris.emf`.

## SEE ALSO

Games, **Match-It**(3), **Patience**(3).

Ref: Metris(3) File: m3mac014.3 Date: 2022/06/12

## NAME

narrow-search-forward - Search for a string in the forward direction
narrow-search-backward - Search for a string in the backward direction

## SYNOPSIS

*n* **narrow-search-forward** "*string*" (**C-c s**)
*n* **narrow-search-backward** "*string*" (**C-c r**)

## DESCRIPTION

**narrow-search-forward** searches for a string from the current cursor position to the end of the file and will open up and search through **narrow**(2m) regions to search for the string. This differs from the **search-forward**(2) command that does not search through narrow regions.

The string is typed on the bottom line of the screen, and terminated with the <ESC> key. Special characters can be typed in by preceding them with a ^Q. A single ^Q indicates a null string. On successive searches, hitting <ESC> alone causes the last search string to be reused.

Searching is affected by **magic**(2m) mode, which allows regular expression pattern matching, and **exact**(2m) mode which makes the search case sensitive.

The numeric argument *n* is interpreted as follows:

**n > 0** The *n*th occurrence of the *string* is located.

**n < 0** The first occurrence of the *string* is located in the next *n* lines.

**narrow-search-backward** searches backwards in the file. In all other ways it is like **narrow-search-forward**.

## DIAGNOSTICS

The command returns a status of FALSE if the *string* could not be located (or *n*th *string* where *n* occurrences are requested). If the *string* is found within the given search criteria the return status is TRUE.

## SEE ALSO

**buffer-mode**(2), **exact**(2m), **hunt-backward**(2), **hunt-forward**(2), **isearch-forward**(2), **magic**(2m), **replace-string**(2). **search-forward**(2),
Regular Expressions

## NAME

normal-tab - Insert a normal tab

## SYNOPSIS

*n* **normal-tab**

## DESCRIPTION

**normal-tab** insert a tab into the current buffer by temporarily disabling any auto indentation schemes. The macro first disables any indentation rules by setting **$buffer-indent**(5) to 0 the command **tab**(2) is then called with the given argument *n*. This means that the buffer's **tab**(2m) mode setting will be respected, i.e. whether a tab character or spaces are inserted. The initial indentation rules are restored on exit.

## NOTES

**normal-tab** is a macro implemented in `format.emf`.

## SEE ALSO

**tab**(2), **insert-tab**(2), **tab**(2m).

Ref: normal-tab(3) File: m3mac054.3 Date: 2022/06/12

## NAME

notes - Tagged hierarchical database of notes.

## SYNOPSIS

**notes**
**notes-file**
**notes-close**
**notes-context**
**notes-instant**

## DESCRIPTION

**notes** is a note database that allows miscellaneous text information to be organized and retained in a database. Each note is has a label associated with it which may be hierarchically organized into a tree of notes.

Information that a programmer would typically put into a notes database may include:

- How to mount a CD-ROM on a remote UNIX system.
- Boot strapping Linux.
- Configuring the C compiler.
- etc..

This information would be hierarchically organized by system, function etc. The content typically represents a snippet of information from a diverse range of subjects that probably needs to be accessed infrequently but organized in such a way that it may be located quickly when required.

On running **notes** then a new frame is created (retaining the existing window configuration) and the user is prompted to create a new notes file with their user name, accept this option and a new notes file is created in the JASSPA home directory. The user is presented with a split view window as follows:

```
+-------------+-----------------------------------+
| + user Pad  |Text associated with the note.     |
|*  +- info 1 |                                   |
|   +- info 2 |Anything may be typed in here.     |
|             |                                   |
|             |                                   |
|             |                                   |
|             |                                   |
+=User Pad====+=*User Pad: info 1* ===============+
|                                                 |
+-------------------------------------------------+
```

The left pane comprises a tree of note titles, the right pane contains the text note body of the selected note title. Free form text may be entered into the text body pane.

Notes are add and removed from the notes tree pane. A mouse right click presents a menu of options that operate on the current node as follows:

**Tidy Tree** - `t/return`

At the current node then close the note hierarchy. `t` will close the node a RETURN at the node will open or close it, depending on the current state.

**Rename Note** - `r`
Edit the name of the note, a dialog is presented and a free form label of the note may be edited.

**Delete Note** - `delete`
Delete the note from the tree. This deletes the label and the note. if the note has any children then all children are deleted.

**Cut Note** - `x`
Delete a note, including any child notes from the tree, allowing the cut information to be pasted back to a different location in the tree. (See **Paste Note**).

**Paste Note** - `v`
Inserts a cut note (or note tree) back into the list at the current position.

**Move Note Up** - `u`
Move a note up the tree hierarchy, at the same level.

**Move Note Down** - `d`
Move a note down the tree hierarchy, at the same level.

**Insert New Note** - `insert`
Insert a new note as a child of the current node. The user is prompted for the name of the note, the name may be entered and the cursor is left in the body pane for text entry.

**Forward Note** - `n`
Moves to the next note. Within the note window then **C-page-down** similarly advances to the next note.

**Backward Note** - `p`
Moves to the previous note. Within the note window then **C-page-up** similarly advances to the previous note.

**Search Forward** - `esc s`
Search in a forwards direction from the current note.

**Search Backward** - `esc C-s`
Search in a backwards direction from the current note.

**Open Notes File**
Opens a new notes file, the user is explicitly prompted for the name of the notes file to open.

**Save Notes File**
Saves the notes file to the file system.

**Hide Notes**
Hides the notes window and restores the exiting buffer windows prior to the **notes** invocation.

To quit **notes** then delete the buffer using **C-x C-k** or run the command **notes-close** command.

The notes file may be saved like any other buffer using **save-buffer**(2) (`C-x C-s`). This saves the tree and the notes. Auto-save is automatically enabled.

**notes-close** closes the notes frame and returns the user to the existing windows, retaining the same screen layout. The notes file is still resident in the editor and may be restored with a **notes** command.

**notes-file** opens a new notes file, the user is explicitly prompted for the name of the notes file. There are no restrictions on the number of notes files that may be created. **notes-import-skwyrul** imports a Skwyrul index file (`skwyrul_l.htm`) into MicroEmacs notes format. This command is not available on line as it is likely to be used very infrequently. To load the command then load the file `notesutl.emf` and execute the buffer **execute-buffer**(2) (`esc x execute-buffer`). The command may be explicitly run using `esc x notes-import-skwyrul`.

**notes-instant** creates a new note under the *Instant Notes* folder, it is designed to enable the user to jot something down quickly and in a safe place. The note is automatically named and is the fastest method of adding a note. This may be invoked outside of the notes window, on invocation then a new notes node is created and the content of the note added.

**notes-context** is similar to **notes-instant** but a bit more advanced and allows a note to be created that is associated with a buffer. When working on a file (source code, document etc.). It is sometimes useful to be able to associate some information with that file, typically this is done with something like a `TODO:` comment, however this is not always appropriate. **notes-context** attempts to address this, it works from the current buffers filename path and attempts to locate an existing note for it, if no note is found then a new one is created. When locating a note it looks down the path. The command may be invoked outside of the notes window, on invocation then a new notes node is created and the content of the note added.

### Text Hilighting

Blocks of text within the notes body may be highlighted. The special constructs of `>format>` ... `>end>` located on the left margin introduce and close a highlighting style. i.e.

```
>c>
/* This is a C code insert */
>end>

>perl>
# This is a perl insert
>end>
```

The keywords that are supported are:

`>emf>` - MicroEmacs macro files.
`>pseudo>` - Pseudo code.
`>bnf>` - BNF.
`>c>` - C programming language
`>cpp>` - C++ programming language
`>perl>` - Perl programming language
`>tcl>` - TCL programming language
`>shell>` - UNIX shell scripts
`>make>` - Makefile scripts
`>end>` - Ends the highlighting block.

### NOTES

**notes** is a macro implemented in `notes.emf`. The database information is stored in a single file *user***.enf**. The file comprises both the index and the text.

The implementation of **notes** uses **narrow-buffer**(2) to hide information for each note page. The Registry is used to store and display the note tree labels.

## SEE ALSO

**osd-search**(3).

Ref: notes(3) File: m3mac078.3 Date: 2022/06/12

## NAME

number-items - Renumber a vertical list of items

## SYNOPSIS

*n* **number-items** *start-number char-width num-lines*

## DESCRIPTION

**number-items** provides a short cut method of re-numbering a list of vertically aligned numbered items. The command assumes a fixed width number per line. The command operates from the current cursor position in a downward direction. The parameters are defined as follows:

*start-number*
: The value of the first item to renumber.

*char-width*
: The width of the number space i.e. 1-10 is a width of 2, 89-101 is a width of 3.

*num-lines*
: The number of lines to re-number, including the current line.

  If a numeric argument of `0` is given the inserted numbers are all prefixed with 0's to ensure each is at least *char-width* characters long.

## EXAMPLE

The following code fragment shows how a list is renumbered.

```
MY_COMMAND (myCMD_UP,    # 9, "Move Up")
MY_COMMAND (myCMD_DOWN,   10, "Move down")
MY_COMMAND (myCMD_LEFT,   15, "Move down")
MY_COMMAND (myCMD_RIGHT, 10, "Move down")
```

To to use the command to renumber the list from 100 to 103 then the cursor is placed at the left hand side where the number should start. In the example this is `#`, then issue the command **esc x number-items 100 3 4**. This renumbers the list resulting in the following:

```
MY_COMMAND (myCMD_UP,    100, "Move Up")
MY_COMMAND (myCMD_DOWN, 101, "Move down")
MY_COMMAND (myCMD_LEFT, 102, "Move down")
MY_COMMAND (myCMD_RIGHT,103, "Move down")
```

## NOTES

**number-items** is a macro defined in `misc.emf`.

## SEE ALSO

Ref: number-items(3) File: m3mac092.3 Date: 2025/03/16

## NAME

## SYNOPSIS

*n* **occur** "*search-string*"

## DESCRIPTION

**occur** performs a search for a string in the current buffer; generating a list of every occurrence found. If the *search-string* has not been given via the command-line and the buffer has an active region when the command is called, the content of the region will be used as the search string, otherwise the string will be prompted for.

The buffer's **magic**(2m) mode setting is used to determine whether the *search-string* is a the regular expression or not, except with the current region is used.

The results are presented in the `*item-list*` window, the user may interact with the `*item-list*` buffer using the mouse or <RETURN>, on selecting a line then the user is moved to the corresponding line in the original buffer.

If an argument **n** is given then the width of the match sample string is limited to 15 or 28 chars, with a value of 1 or 2, on each side of the match making the output narrower; this can be used to allow the buffer to be placed in a side window such as a toolbar.

## NOTES

The `*item-list*` window may be closed with the command **item-list-close**(3) typically bound to `esc-F7`.

**occur** is a macro defined in `itemlist.emf`.

## SEE ALSO

**item-list**(3), **item-list-close**(3), **search-forward**(2), **magic**(2m), Regular Expressions

## NAME

open-line - Open new line at cursor position

## SYNOPSIS

*n* **open-line** (**C-o**)

## DESCRIPTION

**open-line** inserts *n* new lines after the current cursor position Any text following the cursor is moved to the newly created line(s). The cursor position is maintained by the command.

## SEE ALSO

**insert-newline**(2), **newline**(2).

Ref: open-line(3) File: m3mac071.3 Date: 2022/06/12

## NAME

osd-dialog - OSD dialog box
osd-xdialog - OSD Extended dialog box
osd-entry - OSD entry dialog box

## SYNOPSIS

*n* **osd-dialog** "*title*" "*prompt*" ["*x-pos*" "*y-pos*"] "*but1*"
*n* **osd-xdialog** "*title*" "*prompt*" *default* ["*x-pos*" "*y-pos*"] "*but1*" "*but2*" ...
*n* **osd-entry** "*title*" "*prompt*" *variable* ["*x-pos*" "*y-pos*"] [["*entry-xsize*"—"*entry-xsize*x*entry-ysize*"] ["*type*"]]

## DESCRIPTION

**osd-dialog** constructs a OSD dialog prompt with a title string *title*, a prompt string within the dialog of *prompt*. A single button, with text rendering *but1*, is placed within the dialog. The dialog remains on the screen until the button is selected or the user aborts.

**osd-xdialog** creates an extended dialog with multiple buttons similar to **osd-dialog**, the number of buttons created (*#*) is determined from the number of *but* arguments. The *default* integer argument specifies the default button (1..*#*), a value of 0 specifies that there is no default button.

The commands return the button pressed in the variable **$result**(5).

**osd-entry** constructs a simple OSD entry dialog which prompts the user to type in a value. The value of the supplied variable is used as an initial entry value, the variable is set to the entered value when the user presses the "`Okay`" button but remains unchanged if the user Cancel or aborts.

The size of the entry defaults to 30 characters if not specified by the user, when a size parameter is given it can take one of two forms, either simply "`w`" specifying the width, the height defaulting to 1, or "`w`**x**`h`" (i.e. "`40x5`") specifying both. The last optional argument *type* sets the type of value being entered (e.g. file name, buffer name, etc) see flag **h** on the help page for **@ml**(4) for a list of entry types and the numerical value to be supplied.

The argument *n* can be used to change the default behavior of the commands described above, *n* is a bit based flag where:

**0x01** Enables command abort (default), except **osd-entry** which ignores the setting of this bit. When enabled, if the user abort by either closing the dialog (top right button) or using the **abort-command** the dialog command will also abort. If bit 0x01 is not set the command will not abort and **$result** will be set to -1.

**0x02** When set, flags that a dialog position has also been provided, extra arguments **x-pos** and **y-pos** must also be given. By default the dialog is placed under the mouse.

## EXAMPLE

A simple query dialog is typically constructed using **osd-dialog**, as follows:

```
!if &seq %osd-search-str ""
    osd-dialog "Replace" "Error: Search string is empty!" "  &OK  "
    !return
!endif
```

The following example uses multiple buttons within a single dialog, using **osd-xdialog**, as follows:

```
0 define-macro osd-close
    !if &bmod "edit"
        set-variable #l0 &spr "Buffer \"%s\" changed" $buffer-bname
        osd-xdialog "Buffer Close" #l0 1 "&Save First" \
                                         "&Loose Changes" "&Cancel"
        !if &equ $result 3
            !abort
        !elif &equ $result 2
            -1 buffer-mode "edit"
        !else
            !if &seq $buffer-fname ""
                !nma write-buffer
            !else
                !nma save-buffer
            !endif
        !endif
    !endif
    delete-buffer $buffer-bname @mna
!emacro
```

The next example macro can be used to change the value of a user variable to a user supplied file name:

```
set-variable %source-root "~/"

define-macro set-source-root
    osd-entry "Source Root" "&Path : " %source-root 35 1
!emacro
```

## NOTES

**osd-dialog**, **osd-xdialog** and **osd-entry** are macros defined in `osd.emf`, using **osd**(2) to create the dialog.

## SEE ALSO

**$result**(5), **osd**(2).

Ref: osd-dialog(3) File: m3mac037.3 Date: 2026/02/17

## NAME

page-file - Load a large file in page mode.
page-bfile - Load a large binary file as binary data.
page-file-save-buffer - Save contents of a page mode buffer to file.
page-file-beginning-of-buffer - Move to the start of the paged file.
page-file-end-of-buffer - Move to the end of the paged file.
page-file-goto-byte - Page to the buffer byte offset.
page-file-goto-page - Move to page.
page-file-scroll-down - Scroll paged buffer down.
page-file-scroll-up - Scroll paged buffer up.
page-file-forward-line - Move forwards one line in paged buffer.
page-file-backward-line - Move backwards one line in paged buffer.
page-file-search-buffer - Search the buffer for the pattern.
page-file-search-forward - Search for a string in the forward direction.
page-file-search-backward - Search for a string in the backward direction.
page-file-isearch-forward - Search forward incrementally (interactive).
page-file-isearch-backward - Search backwards incrementally (interactive)

## SYNOPSIS

*n* **page-file** "*file-name*"
**page-bfile** *file-name*

**page-file-save-buffer** (**C-x C-s**)

**page-file-beginning-of-buffer**
**page-file-end-of-buffer**
**page-file-goto-byte** *byte-offset*
**page-file-goto-page** *page-offset*

*n* **page-file-scroll-down** (**C-n**)
*n* **page-file-scroll-up** (**C-p**)

*n* **page-file-forward-line** (**C-n**)
*n* **page-file-backward-line** (**C-p**)

**page-file-search-buffer** *option pattern*
**page-file-search-forward** (**C-x s**)
**page-file-search-backward** (**C-x r**)
**page-file-isearch-forward** (**C-s**)
**page-file-isearch-backward** (**C-r**)

## DESCRIPTION

The **page-file** commands allow large files, up to 2GB in size, to be viewed and edited. **page-file** operates by loading a small part (or *page*) of the file into a buffer for viewing and editing hence the whole file is not loaded into memory. The file may be viewed and edited while in a page mode, MicroEmacs automatically manages the page swapping and updates to the file.

While a file is being paged then a replacement set of commands are used to navigate through the buffer. These replacement commands automatically load the next page into the buffer, discarding or saving any edits from existing pages.

### File Loading

**page-file** finds the named file *file-name*. If it is already in a buffer, make that buffer active in the current window, otherwise attempt to create a new paged buffer and read the file into it. Syntax hilighting is available following any file loading. **page-file** is the paged equivalent of **find-file**(2).

The numeric argument *n* is used to modify the default behavior of the command, where the bits are defined as follows:

**0x01** If the file does not exist and this bit is not set the command fails at this point.

**0x02** If this bit is set the file will be loaded with **binary**(2m) mode enabled. See help on **binary** mode for more information on editing binary data files.

**page-bfile** loads a file as binary data, providing binary editing capabilities. The command operate as command **page-file**(3) except that it loads the given file with **binary**(2m) mode set.

### Saving

**page-file-save-buffer** saves the contents of the current buffer if the contents have been changed, writing the buffer back to the file it was read from. This command is the same as **save-buffer**(2) for a paged file.

### Navigation

**page-file-beginning-of-buffer** moves to the beginning of the buffer. If any edits have been performed at the current viewing position that need to be saved then the user is prompted to save the changes and the new pages are loaded from the file at the beginning of the buffer.

**page-file-end-of-buffer** moves to the end of the buffer in a similar way to **page-file-beginning-of-buffer**.

**page-file-goto-byte** advances the buffer to the byte offset specified by *byte-offset*. If any edits have been performed at the current viewing position that need to be saved then the user is prompted to save the changes and the new pages are loaded from the file at the specified file byte offset.

**page-file-goto-page** advances the buffer to the page offset specified by *byte-offset*. The behavior is the same as **page-file-goto-byte**.

**page-file-scroll-up** and **page-file-scroll-down** moves the window in the current buffer down by *n* lines, the default when *n* is omitted is 1 windows worth of lines i.e. a next page operation. The user is prompted to save changes if a file paging operation is performed.

**page-file-forward-line** moves the cursor down *n* lines, default 1. If the line is not on the current screen then display the next page and move to the line. Similarly **page-file-backward-line** moves the cursor up *n* lines, if the line is not on the current screen then display the previous page and move to the line. The user is prompted to save changes if a file paging operation is performed.

### Searching

**page-file-searching-buffer** is designed to be called from a macro and searches for a string in the buffer. The *option* specifies the search direction.

**page-file-search-forward** and **page-file-search-backward** searches for a string from the current cursor position to the end/start of the file and is simular to **search-forward**(2) and **search-**

**backward**(2), respectively. If the search string is not found in the current buffer then the search continues by paging the file until the start/end of the file is reached. The user is prompted to save changes if a file paging operation is performed.

**page-file-isearch-forward** and **page-file-isearch-backward** perform and incremental forward and backward search in the paged file and are similar to **isearch-forward**(2) and **isearch-backward**(2), respectively. If the search string is not found in the current buffer then the search continues by paging the file until the start/end of the file is reached. The user is prompted to page and save changes if a file paging operation is performed.

## NOTES

The **page-file** macros are implemented in macro file `pagefile.emf`. The paging is managed in macro space and uses **insert-file**(2) to perform the read.

## SEE ALSO

**binary**(2m), **find-bfile**(3), **find-file**(2), **isearch-backward**(2), **isearch-forward**(2), **save-buffer**(2), **search-backward**(2), **search-forward**(2).

Ref: page-file(3) File: m3mac065.3 Date: 2022/06/12

## NAME

paragraph-to-line - Convert a paragraph to a single line

## SYNOPSIS

*n* **paragraph-to-line**

## DESCRIPTION

**paragraph-to-line** is a variation of **fill-paragraph**(2). **paragraph-to-line** reduces each of the next *n* paragraphs of text to single lines. This command is typically used to prepare text for import into a word processor such as **Microsoft Word** or **Word Perfect**. Reduction of text to a single line allows the word processor to import the raw text file and keep the text within paragraph blocks. If the text is not prepared then all of the line-feeds have to be manually deleted.

**paragraph-to-line** allows text based documents to be prepared in MicroEmacs and imported into the word processor at the final stage for formatting and layout.

## NOTES

**paragraph-to-line** is a macro defined in `format.emf`.

## SEE ALSO

**fill-paragraph**(2).

Ref: paragraph-to-line(3) File: m3mac003.3 Date: 2025/08/28

## NAME

Patience - MicroEmacs version of Patience (or Solitaire)

## SYNOPSIS

**Patience**

## DESCRIPTION

Patience (or Solitaire) is a solitaire game using a standard set of playing cards. The object of the game is to use all of the cards in the deck to build up four suit stacks from Ace to King.

The board is laid out with the dealer pile at the top right hand corner, to the left are four suit stacks onto which cards of the same suit are placed, in ascending order from the Ace. Below these two areas of the board are seven row stacks, organized in a triangular shape with zero to six downward facing cards.

Cards may be moved around the playing area by stacking alternative red and black cards in descending order on the row stacks. When a row stack has no upturned cards on the stack then the top card may be turned over and may be played. If a stack becomes empty then only a King may be moved into the vacant position. Cards may be removed from the dealer, they are over-turned in sets of three cards, the underlying 2 cards are visible, but are not accessible, only the top card may be removed and played from the dealer.

Cards are moved around the board using the mouse. Cards may be moved from the dealer or between the row stacks by placing the mouse over the card to be moved and pressing the left mouse button. Move the cursor to the new card position and release the left mouse button. If the move is legal then the card(s) are moved to the new stack. Multiple cards may be moved from the row stacks, the appropriate card(s) to be moved is automatically determined.

Cards may be moved onto the suit stacks by a single left mouse press and release on the same card, the card is moved to the appropriate suit stack. The same technique is used to turn cards over in the suit stacks, and to deal the next set of cards by the dealer. To deal, then click on the down-turned card stack, if there are no further cards at the dealer then click on the empty position and the dealer will turn over the dealer stack and deal from the top again.

Note that once a card is played onto the suit stacks then it cannot be removed.

To the right of the board are a number of control buttons. To select an option, click the left mouse button on it, the buttons are labeled:

**DEAL** Start a new game by dealing new cards.

**QUIT** Exit the game

**HELP** This help page

Note that the screen may be updated at any time using "*C-l*".

## NOTES

**Patience** is a macro defined in `patience.emf`.

The game is best played with a mouse, it is possible to play with the keyboard, as follows:

"*esc h*" for help

To move a card between stacks enter the source and destination column number (”*1*”,”*2*”,..”*7*”). To move from the dealer pile then the source is the ”*space*” key.

”*tab*” deals the next cards.

To overturn a card on the row stacks then enter the card column twice i.e. source and destination are the same.

To move a card from the row to the suit stacks then either enter the card column twice, or enter the destination as ”*h*”,”*d*”,”*c*”,”*s*” (i.e. ”*2 2*” or ”*2 s*” to move the card in column 2 to the spades stack).

”*C-c C-c*” to deal the cards again.

”*C-l*” redraw the screen.

”*q*” to quit the game.

## SEE ALSO

Games, **Triangle**(3), **Mahjongg**(3).

Ref: Patience(3) File: m3mac015.3 Date: 2022/06/12

## NAME

perldb - Perl Debugger

## SYNOPSIS

*n* **perldb** [ "*script-name*" ] "*script-args*"

## DESCRIPTION

**perldb** provides an editor interface to the Perl debugger, on running the command an interactive shell window is opened to the debugger command line interface. MicroEmacs then interprets the information from the debugger interface and opens files and hilights the current line as required. The current line is maintained while single stepping through the script.

Buffers opened and referenced by the debugger have the key `F9` bound to setting a break point, this only works if the buffer contains the current execution point.

If an argument *n* of 2 is given to **perldb** the command assumes that the current buffer is the script file to debug, the "`script-name`" argument is not prompted for.

## NOTES

**perldb** is a macro defined in file `hkipipe.emf`.

## BUGS

The '**R**' rerun command does not work correctly on Windows platforms, perldb is rerun in a newly created external dos command window instead of inside the MicroEmacs ipipe buffer.

## SEE ALSO

**gdb**(3), **ipipe-shell-command**(2).

Ref: perldb(3) File: m3mac055.3 Date: 2022/06/12

## NAME

query-replace-all-string - Query replace string in a list of files

## SYNOPSIS

*n* **query-replace-all-string** "*from*" "*to*" "*files*" ["*grep-from*"]

## DESCRIPTION

**query-replace-all-string**, similar to **query-replace-string**(2), replaces all occurrences of "*from*" to "*to*" in the given list of files prompting the user before replacing each occurrence.

The command finds all occurrences of "*from*" by calling the command **grep**(3) to search for string "*from*" in files "*files*". Thus all relevant edited files must be saved or **grep** may return the wrong line numbers. This is achieved by a call to **save-some-buffers**(2) which prompts the user to save any changed buffers one at a time.
Each occurrence of "*from*" is jumped to using **get-next-line**(2) and the string is replaced by the call:

```
-1 query-replace-string "from" "to"
```

This query-replaces all occurrences of "*from*" to "*to*" on the current line only, hence the line numbers must be correct. This also means that the "*from*" search string must be correctly formatted for both grep and query-replace-string, unless bit 0x02 is set (see below).

The given argument *n* is a bit based flag which changes the default behavior described above. The bits have the following effect:

**0x01** Prompt before saving any changed buffer, enabled by default. If this bit is not set then any changed buffer is automatically saved before the **grep** is performed.

**0x02** If set then a fourth argument "*grep-from*" must also be given. This string is used in place of the "*from*" string for the **grep** only.

## NOTES

**query-replace-all-string** is a macro defined in `search.emf`.

The **grep** command must be working before this command can function properly.

It is not recommended to use a "from" or "to" string which uses more that one line as the results may be unpredictable.

As the change is likely to be over several files a single call to **undo**(2) at the end of execution will not undo all the changes made. To undo all the changes made, use **get-next-line**(2) to loop through all the occurrences and call **undo** for each occurrence

## SEE ALSO

**query-replace-string**(2), **save-some-buffers**(2), **grep**(3), **get-next-line**(2), **undo**(2), **replace-all-string**(3), **search-forward**(2).
Regular Expressions

Ref: query-replace-all-string(3) File: m3mac007.3 Date: 2022/06/12

## NAME

read-session - Read in session information

## SYNOPSIS

*n* **read-session** [ "*session-file*" ]

## DESCRIPTION

**read-session** reads in a MicroEmacs session file, setting the current session information. If argument **n** is not given then the given "*session-file*" is simply read in. If a non-zero argument is specified then default session is set to the given file-name and the file is read. If an argument of zero is given then the default session is re-read. Information read in (and saved) from the history file includes:

- Searching and replacing history.
- Buffer name history.
- Command name history.
- File name history.
- General (all the rest) history.
- Buffer and file list with line numbers.

MicroEmacs's environment may be retained almost intact by the use of the default session and using the **-c** (continue) command-line option to re-load all files that were being edited in a previous session.

On loading a new session then the user is prompted to save the current session. If the numeric argument is zero (bit 0 is 0) then the user is not prompted and the *session-file* is effectively loaded and merged into the current session.

## NOTES

When running multiple MicroEmacs sessions on the same work-station (or different workstations sharing the same home directory), the default history is saved when MicroEmacs exits. As a result the last MicroEmacs sessions that terminates writes the history information used next time.

The session information is saved in a registry format file (see **esf**(8)). Reference should be made to the notes included in **erf**(8) as to how the history file may be edited and effected in the same MicroEmacs session.

The user may have multiple sessions, this is useful when one is working on different projects, the sessions are created using **save-session**(3) using a new session name or by using the **-c** command line option. Starting the editor with a different session means that the hilighting scheme may be different between different sessions i.e. "*Midnight X*" and "*Sherwood Forest*". Note that the color scheme should be changed with **session-setup**(3) or **change-scheme**(3) and not with **user-setup**(3) if different sessions are to use different color schemes.

### Command Line

Reading a session is generally performed when a MicroEmacs session is started using the "**-c**" command line option rather than from the emacs mode line or main menu, see **me**(1). Invoking

**me -c**<*mySessionName*>

where <*mySessionName*> is the name of the session (the **$MENAME**(5) or system login name) will restore all of of the buffers, windows layout, point and mark positions, main window size etc. at the point of the last shut down. e.g. given a system login name of *jasspa* then

**me -c***jasspa*

restores the session to the state when *jasspa* last exited MicroEmacs. When the session name is specified then no other files should be specified, so the command line

**me -c***jasspa* `morefiles.txt`

restores the buffers but does not restore the window layout because this has been destroyed by the introduction of the new file "`morefiles.txt`". This type of command with another file behaves as the old "`me -c`" command prior to the 2006 release that did not support sessions. If <*mySessionName*> is not specified then the buffers are restored but the window layout is not restored.

### SEE ALSO

**save-session**(3), **change-scheme**(3), **erf**(8), **esf**(8), **@h0**(4), **me**(1), **$MENAME**(5).

Ref: read-session(3) File: m3mac081.3 Date: 2026/08/03

## NAME

replace-all-pairs - Replace string pairs in a list of files

## SYNOPSIS

*n* **replace-all-pairs** "*files*"

## DESCRIPTION

**replace-all-pairs** uses the current buffer to extract "*from*" and "*to*" pairs and then replaces all occurrences of "*from*" to "*to*" in the given list of files without prompting the user. An optional third argument "*grep*" can be given which will be used as the grep string, if not given the "*from*" string is used. The format of the current buffer must be:

```
/from1/to1/
Xfrom2Xto2X
?from3?to3?
/from4/to4/grep4/
  .
  .
/fromN/toN/
```

For each pair the command finds all occurrences of "*from*" (or "*grep*" if specified) by calling the command **grep**(3) to search for string "*from*" in files "*files*". Thus all relevant edited files must be saved or **grep** may return the wrong line numbers. This is achieved by a call to **save-some-buffers**(2) between each replace pair, it is called with an argument of 0 to ensure that any changed buffers are automatically saved.
Each occurrence of "*from*" is jumped to using **get-next-line**(2) and the string is replaced by the call:

```
-1 replace-string "from" "to"
```

This replaces all occurrences of "*from*" to "*to*" on the current line only, hence the line numbers must be correct. This also means that the "*from*" search string must be correctly formatted for both grep and replace-string.

The given argument *n* is a bit based flag which changes the default behavior described above. The bits have the following effect:

**0x01** Prompt before saving any changed buffers FIRST time ONLY, enabled by default. If set then the user is also prompted to continue before any changes are made. If this bit is not set then the command executes without any user input.

## NOTES

**replace-all-pairs** is a macro defined in `search.emf`.

The **grep** command must be working before this command can function properly.

It is not recommended to use a "from" or "to" string which uses more that one line as the results may be unpredictable.

As the change is likely to be several pair strings with each changed buffer being saved between pairs **undo**(2) cannot be used to undo the changes. Neither can the backups be relied on as a buffer may be saved more than once in this process, therefore it is strongly recommend that a backup of the files is made before commencing with this command.

## SEE ALSO

**replace-all-string**(3), **replace-string**(2), **save-some-buffers**(2), **grep**(3), **get-next-line**(2), **undo**(2), **query-replace-all-string**(3), **search-forward**(2).
Regular Expressions

Ref: replace-all-pairs(3) File: m3mac020.3 Date: 2022/06/12

## NAME

replace-all-string - Replace string with new string in a list of files

## SYNOPSIS

*n* **replace-all-string** "*from*" "*to*" "*files*" ["*grep-from*"]

## DESCRIPTION

**replace-all-string**, similar to **replace-string**(2), replaces all occurrences of "*from*" to "*to*" in the given list of files without prompting the user.

The command finds all occurrences of "*from*" by calling the command **grep**(3) to search for string "*from*" in files "*files*". Thus all relevant edited files must be saved or **grep** may return the wrong line numbers. This is achieved by a call to **save-some-buffers**(2) which prompts the user to save any changed buffers one at a time.
Each occurrence of "*from*" is jumped to using **get-next-line**(2) and the string is replaced by the call:

```
-1 replace-string "from" "to"
```

This replaces all occurrences of "*from*" to "*to*" on the current line only, hence the line numbers must be correct. This also means that the "*from*" search string must be correctly formatted for both grep and replace-string, unless bit 0x02 is set (see below).

The given argument *n* is a bit based flag which changes the default behavior described above. The bits have the following effect:

**0x01** Prompt before saving any changed buffer, enabled by default. If this bit is not set then any changed buffer is automatically saved before the **grep** is performed.

**0x02** If set then a fourth argument "*grep-from*" must also be given. This string is used in place of the "*from*" string for the **grep** only.

## NOTES

**replace-all-string** is a macro defined in `search.emf`.

The **grep** command must be working before this command can function properly.

It is not recommended to use a "from" or "to" string which uses more that one line as the results may be unpredictable.

As the change is likely to be over several files a single call to **undo**(2) at the end of execution will not undo all the changes made. To undo all the changes made, use **get-next-line**(2) to loop through all the occurrences and call **undo** for each occurrence

## SEE ALSO

**replace-string**(2), **save-some-buffers**(2), **grep**(3), **get-next-line**(2), **undo**(2), **query-replace-all-string**(3), **replace-all-pairs**(3), **search-forward**(2).

Ref: replace-all-string(3) File: m3mac006.3 Date: 2022/06/12

## NAME

replace-string-in-files - Search and replace strings across files

## SYNOPSIS

**replace-string-in-files** "*from-pattern*" "*to-pattern*"

"[*find-options*] <*file-mask*> [*file-mask*] ..." ["*starting-path*"]

**query-replace-string-in-files** "*from-pattern*" "*to-pattern*"

"[*find-options*] <*file-mask*> [*file-mask*] ..." ["*starting-path*"]

## DESCRIPTION

**replace-string-in-files** performs a search for a *from-pattern* and replaces the string with *to-pattern* in all files that match *file-mask* in the current directory (or from *starting-path*) using built in MicroEmacs commands rather than relying on any external executables such as **grep**(1).

**query-replace-string-in-files** is identical to **replace-string-in-files** except the user is prompted to accept each potential replacement.

*find-options*

Defines the *options* that are given to the underlying call to the **grep**(3) command, the default options may also be defined with the **find-setup**(3) command invoked from the command line or from the the **Tools** menu. The *find-options* are defined as follows:

**–** Last option, the following string must be the first *file-mask* which may start with a '-'.

**-a** Re-read all out of date buffers, prevents a manual prompt, equivalent of Yes to all. This is the inverse of **-o**

**-b** Ignore binary files.

**-B** Do not ignore binary files (not advisable as this is a text based search and replace operation).

**-c** Command return the number of matchs found/replaced rather than the default, which is simply 0 for one or more matches or 1 if no matches were found/replaced.

**-C** Command returns 0 (non-error for command-line tool) regardless of if it finds any matches.

**-g** File masks are regex (i.e. .*.emf)

**-G** File masks are not regex (i.e. *.emf)

**-i** Ignore search pattern case (case insensitive).

**-I** Do not ignore search pattern case (case sensitive).

**-l** Use existing buffer of loaded files.

**-L** Do not use existing buffer of loaded files.

**-m** Search pattern is a regular expression.

**-M** Search pattern is not a regular expression.

**-o** Do not re-read all out of date buffers, prevents a manual prompt, equivalent of No to all. This is the inverse of **-a**.

**-p** Starting path specified

**-P** Starting path not specified

**-q** Enables quiet mode for reduced print-out.

**-r** Recurse into sub-directories

**-R** Do not recurse into sub-directories

**-S** <*size*>
File size - less than or equal, the following size formats are supported:

**-S** <*number*> File size specified bytes. **-S** <*number*>**k** File size specified in Kilobytes.

**-s** <*size*>
File size greater than; supported size formats are the same as **-S**.

**-T** <*time*>
Modification time - older than, the following time formats are supported:

**-T** *yy-mm-dd* **-T** *yyyy-mm-dd* **-T** *yyyy-mm-dd hh:mm:ss* **-T** *days* Offset time in days from midnight this morning, e.g. **-T 0** would match all files not modified today. **-T** *days.hours[.minutes[.seconds]]* Offset time in , e.g. **-T 1.1.30** matches all files 25 hours and 30 minutes old.

Note that **-T 1** is not the same as **-T 1.0**, except at midnight.

**-t** <*time*>
Modification time - newer than or equal, supported time formats are the same as **-T**.

**-u** When used with **-w**, allows the file size limit of 100MB to be disabled and strings are searched in files of any size.

**-U** When used with **-U**, files are limited in size 100MB otherwise they will not be searched.

**-w** <*search-pattern*>
File must contain a string matching the given *search-pattern*.

**-W** <*search-pattern*>
File must not contain a string matching the given *search-pattern*.

**-x** Use MicroEmacs **$file-ignore**(5) to exclude unimportant files

**-X** Do not use MicroEmacs **$file-ignore**(5).

**-&** Launch command using a separate process, this stops the current MicroEmacs hanging during the replace process which can be useful for time consuming searches, not supported by **query-replace-string-in-files**.

*file-mask*
The files to search for e.g. ***.txt**

*starting-path*
Specifies the directory location where the search should commence when the **-p** option is used, the default when **-p** is omitted is the location of the current buffer.

## NOTES

- The location of the find options in the 3rd argument may seem odd, but the command is not required to have a single <*options*> <*from*> <*to*> <*mask*> interface as this becomes unwieldy.
- The default when no options are given is **-XGPRDfMIbL**, default options may be set via the **find-setup**(3) command (sets **/history/rsif-com** registry entry).
- The **-d** option is not supported as **find**(3) and error is reported if supplied.
- It may be advantageous to enable the **-l** option default using **find-setup**(3).
- Use of **-w** is supported and is separate from the <*from-pattern*>, i.e. this option can still be used to filter the files to operate on.
- **-w** shares the **-m** and **-i** options, i.e if **-m** is used both the <*from-pattern*> and the <*search-pattern*> must be a regular expression pattern.

The other options of the underlying **find**(3) command are allowed but provide no benefit to these commands.

## SEE ALSO

**find**(1), **grep**(1), **find**(3), **grep**(3), **find-setup**(3), **get-next-line**(2), **grep**(1), **xfind**(3), **xgrep**(3), **rgrep**(3).

Ref: replace-string-in-files(3) File: m3mac090.3 Date: 2025/03/16

## NAME

reread-file - Reload the current buffer's file

## SYNOPSIS

*n* **reread-file**
*n* **reread-all** ["*path*"]

## DESCRIPTION

**reread-file** reloads from disk the file associated with the current buffer, this command is particularly useful when the file is continually updated by an external program. If the buffer has been edited and its name does not start with a '*' then the user is prompted as to whether the changes should be discarded. Also if the buffer has an active process running in it then confirmation is sort from the user before the process is killed.

The argument *n* can be used to change the default behavior of reread-file described above, *n* is a bit based flag where:

**0x01** Enables loss of work checks (default), if this flag is not supplied then the buffer is reread without any user prompts (useful in macros).

**reread-all** reloads out-of-date buffers. The argument *n* is a bit based flag where:

**0x01** Prompt before discarding edits in a modified buffer. This flag takes priority over **0x02**.

**0x02** Yes to all, discard edits in modified buffers without prompting.

**0x04** Reload all buffers, not just those that are out-of-date.

**0x08** Reload only those buffers whose file is within the given *path*. If *path* is not supplied on the command line the user is prompted for it.

Note that when both **0x01** and **0x02** are clear, buffers with unsaved edits are left untouched (not reloaded) and no prompt is given. The default with no numeric argument (i.e. *n* is zero) is thus to reload out-of-date buffers that have no unsaved edits, leaving any buffer with unsaved edits untouched.

## NOTES

**reread-file** is a macro implemented in `tool.emf`.

## SEE ALSO

**find-file**(2), **read-file**(2), **view-file**(2).

Ref: reread-file(3) File: m3mac053.3 Date: 2026/07/16

## NAME

restyle-buffer - Automatically reformat a buffer's indentation.
restyle-region - Automatically reformat a regions indentation.

## SYNOPSIS

**restyle-buffer** (**esc C-**)
**restyle-region**

## DESCRIPTION

**restyle-buffer** automatically re-formats the indentation of a buffer. The indentation only operates if the indentation method is defined with **$buffer-indent**(5), otherwise the command has no effect.

**restyle-region** modifies the indentation between *point* and *mark*.

## NOTES

**restyle-buffer** and **restyle-region** are macros defined in `format.emf`.

## SEE ALSO

**indent**(2), **$buffer-indent**(5).

Ref: restyle-buffer(3) File: m3mac025.3 Date: 2022/06/12

## NAME

save-all - Save all modified files (with query)

## SYNOPSIS

*n* **save-all**

## DESCRIPTION

**save-all** cycles through all buffers, dictionaries, registry files and current session writing back any changes made. For each buffer, dictionary, session or registry file which has been modified the user is prompted before the changes are saved, a value of **y** initiates the save, **n** skips the save.

The argument *n* can be used to change the default behavior of save-all described above, *n* is a bit based flag where:

**0x01** Enables the user prompt before the file is saved (default). If this flag is not supplied then all modified files will automatically be written.

## NOTES

**save-all** is a macro defined in `me.emf`, using commands **save-some-buffers**(2), **save-dictionary**(2), **save-session**(3), and **save-registry**(2).

## SEE ALSO

**save-dictionary**(2), **save-registry**(2), **save-session**(3), **save-some-buffers**(2).

Ref: save-all(3) File: m3mac002.3 Date: 2022/06/12

## NAME

save-session - Write session information to session file
session-setup - Configure the data written to a session file

## SYNOPSIS

*n* **save-session** "*session-file*"
**session-setup**

## DESCRIPTION

**save-session** writes out MicroEmacs's current session information into the given session file.

The command **read-session**(3) can set a default session file in which case the session is automatically written out to this file if an argument of zero is given; the user is not prompted for a file. MicroEmacs automatically tries to write the default session whenever it is exited.

The **session-setp** command opens a simple dialog to configure the current session. The session name will be put into the window title, if using a window based version of MicroEmacs, and using a different color scheme can help in distinguishing between different sessions.

By default, each session will create its own favorites list, see Favorites section within the File section on the main menu, which will be saved to *session-name*`.eff`. If the **Use session specific favorites** option is disaled a shared favorites file is used instead which will be saved to *$user-name*`.eff`

The saving of the kill chain and bookmarks can be extremely useful, but can also greatly increase the size of a session file and the time taken to read and write it.

## NOTES

The session information is saved in a ascii format file (see **esf**(8)). Reference should be made to the notes included in **esf**(8) as to how the session file may be edited and effected in the same MicroEmacs session.

## SEE ALSO

**esf**(8), **@h0**(4), **read-session**(3).

Ref: save-session(3) File: m3mac080.3 Date: 2025/12/01

## NAME

sort-lines-ignore-case - Alphabetically sort lines ignoring case”

## SYNOPSIS

*n* **sort-lines-ignore-case**

## DESCRIPTION

**sort-lines-ignore-case** forces the current buffers **exact**(2m) mode to off and then calls **sort-lines**(2) which will perform a case insensitive alphabetical line sort from the mark position to the current cursor position. The state of the current buffers **exact** mode is restored on completion.

## NOTES

**sort-lines-ignore-case** is a macro defined in `format.emf`, see help on command **sort-lines**(2) for a complete definition.

## SEE ALSO

**sort-lines**(2), **buffer-mode**(2), **exact**(2m), **transpose-lines**(2).

Ref: sort-lines-ignore-case(3) File: m3mac004.3 Date: 2022/06/12

## NAME

spell-add-word - Add a word to the main dictionary

## SYNOPSIS

*n* **spell-add-word** ["*word*"]

## DESCRIPTION

**spell-add-word** adds words to the last dictionary added using the command **add-dictionary**(2). If no argument is supplied the user is prompted for the word and rule flags, only a 'Good' word can be added (see below). If an argument *n* is given then the next *n* words from the current buffer are added. The words must take one of the following three forms:

`xxxx` - Good word `xxxx` with no spell rules allowed
`xxxx/abc` - Good word `xxxx` with spell rules `abc` allowed
`xxxx>yyyy` - Erroneous word with an auto-replace to `yyyy`

## NOTES

**spell-add-word** is a macro defined in file `spellutl.emf`. It is not defined by default so `spellutl.emf` must be executed first using **execute-file**(2).

## SEE ALSO

**add-dictionary**(2), **edit-dictionary**(3), **save-dictionary**(2), **delete-dictionary**(2).

Ref: spell-add-word(3) File: m3mac023.3 Date: 2022/06/12

## NAME

sql-format - SQL language, format SQL query

## SYNOPSIS

*n* **sql-format** [ "*sql-query*" ]

## DESCRIPTION

**sql-format** creates a new "`*sql-format-#*`" temporary buffer and inserts a formatted form of the current kill buffer. The formatting process attempts to make the SQL query more readable and manageable. The formatted string is also copied into the kill buffer, making it ready for pasting elsewhere.

The numeric argument **n** can be used to change the default behaviour of the command, where the bits are defined as follows:

**0x01** If this bit is not set the *sql-query* string is prompted for rather than using the kill buffer.

**0x02** If this bit is set then the reformatting takes the form of removing all excess white-spaces, reducing the query down to a single line.

## SEE ALSO

**sql**(9).

Ref: sql-format(3) File: m3mac096.3 Date: 2024/05/21

## NAME

start-up - Editor startup callback command
shut-down - Editor exit callback command

## SYNOPSIS

**start-up**
**shut-down**

## DESCRIPTION

By default **start-up** is not defined, if the command is defined (via a user macro) then it is executed immediately after MicroEmacs has completed its initialization.

This command may initially seem redundant as the user may execute any command at start-up by editing the "`me.emf`" file or using the '@' command-line argument. At the point of "`me.emf`" file execution none of the files specified on the command-line will be loaded, thus any actions required on the given command-line files will not work (the only buffer present will be the "***scratch***" buffer).

The **start-up** command is executed AFTER the execution of "`me.emf`" and initialization of buffers, but before MicroEmacs waits for user input.

The **shut-down** command is also not defined by default, but if it is defined during the running of MicroEmacs the command will be called when MicroEmacs exits. The command is not called if MicroEmacs has to perform an emergency exit (due to the system being shut down or process being killed etc).

## SEE ALSO

**me**(1).

Ref: start-up(3) File: m3mac024.3 Date: 2026/02/17

## NAME

str-to-regex - Generate regex safe variant of a string
filemask-to-regex - Convert a file mask to a regex

## SYNOPSIS

**str-to-regex** "*in-out-var*"
**filemask-to-regex** "*in-out-var*"

## DESCRIPTION

**str-to-regex** converts a given string to a regex-safe variant so that it can be used as part of a regex search. Regular expressions use several special characters, such as '`.`', '`*`', '`[`' and str-to-regex inserts a '`\`' quote character to make it a literal string. For example "`A.B*C\`" will be converted to "`A\.B\*C\\`".

**filemask-to-regex** converts a typical file mask, used by command-line tools in a console, to a regex. This typically involves the conversion of '`?`' to '`.`', '`*`' to '`.*`' and the quoting of other regular expression special characters, such as '`.`' to '`\.`'.

## NOTES

Both of these macros take a single argument, *in-out-var*, which must be the name of a variable. The initial value of this variable is used as the string to be converted and on completion the value of the variable is updated to the output regex string.

The **str-to-regex** and **filemask-to-regex** commands are implemented in `utils.emf` and are not publically available from the command line, but may be used within macros.

## SEE ALSO

**search-buffer**(2), **file-list**(3), **$file-names**(5), **&xsequal**(4).
Regular Expressions

Ref: str-to-regex(3) File: m3mac107.3 Date: 2025/12/01

## NAME

string-rectangle - Add a string vertically.
space-rectangle - Adds space(s) vertically.

## SYNOPSIS

**string-rectangle** "*string*"
*n* **space-rectangle**

## DESCRIPTION

**string-rectangle** inserts the *string* at the same column position on each line of the current region. The command prompts the user for a string and inserts it at the current cursor column on all the lines between the cursor and the mark inclusive. This is generally used to pre-pend lines with new text.

**space-rectangle** is the same as **string-rectangle** except it inserts *n* spaces, default being 1, rather than a given *string*. This command is very useful when moving a column of text over to align with another column.

## SEE ALSO

**exchange-point-and-mark**(2), **insert-space**(2), **kill-rectangle**(2), **set-mark**(2), **yank-rectangle**(2).

Ref: string-rectangle(3) File: m3mac063.3 Date: 2026/02/17

## NAME

svn - MicroEmacs SVN interface
svn-setup - MicroEmacs SVN interface setup
svn-add - MicroEmacs SVN interface - add items
svn-blame - MicroEmacs SVN interface - annotate a file
svn-cat-file - MicroEmacs SVN interface - output specific version of a file
svn-checkout - MicroEmacs SVN interface - checkout a working copy
svn-commit - MicroEmacs SVN interface - commit changes back
svn-copy - MicroEmacs SVN interface - copy items
svn-delete - MicroEmacs SVN interface - delete items
svn-diff - MicroEmacs SVN interface - diff changes
svn-gdiff - MicroEmacs SVN interface - graphical diff changes
svn-ignore - MicroEmacs SVN interface - edit directory item ignore list
svn-info - MicroEmacs SVN interface - display info about an item
svn-log-commit - MicroEmacs SVN interface - get log for a commit
svn-log-file - MicroEmacs SVN interface - list log of changes
svn-move - MicroEmacs SVN interface - move an item
svn-resolve - MicroEmacs SVN interface - resolve conflicts
svn-revert - MicroEmacs SVN interface - revert local changes
svn-server-copy - MicroEmacs SVN interface - server based copy items
svn-server-move - MicroEmacs SVN interface - server based move items
svn-status - MicroEmacs SVN interface - status of items
svn-update - MicroEmacs SVN interface - update directory files
svn-xdiff - MicroEmacs SVN interface - external diff changes

## SYNOPSIS

*n* **svn**
**svn-setup**
**svn-add**
**svn-blame**
**svn-cat-file**
**svn-checkout**
**svn-commit**
**svn-copy**
**svn-delete**
**svn-diff**
**svn-gdiff**
**svn-ignore**
**svn-info**
**svn-log-commit**
**svn-log-file**
**svn-move**
**svn-resolve**
**svn-revert**
**svn-server-copy**
**svn-server-move**
**svn-status**

**svn-update**
**svn-xdiff**

## DESCRIPTION

The svn and sub-commands provide MicroEmacs with an interface to Subversion's **svn**(1) command. **Subversion** is a version control system; using it, you can record the history of your source file modifications. Subversion is licensed under the Apache Software Foundation and is freely available on the Internet, see the documentation provided with Subversion for more information on its features and use.

The MicroEmacs **svn** command opens up a modified **file-browser**(3) with an additional "`*svn-console*`" window. The "`*files*`" window includes additional columns showing the SVN state, revisions and commit author. The browser by default opens with 3 windows including a representation of the file tree. When **svn** is provided with a numeric argument *n* of zero (0) then a single window displaying the directory is used only (i.e. **0 svn**)

The functionality of the file-browser is the same as a non-SVN folder with the exception that additional SVN item controls are located in the mouse context menu (opened by clicking the right mouse button in the `*files*` buffer). This menu item opens another sub-menu providing access to the items below. When selecting commands than a pop-up check-box may be displayed which allows options to be selected for the command.

**Current State**
: Executes the svn-status command.

**Checkout files**
: Executes the svn-checkout command.

**Update files**
: Executes the svn-update command.

**Commit files**
: Executes the svn-commit command.

**Diff files**
: Executes the svn-diff command.

**Graphical Diff**
: Executes the svn-gdiff command, one file must be selected.

**Item Log**
: Executes the svn-log-file command, one item must be selected.

**Item Blame**
: Executes the svn-blame command, one file must be selected.

**Copy items**
: Executes the svn-copy command, one or more items must be selected.

**Move items**
: Executes the svn-move command, one or more items must be selected.

**Add items**
: Executes the svn-add command, one or more items must be selected.

**Delete items**
: Executes the svn-delete command, one or more items must be selected.

**Resolve conflicts**
: Executes the svn-resolve command.

**Clear svn console**
: Clears the **svn-console** buffer.

**SVN Setup**
: Executes the svn-setup command.

### Sub-Commands

The **svn-add** command adds the current buffer or selected items to the repository. Note that this command only performs the local addition, a *SVN commit* is required to make the addition permanent.

The **svn-blame** command opens a **svn-blame** window annotating the revision and author of each line in the current buffer or single selected file.

The **svn-cat-file** command opens a **<file-name>:<revision>* window with the contents set to the specified revision of a selected file.

The **svn-checkout** command checks out a working copy of an item (file or directory) from a repository into the current directory. The user specifies the item via an SVN repository URL.

The **svn-commit** command commits any changes made to the current buffer's file or selection list to the repository. The user is prompted for a commit log message.

The **svn-copy** command creates a local copy of the current buffer's file or selected items; alternatively a full SVN repository URL can be given as the destination which will immediately commit the items back to the repository. The advantage of using copy over simply adding a new item is that the item's change log history up to the point of the copy should be maintained by SVN.

The **svn-delete** command flags the current buffer or selected files for removal from the repository, additionally the command can remove the local file too - PLEASE NOTE THIS CAN LEAD TO LOST DATA!!! This command only performs the local removal, the **svn-commit** command must be used to make the deletion permanent. Note that as the local file is typically removed the file itself cannot be selected when committing the change, use the **svn-status** command to list change as this list includes all items flagged for removal.

The **svn-diff** command opens a **svn-diff** window displaying the differences between the current buffer or selected local files and the repository version.

The **svn-gdiff** command opens a **gdiff**(3) window displaying the differences between two versions of the current buffer's file, the two versions being selected from the command's dialog. The command operates on a single file only. Where the right hand side is the local file any changes made may be saved back over it, repository versions cannot be saved.

The **svn-ignore** command opens a dialog to edit the list of items contained with the current directory which are to be ignored by SVN. Currently selected items are appended to this list and any changes made are only to the local working copy until the changes to the directory are committed back. Note that this command is obtaining and modifying the directory's `svn:ignore` property.

The **svn-info** command opens a **svn-info** window displaying the SVN info of the current buffer or single selected file.

The **svn-log-commit** command opens a **svn-log-commit** window displaying the SVN log for the given commit. If the output is processed then clicking on or pressing enter on any of the file names listed will execute a **svn-diff** showing the changes made.

The **svn-log-file** command opens a **svn-log-file** window displaying the SVN commit log history of the current buffer or single selected file.

The **svn-move** command moves (i.e. same as rename when the destination directory is the same) the current buffer's file or selected items, any required changes to the buffer should be saved first and the new file loaded once moved. Note that unlike copy, SVN does not support a direct local to reposition move. The advantage of using move over simply deleting and adding a new item is that the item's change log history will be maintained.

The **svn-resolve** command may be used to resolve conflicts caused by SVN updating the local working copy.

The **svn-revert** command reverts local modifications made to the current buffer's file or selected items back to the last checked out revision, discarding the local changes; unlike **svn-delete** this does not affect the repository. Items previously scheduled for addition via **svn-add** are by default simply unscheduled (returned to an unversioned state), the local file itself is left untouched; enabling the `Remove added file(s)` option additionally deletes these local files. Sub-directories are recursed into by default, this can be disabled via the `Recurse into sub-directories` option so that only the top-level of a selected directory is reverted.

The **svn-server-copy** command performs a repository based copy of the current buffer's file or selected item, to a new location in the repository. In SVN this is the easiest way to create a development branch or a release, for example to create the MicroEmacs 2022 release one could server copy `<svn-rep>/trunk/MicroEmacs` to `<svn-rep>/releases/MicroEmacs-2022`. As the copy is performed on the server it is immediately committed so the command prompts for a commit log comment.

Similar to **svn-server-copy**, the **svn-server-move** command performs a repository based move of the current buffer's file or selected item.

The **svn-status** command opens a **svn-state** window and shows the state of all the items within the current directory, including the modification state, and optionally whether items are out-of-date (i.e. have been modified in the repository).

The **svn-update** command updates all files in the current directory, the output being reported to a new **svn-update** window. Note that unlike most svn sub commands this command executes over all files in the current buffer's file directory.

The **svn-xdiff** command launches an external diff application to display differences, the command be configured first by setting the **%svn-xdiff-com** to an appropriate command-line, e.g. '`Meld -n %f`'. See **%xdiff-com**(5) for more information.

### Setup

**svn-setup** provides a dialog interface to configure the MicroEmacs SVN interface if required. On a typical system installation then it is sufficient to have the **svn** program in the **$PATH** and the SVN tools will operate from a working copy of a repository.

**svn-setup** may also be accessed from **user-setup**(3) from the **Miscellaneous** configuration. The the dialog provides the following configuration tabs and controls:

#### General

`SVN Command`
: Sets the svn command-line program name, this can be an absolute path.

`Global opts`
: Sets command-line arguments which are to be always given to the svn command, the default of `--non-interactive` informs SVN that the program is being run in the background so cannot interact with the user.

`Def Rep URL`
: Defines the default SVN repository URL, this is only used when using the **svn-checkout** command.

`Enable Support on Start-Up`
: When enabled the svn support macros are loaded on start-up enabling the file-browser to detect SVN controlled directories.

`Quick browser mode`
: When enabled the SVN file browser does not include the '`-u`' option when launching the svn status command to get the current state of all files in the directory, this means SVN does not need to go to the repository server to get the update information. This can make a big improvement to performance, particularly if the connection to the server is slow. In Quick mode the status of locally unmodified files is set to '`Q`' indicating the repository state of the file has not been ascertained. The full SVN information can be obtained by executing the command **svn-full-reread**, typically bound to `C-f5`.

**Tools** In this tab each sub-command can be selected and their default option settings changed as required.

## NOTES

**svn** and sub-commands are macros defined in file `svn.emf`.

When filtering is enabled in the file browser the filter used by the SVN interface depends on the file browser mode, if **Quick browser mode** is enabled the same filter is used, but when disabled all files ignored by **svn**(1) are also removed by the filter. Use **buffer-setup**(3) in the buffer to enable/disable the filter.

The svn-cat, svn-log and svn-commit-log commands were renamed to svn-cat-file, svn-log-file and svn-log-commit in August 2026 to align them with the equivalent Git and VCS commands.

The svn-gdiff command was added in August 2026 to align with the equivalent CVS, Git and VCS commands.

## SEE ALSO

**git**(3), **cvs**(3), **ClearCase**(3), **gdiff**(3), **file-browser**(3), **$rcs-file**(5).

Ref: svn(3) File: m3mac093.3 Date: 2026/08/03

## NAME

tabs-to-spaces - Converts all tabs to spaces

## SYNOPSIS

**tabs-to-spaces**

## DESCRIPTION

**tabs-to-spaces** converts all tab characters found in the current buffer with spaces. The number of spaces a tab is replaced with depends on the column of the tab character and the setting of **$buffer-tab-width**(5).

The cursor is restored to the start of the current line after completion.

## NOTES

**tabs-to-spaces** is a macro defined in `format.emf`.

## SEE ALSO

**$buffer-tab-width**(5), **$tab-width**(5), **tab**(2), **tab**(2m), **clean**(3), **space-rectangle**(3).

Ref: tabs-to-spaces(3) File: m3mac008.3 Date: 2022/06/12

## NAME

time - Command time evaluator

## SYNOPSIS

**time** "*string*"

## DESCRIPTION

**time** evaluates the time take to execute line "*string*". **time** uses command **execute-line**(2) to execute the given string.

## EXAMPLE

The following example simply times the time take to save the current buffer:

```
time "save-buffer"
```

## NOTES

**time** is a macro defined in `misc.emf`.

On multi-task systems like UNIX **time** cannot take into account the number of other processes running at the same time, it can only return the actual time elapse. This leads to inaccuracies and variation in results.

## SEE ALSO

**execute-line**(2).

Ref: time(3) File: m3mac010.3 Date: 2022/06/12

## NAME

toolbar - Toggle tool bar visibility.
toolbar-open - Open the tool bar.
toolbar-close - Close the tool bar
toolbar-refresh - Update the tool bar

## SYNOPSIS

*n* **toolbar**
**toolbar-open**
**toolbar-close**
**toolbar-refresh**

## DESCRIPTION

The toolbar is one or more fixed buffers that typically appear at the left-hand edge of the window that contains different tools that assist in editing the buffer. The toolbar is really a container for tools and short cuts which are configured by the user depending on the tool that has been loaded. The tools themselves are buffer context sensitive, the **buffer-init**(3) command automatically re-configures the toolbar for the buffer at run time.

The configuration of the toolbar is stored in command variables `.toolvar.*` which are used to control the toolbar. The tools that are available in toolbar are determined from the file `toollist.emf` which is loaded when the toolbar is started. User defined toolbars may be added by redefinging the toolbar variables at runtime.

**toolbar** toggles the visibility of the toolbar, i.e. if the toolbar is open it is closed and vica versa. If the numeric argument *n* is specified then zero closes the toolbar and non-zero opens the toolbar. **toolbar-open** initializes and starts the toolbar if it is not already running. The toolbar is loaded and instantiated in the system. The toolbar may be subsequently closed using **toolbar-close**. Whilst the toolbar is running then the content of the tollbar may become out of date, the command **toolbar-refresh**(3) forces the toolbar to be re-evaluated and the content is updated.

## CONFIGURATION

The toolbar is configured with command variables **.toolbar.***. The size information of the toolbar is defined with the following variables:

*.toolbar.width*
: Defines the default width of the left-hand vertical toolbar.

*.toolbar.depth*
: Defines the default depth of the top horizontal toolbar.

The tools within the toolbar are configured with the following variables. Each variable is divided into fields, each field represents a single tool. The fields are separated with the `\CH` character. The order of the fields MUST be the same in each of the variables such that the field content matches the tool that is being defined.

*.tool-flag*
: The **.tool-flag** variable describes the behaviour of the tool, this is defined as a bitmask as follows:

`0x000001` - Update on creation
: Call the tool update command when the tool is created, this is typically required when the content of the toolbar is generated.

`0x000002` - Update on redraw
: Call the tool update command when a redraw operation has been performed.

`0x000004` - Buffer type changed
: Call the tool update command when the buffer type has changed, a change of buffer type requires that a buffer specific tool is reconfigured.

`0x000008` - Current buffer changed
: Call the tool update command when the current buffer is changed, a change of buffer buffer requires that a buffer content specific tool is reconfigured.

`0x000010` - Current buffer modified or saved
: Call the tool update command when the current buffer is modified or saved.

`0x000080` - Perform update checks
: On an update check the toolbar for changes in content.

`0x000100` - Lock depth and width
: Lock the depth and width of the tool.

`0x000200` - Tool uses a paged window
: The tool uses a paged window in the toolbar (i.e. only one tool has focus).

`0x000400` - Tool is hidden
: The tool is currently hidden. When in this state the not active sub-page `0x200` will also be set.

`0x000800` - Tool paged window, not first user.
: The tool is part of a paged window and is not the first tool.

`0x001000` - position in the left toolbar
: The tool is present in the left toolbar, on toolbar creation the tool is automatically loaded into the toolbar.

`0x002000` - position in the top toolbar
: The tool is present in the top toolbar, on toolbar creation the tool is automatically loaded into the toolbar.

`0x004000` - position in the right toolbar
: The tool is present in the right toolbar, on toolbar creation the tool is automatically loaded into the toolbar.

`0x008000` - position in the bottom toolbar
: The tool is present in the bottom toolbar, on toolbar creation the tool is automatically loaded into the toolbar.

`0x010000` - Include buffer type in mode-line (Prefix)
: The buffer type is included in the toolbar mode line as a prefix.

`0x020000` - Include buffer name in mode-line (suffix)
: The buffer name is included in the toolbar mode line as a suffix.

`0x040000` - Include buffer filename in mode-line (suffix)
: The buffer file name is included in the toolbar mode line as a suffix.

`0x100000` - Restore window location after an update
: The position of the hilighted line in the toolbar is restored after an update.

`0x200000` - Move position to beginning of buffer after an update
: The position of the hilighted line is restored to the top of the buffer following an update.

*.toolbar.tool-name*
: The name of the tool that is displayed to the user.

*.toolbar.tool-descs*
: A short description of the tool which is used in the tool selection dialog.

*.toolbar.tool-buffs*
: The name of the buffer which is used to store toolbar information. The names are usually ***name*** so that they are not saved.

*.toolbar.tool-files*
: The macro file to load when the tool is loaded, if a tool is not used then the macros that define the tool are not loaded.

*.toolbar.tool-cmds*
: The creation and update commands to run when the command is created.

## FILES

Notable toolbox macro files:

- `toolbar.emf` - The main toolbar file.
- `toolstp.emf` - The toolbar setup menu i.e. the right click of the mouse on the toolbar mode line.
- `toollist.emf` - List off tools available in the toolbar.
- `bookmark.emf` - The bookmark tool.
- `toolstd.emf` - Basic tools (Info, Buffer list)
- `abbrlist.emf` - The buffer tools and buffer abbreviation tools.
- `itemlist.emf` - Buffer item list tool.
- `killlist.emf` - The kill list tool.

## SEE ALSO

**osd**(2), **recenter**(2).

Ref: toolbar(3) File: m3mac066.3 Date: 2022/06/12

## NAME

transpose-region-line - Exchange (swap) region with adjacent line

## SYNOPSIS

*n* **transpose-region-line**

## DESCRIPTION

**transpose-region-line** swaps the current region with the next line and moves the cursor with the region, effectively retaining the same text position. Repeating this *n* times moves the current region *n* lines down, if *n* is negative the region is moved up 0-*n* lines.

If no region is currently active the command moves the current line instead, defaulting to the same behaviour as **transpose-lines**(2).

## NOTES

**transpose-region-line** is a macro defined in `format.emf`.

## SEE ALSO

**transpose-lines**(2).

Ref: transpose-region-line(3) File: m3mac103.3 Date: 2025/07/23

## NAME

Triangle - MicroEmacs version of Triangle patience game

## SYNOPSIS

**Triangle**

## DESCRIPTION

**Triangle** is a solitaire game using a standard set of playing cards. The object of the game is to use all of the cards in the deck to build up four suit stacks from Ace to King.

The board is laid out so that every card is used to create a triangle shape. In the first column there is one up-turned card, in the second column there is one down-turned card and 2 up-turned, third has 2 down 3 up etc. The only break form this pattern is in the last 3 columns where there is an extra up-turned card so that all the deck is used.

Cards may be moved around the playing area by stacking the same suit cards in descending order on the row stacks. When a row stack has no up-turned cards on the stack then the top card may be turned over and may be played. If a stack becomes empty then only a King may be moved into the vacant position.

If the last card in a stack is an Ace then it can be moved to its suit stack, then the 2 of that suit etc. until finally the King is removed.

Cards are moved around the board using the mouse. Cards may be moved from one row stack to another row stack by placing the mouse over the 'from' stack and pressing the left mouse button. Move the cursor to the 'to' stack and release the left mouse button. If the move is legal then the card(s) are moved to the new stack. Multiple cards may be moved from the row stacks, the appropriate card(s) to be moved is automatically determined.

Cards may be moved onto the suit stacks by a single left mouse press and release on the same card, the card is moved to the appropriate suit stack. The same technique is used to turn cards over in the suit stacks.

Note that once a card is played onto the suit stacks then it cannot be removed.

To the right of the board are a number of control buttons. To select an option, click the left mouse button on it, the buttons are labeled:

**DEAL** Start a new game by dealing new cards.

**QUIT** Exit the game

**HELP** This help page

Note that the screen may be updated at any time using "*C-l*".

## NOTES

**Triangle** is a macro defined in `triangle.emf`.

The game is best played with a mouse, it is possible to play with the keyboard, as follows:

"*esc h*" for help

To move a card between stacks enter the source and destination column number ("*1*","*2*",.."*7*").

To overturn a card on the row stacks then enter the card column twice i.e. source and destination are the same.

To move a card from the row to the suit stacks then either enter the card column twice, or enter the destination as "*h*","*d*","*c*","*s*" (i.e. "*2 2*" or "*2 s*" to move the card in column 2 to the spades stack).

"*C-c C-c*" to deal the cards again.

"*C-l*" redraw the screen.

"*q*" to quit the game.

## SEE ALSO

Games, **Patience**(3), **Mahjongg**(3).

Ref: Triangle(3) File: m3mac016.3 Date: 2022/06/12

## NAME

uniq - Make lines in a sorted list unique

## SYNOPSIS

**uniq**

## DESCRIPTION

**uniq** reduces a sorted lines of text in the current buffer to a unique list such that no entries are repeated. The list is made unique from the mark position to the current cursor position (point). The operation is case sensitive.

## NOTES

**uniq** is a macro implemented in `tools.emf`.

For **uniq** to operate correctly then the list must have been previously sorted, see **sort-lines**(2).

## SEE ALSO

**sort-lines**(2), **sort-lines-ignore-case**(3), **transpose-lines**(2),

Ref: uniq(3) File: m3mac049.3 Date: 2022/06/12

## NAME

vcs - MicroEmacs VCS interface - VCS-agnostic front-end
vcs-id - MicroEmacs VCS interface - identify the VCS controlling a file/directory
vcs-add - MicroEmacs VCS interface - add items
vcs-annotate - MicroEmacs VCS interface - annotate a file
vcs-blame - MicroEmacs VCS interface - annotate a file
vcs-cat-file - MicroEmacs VCS interface - output a specific revision/commit of a file
vcs-checkout - MicroEmacs VCS interface - checkout (create) a working copy
vcs-coci - MicroEmacs VCS interface - check-out/check-in a file
vcs-commit - MicroEmacs VCS interface - commit changes to repository
vcs-delete - MicroEmacs VCS interface - delete/remove items
vcs-diff - MicroEmacs VCS interface - diff changes
vcs-fetch - MicroEmacs VCS interface - download from another repository
vcs-gdiff - MicroEmacs VCS interface - graphical diff changes
vcs-log-commit - MicroEmacs VCS interface - show log for a commit
vcs-log-file - MicroEmacs VCS interface - list log of changes
vcs-move - MicroEmacs VCS interface - move an item
vcs-pull - MicroEmacs VCS interface - download and integrate another repository
vcs-push - MicroEmacs VCS interface - update remote repository
vcs-remove - MicroEmacs VCS interface - remove items
vcs-resolve - MicroEmacs VCS interface - resolve conflicts
vcs-revert - MicroEmacs VCS interface - revert local changes
vcs-setup - MicroEmacs VCS interface - setup the underlying VCS
vcs-status - MicroEmacs VCS interface - status of items
vcs-switch - MicroEmacs VCS interface - switch or create branches
vcs-unstage - MicroEmacs VCS interface - unstage items
vcs-update - MicroEmacs VCS interface - update directory files

## SYNOPSIS

*n* **vcs**

*n* **vcs-id** [ "*file-path*" ]

**vcs-add**
**vcs-annotate**
**vcs-blame**
**vcs-cat-file**
**vcs-checkout**
**vcs-coci**
**vcs-commit**
**vcs-delete**
**vcs-diff**
**vcs-fetch**
**vcs-gdiff**
**vcs-log-commit**
**vcs-log-file**
**vcs-move**
**vcs-pull**
**vcs-push**

**vcs-remove**
**vcs-resolve**
**vcs-revert**
**vcs-setup**
**vcs-status**
**vcs-switch**
**vcs-unstage**
**vcs-update**

## DESCRIPTION

The **vcs** and sub-commands provide a VCS-agnostic front-end over MicroEmacs's individual version control system interfaces. Each command determines which VCS controls the current buffer's file (or the current directory) by calling **vcs-id**, and then forwards the request to the equivalent command of the underlying interface, i.e. **vcs-status** run against a file in a **git**(3) working copy executes **git-status**, in a **svn**(3) working copy executes **svn-status**, and so on. This allows a single set of key-bindings and menu entries to be used regardless of which VCS happens to control the file being edited.

This page only describes the behaviour common to **vcs** and its sub-commands; each command's action, its prompts and its options are ultimately those of the underlying VCS command it forwards to, so see **git**(3), **svn**(3), **cvs**(3) and **ClearCase**(3) for the details, dialogs and command-line options specific to that system.

Not every sub-command is implemented by every VCS interface; where this is the case it is noted below. If a sub-command is run against a VCS that does not implement it, an **ml-write**(2) message of the form `[VCS Error: <vcs> does not support <action>]` is reported and the command is abandoned. Similarly, a few pairs of sub-commands are simply synonyms of one another (e.g. **vcs-blame/vcs-annotate**) because the underlying VCS interfaces themselves use different names for the same action; these are noted below too.

The MicroEmacs **vcs** command itself opens the file-browser of the detected VCS, i.e. it is equivalent to running **git**(3), **svn**(3) or **cvs**(3) directly, including the same use of a numeric argument *n* of zero to open a single directory-only window (i.e. **0 vcs**). Only supported by Git, SVN and CVS; ClearCase does not currently provide an equivalent browser entry point via **vcs**, use **ClearCase**(3) directly instead.

### Identifying the VCS

**vcs-id** identifies which VCS (Git, SVN, CVS or ClearCase) controls a given file or directory, walking up the directory tree as necessary to locate a `.git`, `.svn` or `CVS` control directory, or, for ClearCase, a matching view record. The result of the last two lookups is cached so that repeated calls for files in the same directory are cheap.

All of the **vcs-*** sub-commands described below use **vcs-id** internally to select which `<vcs>-<command>` macro to forward to, and it is also used by the file-browser's directory-listing file-hook to select the appropriate VCS state format for a directory listing, see **file-browser**(3).

**vcs-id** optionally takes a file or directory path as its argument; if omitted the current buffer's file name is used. When called with a numeric argument (as it is internally by the other **vcs-*** commands and by the file-browser hook) it operates silently, only setting up its result for the calling macro. When run directly by the user with no numeric argument it instead reports the identified VCS on the message line, e.g. `VCS for "<file>" is Git (git)`. If no controlling VCS can be identified for the given file or directory an error is reported and the

command is abandoned.

### Sub-Commands

The **vcs-add** command adds the current buffer or selected items to the VCS.

The **vcs-annotate** command is a synonym for **vcs-blame**.

The **vcs-blame** command opens a window annotating each line of the current buffer or a single selected file with the revision/commit and author that last changed it.

The **vcs-cat-file** command opens a window with the contents of the current buffer's file, or a single selected file, set to a specific revision/commit. Currently not supported by ClearCase.

The **vcs-checkout** command checks out (creates) a working copy from the repository. Not supported by Git; Git users should use **vcs-switch** to change branches, or **vcs-revert/vcs-unstage** to discard local file changes.

The **vcs-coci** command performs a simple RCS-style check-out/check-in of the current buffer's file, see **rcs-file**(2). Of the supported VCS systems only ClearCase has this per-file check-out/check-in concept (Git, SVN and CVS have no equivalent lock/checkout model), toggling between checking the file out and checking it back in depending on the buffer's current **view**(2m) mode state; only supported by ClearCase. If the file is not controlled by any of Git, SVN, CVS or ClearCase then **rcs-file**(2) is invoked directly instead, giving equivalent behaviour on a standalone file kept under plain **$rcs-file**(5) control. Directories are not supported by either form of this command.

The **vcs-commit** command commits changes made to the current buffer's file or selection back to the repository, the user typically being prompted for a commit log message; exactly what gets committed (e.g. the index/staging area for Git, or the working copy directly for SVN/CVS) depends on the underlying VCS. Not currently supported for ClearCase, use **ClearCase**(3)'s own check-in command instead.

The **vcs-delete** command removes the current buffer or selected items from VCS control, and typically the local file too - PLEASE NOTE THIS CAN LEAD TO LOST DATA!!! **vcs-remove** is a synonym for this command.

The **vcs-diff** command displays the differences between the current buffer or selected local files and the repository version.

The **vcs-fetch** command downloads changes from a remote repository without altering the local files or index. Only supported by Git at present.

The **vcs-gdiff** command launches a graphical diff to display changes. Currently not supported by ClearCase.

The **vcs-log-commit** command displays the log message and affected files of a specific commit. Only supported by Git and SVN at present.

The **vcs-log-file** command displays the commit/revision log history of the current buffer or selected files. Not currently supported for ClearCase.

The **vcs-move** command moves (renames) the current buffer's file or selected items, preserving the item's change log history where the underlying VCS supports this. Not currently supported for CVS.

The **vcs-pull** command is a synonym for **vcs-update**.

The **vcs-push** command updates the remote repository with committed local changes. Only supported by Git.

The **vcs-remove** command is a synonym for **vcs-delete**.

The **vcs-resolve** command is used to resolve conflicts caused by updates from the remote repos-

itory. Only supported by SVN CVS at present.

The **vcs-revert** command reverts local modifications back to the repository version, discarding the local changes. Currently not supported by CVS or ClearCase.

The **vcs-setup** command opens the setup dialog of the current buffer's underlying VCS interface, e.g. **git**(3)'s **git-setup**; it does not have any configuration of its own, each underlying interface must be set up individually.

The **vcs-status** command opens a window showing the state of all the items within the current directory. Not currently mapped for ClearCase, which names its equivalent command **cc-state**, see **ClearCase**(3).

The **vcs-switch** command switches to an existing branch, or creates and switches to a new one. Only supported by Git at present; SVN, CVS and ClearCase have no equivalent branch-switching concept for this command to forward to.

The **vcs-unstage** command removes items from the staging area/index without discarding the underlying local modifications. Only supported by Git at present.

The **vcs-update** command updates the local working files with changes from the repository. **vcs-pull** is a synonym for this command.

## NOTES

**vcs** and sub-commands are macros defined in file `vcs.emf`. They are a thin dispatch layer only; the underlying VCS interface (`git.emf`, `svn.emf`, `cvs.emf` or `cc.emf`) for a given file's VCS must be set up correctly, e.g. via **git-setup**, **svn-setup**, **cvs-setup** or **cc-setup**, for its **vcs-*** commands to function.

## SEE ALSO

**git**(3), **svn**(3), **cvs**(3), **ClearCase**(3), **file-browser**(3), **rcs-file**(2), **$rcs-file**(5).

Ref: vcs(3) File: m3mac109.3 Date: 2026/08/03

## NAME

vm - Email viewer
mail-check - Check for new email
stop-mail-check - Disable the check for new email
mail - Compose and send an email
mail-setup - Email setup dialog

## SYNOPSIS

**vm**
**mail-check**
**stop-mail-check**
**mail**
**mail-setup**

## DESCRIPTION

**vm** is a simple email manager, it is configured to send and receive emails using the **mail-setup** dialog. It must be stressed that **vm** has only been tested in one environment, caution should be used as system differences may cause problems, such as loss of data, which the author does not except any responsibility for.

**mail-check** tests the size of this incoming mail box, a non-zero length indicates that new mail has arrived and **mail-check** informs the user by inserting a 'M' in the mode-line (2nd character for the left) and ringing the system bell. **mail-check** uses **create-callback**(2) to check for new mail every 10 minutes, this can be disabled by executing **stop-mail-check**.

When **vm** is executed it checks for new mail, if found it first copies the new mail to a file called "new_mail" in the users mail directory. The incoming box is then emptied by truncating the file to zero length. The users main mail box is then loaded and the new mail (if any) is appended. The mail box is then processed after which 2 windows are created the bottom window listing all messages in the box and the top displaying the current message.

**vm** is capable of:

- Scrolling through the mail box displaying each message (up, p, down, n, return, space).
- Check and get new mail messages (g).
- Extract and cut embedded data files (x, C, c).
- Reply to and forward mail messages (R, r, z).
- Delete mail messages (d, u).
- Archive messages to other mail boxes (A, a).
- Save changes to the current mail box (S, s).
- Delete the current mail box (D).
- Visit another mail box (v).
- Send a mail message (m).

- Hide vm windows (`delete`).

Use the vm help page (bound to "`esc h`") for further information.

**vm** supports two types of embedded data, uuencode and mime encoding and uses **ipipe-shell-command**(2) to extract the data, the commanding to use must be supplied by the user using the setup dialog, which can contain the following special tokens:

`%i` Temporary file name, if used, the embedded data is written to the this file first.

`%o` User supplied output file name, if `%i` is not used, the embedded data is written to this file first.

`%b` The output base name, i.e. `%o` without the path.

If no command line is supplied then the embedded data is written to the user supplied file name as a text file in the form found in the mail message.

**mail** can be used to compose and send an email, it can insert embedded data in a similar way to **vm**'s data extraction, the following special tokens can be used:

`%i` The user supplied data file to be embedded.

`%b` The input base name, i.e. `%i` without the path.

`%o` Temporary file name used to output the processed data file, this file is inserted into the mail message using **insert-file**(2).

**mail** also uses **ipipe-shell-command** to send the mail message, the following special tokens can be used:

`%f` The from user name.

`%s` The email subject.

`%t` A comma separated list of 'To' recipients.

`%c` A comma separated list of 'Cc' recipients.

`%o` A file name of the mail message.

Any field not used in the command-line is left at the head of the mail message.

**EXAMPLE - UNIX**

The following command-line can be used on most UNIX systems to extract uuencoded data:

```
rm -f %o ; uudecode %i ; rm -f %i
```

The following command-line can be used on most UNIX systems to extract mime encoded data:

```
rm -f /tmp/%b ; metamail -B -d -q -w -x -z %i ; mv -f /tmp/%b %o
```

The following command-line can be used on most UNIX systems to uuencode a data file ready for it to be embedded, the original file is not changed:

```
uuencode %b < %i > %o
```

The following command-line can be used on most UNIX systems to send an email:

```
/usr/lib/sendmail -oi -oem -odi -t < %o
```

### EXAMPLE - WIN32

Typically the **cygwin**(1) utilities can be used for data insertion and extraction. These have the advantage of being very similar to the unix ones so only minor changes are required, i.e. try the following for data insertion and mime & uuencode extraction respectively:

```
del %o ^ uudecode %i ^ del %i
del c:\tmp\%b ^ metamail -B -d -q -w -x -z %i ^ move c:\tmp\%b %o
uuencode %b < %i > %o
```

This assumes that the shell you are using supports the '^' multiple commands on a single line feature, this is supported by **4dos**(1) and **4nt**(1). If your shell does not support this feature a simple batch file command could be used instead.

**postie**(1) is a freely available pop3/smpt e-mail support program, available on the net, which can be used to provide a fully working **vm** on windows systems. As it is typically used in a dial-up connect environment, the **user-setup** 'Queue Outgoing Mail' option will be enabled while the 'Check Mail' and 'VM Gets Mail' will be disabled. This ensures that a connection is only made when the **vm** '`g`' command is used which sets all queued outgoing mail and gets any incoming mail.
The following command-line can be used to get mail from your pop server using postie:

```
postie -host:pop-mail-addr -user:user-addr -pass:password -file:inbox
        "-sep:From root Mon Jan 11 20:02:02 1999" -raw -rm
```

Where the `inbox` is the 'Incoming Mail Box' file specified in user-setup. The `-sep` option is used to partition each mail message from the previous message, this string is used as it is in a unix standard form so the resulting mail box could be understood by unix mail systems such as netscape etc.

NOTE: The **-rm** option is used to remove the incoming mail messages from the server. It is strongly recommended that the system is thoroughly tested without this option first.
The following command-line can be used to send mail to your smtp server using postie:

```
postie -host:smtp-mail-addr "-from:user@mail-addr" -use_mime:0
        "-to:%t" "-s:%s" "-cc:%c" "-file:%o"
```

**blat**(1) is another freely available windows program which can be used to send mail with the following command-line:

```
blat %o -f %f -s \"%s\" -t \"%t\" -c \"%c\"
```

### E-Mail Setup

The **E-Mail Setup** dialog is opened by executing the **mail-setup** command and configures a user to use part or all of the **vm** E-Mail manager, as follows:

`User Mail Dir`
: Sets the user mail-box directory where all files are to be found and stored (except usually the **Incoming Mail box**). The value of this field is platform independent and must be setup for each one.

  The following fields are used for sending mail:

`Send Mail Signature`
: Sets the signature file name which is inserted at the bottom of every out-going email message, if empty the no signature is inserted. The value of this field is platform independent, is value use by all. The file must be located in the **User Mail Dir** and no path entered for it to work across platforms.

`Carbon-Copy File`
: Sets the sent-mail carbon-copy file, creating the "`Fcc:`" line of the mail buffer. All out-going emails are appended to the end of this file if the "`Fcc:`" line is not altered. If this field is left empty then no "`Fcc:`" line is created. The value of this field is platform independent, the file must be located in the **User Mail Dir**.

`Insert Data (^C^I)`
: Sets the first embedded data command line, bound to "`C-c C-I`". The value of this field is platform dependent.

`Insert Data (^C^Z)`
: Sets the second embedded data command line, bound to "`C-c C-z`". The value of this field is platform dependent.

`Send Mail Command`
: Sets the command-line used for sending email messages. The value of this field is platform dependent.

The following fields are used for receiving mail:

`Check for mail`
: Sets the time interval between the automatic checking for incoming mail in seconds, when set to 0 the automatic checking is disabled. When enabled, the check is performed by **mail-check**(3) which also sends any queued mail and gets any new mail if the **Get Mail Command** is used. The value of this field is platform dependent.

`Get Mail Command`
: The command used to get new mail from the server, if empty it is assumed the **Incoming Mail Box** is automatically updated by the system. If used the command must append new mail to the end of the **Incoming Mail Box** specified below. The value of this field is platform dependent.

`Incoming Mail Box`
: Sets the incoming mail box file which new incoming mail is appended to, either automatically by the system or by the **Get Mail Command**. The value of this field is platform dependent.

`VM Main In Box`
: Sets the main current mail box, or inbox. The value of this field is platform independent, the file must be located in the **User Mail Dir**.

`VM Gets Mail`
: When enabled, executing the command vm will not only create the mail box windows, it will also get and process any new mail. When disabled only the vm '`g`' command can be used to get and process new mail.

`Mime Data Extract`
: Sets the command-line used for extracting Mime encoded embedded data. The value of this field is platform dependent.

`Uuencode Extract`
: Sets the command-line used for extracting Uuencoded embedded data. The value of this field is platform dependent.

`Auto-Archive Setup`
: Sets up the auto-archive of messages in the current inbox to other mail boxes.

## NOTES

**vm** is a macro defined in `vm.emf`, **mail-check**, **stop-mail-check** and **mail** are macros defined in `mail.emf`.

**vm** has only been tested in a couple of environments, the author will not except any responsibility for any loss of data, i.e. use at your own peril. You have been warned! Back-up all data files and test **vm** THOROUGHLY before using it.

## SEE ALSO

**user-setup**(3), **ipipe-shell-command**(2), **create-callback**(2), **sendmail**(1).

Ref: vm(3) File: m3mac028.3 Date: 2024/10/13

## NAME

watch - Watch a file

## SYNOPSIS

*n* **watch**
*n* **watch-reset**
**watch-time**

## DESCRIPTION

**watch** monitors a file on the file system by periodically polling the file and reacquiring the contents into a buffer. The command is typically used to monitor build and logging files as their output changes under control of a background process. Any number of files may be watched simultaneously, however to minimize the number of file system transfers, only those buffers that are being viewed in a window are updated.

**watch** operates in three different modes:

**Mode 1 (Reload)**
: On a re-fresh the existing buffer is emptied and the file contents are re-read in their entirety from the file system if the file has changed. This is the default mode of operation.

**Mode 2 (Reload, Append)**
: On enabling the watch then the complete file is re-read, changes are subsequently read as deltas from the end of the file. This mode ensures that the buffer contents are up to date before incrementally reading the file.

**Mode 3 (Append)**
: On enabling the watch then only the end of the file that has not been read before is appended to the end of the buffer. This method is ideal for large log files.

The operational mode may be selected by passing a numeric value of `1`, `2` or `3` to the **watch** command. If no numeric argument is provided then the user is prompted for the operational mode.

Whilst running in an append mode (`2` or `3`) then byte offset of the end of the file is retained for the next read. On the next read then any new information from the last saved byte offset to the end of the file is read, irrespective of any edits made to the buffer. Therefore if the buffer contents have been deleted on the next *append* read then the buffer will contain the appended information only and not reacquire the information that has been deleted. Where the complete buffer is to be re-acquired then use **Mode 1**.
The buffer mode line is modified to show the watch state. i.e.

```
==W <buffer-name> ==============
```

Invocation of **watch** in a buffer that is being watched disables the watch. All watches may be removed from all buffers using `0 watch-reset`.

**watch-time** modifies the poll time of a watch. The user is prompted for the time period which is entered in seconds.

**watch-reset** re-evaluates the watches that are on the buffers reporting the number of files that are bing watched. When invoked with a `0` argument then any outstanding watch commands are removed.

## NOTES

**watch** is a macro implemented in `watch.emf`.

The commands **watch-reset** and **watch-time** are not available until a **watch** command has been executed.

## HISTORY

**watch** was developed in 2000 and appeared in MicroEmacs releases after this date however the comand was not exported to the user. The **watch** command was fully integrated in the 2005 release.

## SEE ALSO

**osd-search**(3).

Ref: watch(3) File: m3mac077.3 Date: 2022/06/12

## NAME

which - Program finder
.which.result - Program path

## SYNOPSIS

**which** "*progname*"
**.which.result** "*string*"

## DESCRIPTION

**which** searches for the given program "*progname*" on the system path (set by the environment variable **$PATH**). If found the location is printed on the message line, otherwise an error message is printed and the command fails.

The variable **.which.result** is set to the last found program or the string "`ERROR`" if the program was not found.

## NOTES

**which** is a macro defined in `tools.emf`, it uses the **&which**(4) macro function.

## SEE ALSO

**&which**(4).

## NAME

write-region - Write contents of region to named (new) file

## SYNOPSIS

*n* **write-region** "*file-name*"

## DESCRIPTION

**write-region** is used to write the contents of a selected region of a buffer to a NEW file i.e. where the region exists between *point* **and** *mark*.

**write-buffer** writes the contents of the current region to the named file *file-name*. The action of the write also changes the file name associated with the current buffer to the new file name.

On writing the region, if **time**(2m) mode is enabled then the time stamp string is searched for in the file and modified if located, to reflect the modification date and time.

If the region contains a **narrow**(2m) it will automatically be removed before saving so that the whole region is saved and restored when saving is complete.

The argument *n* may be used to change the default behavior of write-region described above, *n* is a bit based flag where:

**0x01** Enables validity checks (default). These include a check that the proposed file does not already exist, if so confirmation of writing is requested from the user. Also MicroEmacs checks all other current buffers for one with the proposed file name, if found, again confirmation is requested. Without this flag the command will always succeed wherever possible.

**0x02** Disables the expansion of any narrows (see **narrow-buffer**(2)) before saving the buffer.

## NOTES

**undo**(2) information may be discarded when the file is written.

**write-region** is a macro defined in `misc.emf`, this macro was used as an example for Writing MicroEmacs Macros.

## SEE ALSO

**append-buffer**(2), **save-buffer**(2), **write-buffer**(2).
Writing MicroEmacs Macros.

Ref: write-region(3) File: m3mac076.3 Date: 2022/06/12

## NAME

xdiff - External graphical file difference
%xdiff-com - Extenal graphical differnce xdiff(1) command line

## SYNOPSIS

**xdiff** "*leftVersion*" "*rightVersion*"

`%xdiff-com` "*string*"

X-Windows "`xdiff -wb %lf %rf`" when using **xdiff** from *http://sgifiles.irixnet.org/sgi/xdiff/xdiff/*

MS-Windows "`bc2.exe %lf %rf /title1=%lt /title2=%rt`" when using **Beyond Compare**.

All "`meld %lf -L %lt %rf -L %rt`" when using **Meld**.

## DESCRIPTION

**xdiff** is a macro wrapper that invokes an external graphical file difference utility, named after the original X-Windows utility of the same name. The utility will usually display two files side by side and color code the differences, the differences may then be selected to merge the changes into a new output file which may (or may not) overwrite one of the original input files.

**xdiff** executes the difference utility **xdiff**(1) command (or equivalent) with the command line set by the **%xdiff-com** variable and the user supplied files *leftVersion* and *rightVersion*. The command line variables are referenced from the **%xdiff-com** as follows:

**%lf** The absolute file name of the left-hand file, *leftVersion*.

**%rf** The absolute file name of the right-hand file, *rightVersion*.

**%lt** The title to use for the left-hand file, *leftVersion*.

**%rt** The title to use for the right-hand file, *rightVersion*.

The **%xdiff-com** variable should include the appropriate command to invoke the difference utility and use **%** place holders for the arguments. MicroEmacs will fill in the **%** arguments with their correct values on invocation.

**Meld**(1) is a free graphical file compare available on Gnome capable UNIX systems, Windows and macOS, the variable command line may be defined as:

```
set-variable %xdiff-com "meld %lf -L %lt %rf -L %rt"
```

Unix environments may use **xdiff**(1) which is available from **http://sgifiles.irixnet.org/sgi/xdiff/xdiff/**. The variable command line may be defined as:

```
set-variable %xdiff-com "xdiff -wb %lf %rf"
```

The **-wb** options will ignore while space and trailing blanks. Microsoft Windows environments may use something like **Beyond Compare**. The variable command line may be defined as:

```
set-variable %xdiff-com "bc2.exe %lf %rf /title1=%lt /title2=%rt"
```

In this case the **%lt** values have been used for the title names for each of the windows in the utility.

## NOTES

**xdiff** is a macro defined in `tools.emf`, inspired by the UNIX utility of the same name **xdiff**(1)

## SEE ALSO

**Meld**(1), **compare-windows**(2), **compile**(3), **diff**(1), **gdiff**(3), **gdiff**(3f), **xgrep**(3), **%xgrep-com**(5).

Ref: xdiff(3) File: m3mac083.3 Date: 2023/02/21

## NAME

xfind - Execute find command
%xfind-com - Find command line

## SYNOPSIS

*n* **xfind** "*base-path*" "*file-search*"
**%xfind-com** "*string*"; Default is "`find`"

## DESCRIPTION

**xfind** executes the **find**(1) command with the command line set by the **%xfind-com**(5) variable and the user supplied *base-path* and *file-search* regular expression. The output of the command is piped into the ***find*** buffer.

If an argument is given then a **pipe-shell-command**(2) is used instead of **ipipe-shell-command**(2), this is useful when used in macros as it ensures that **find** has finished before the command returns.

The find command is specified by the macro variable **%xfind-com** if not defined the default `find` is used. The command line run by **xfind** is

*%xfind-com* -name "*base-path*" -name "*file-search*" -print

The resultant files found are displayed in the ***find*** buffer and may be selected by double-clicking with the mouse left button or selecting the line with a <RETURN>. The selected file under the cursor is loaded.

## EXAMPLE

The **xfind** command does not typically need to be set up in UNIX environments. In the Microsoft Windows environment then the UNIX *tools for Windows* are recommended, available from:

http://unxutils.sourceforge.net/

The **xfind** command may be set up in the startup files as follows:

```
set-variable %xfind-com "c:/usr/local/wbin/find.exe"
```

## SEE ALSO

**%xgrep-com**(5), **find**(1), **find**(3), **grep**(3), **get-next-line**(2), **grep**(1), **xfind**(3), **xgrep**(3), **rgrep**(3).

Ref: xfind(3) File: m3mac075.3 Date: 2022/06/12

## NAME

xgrep - Execute grep command
rgrep - Execute recursive grep command

## SYNOPSIS

**xgrep** "*expression files...*" **rgrep** "*expression*" "*base-path*" "*file-mask*"

## DESCRIPTION

**xgrep** executes the **grep**(1) command with the command line set by the **%xgrep-com**(5) variable and the user supplied *expression* and file list *files...*. The output of the command is piped into the ***grep*** buffer ready for the **get-next-line**(2) command to step through all matched lines. The syntax from the grep output must be setup using **add-next-line**(2).

If an argument is given then a **pipe-shell-command**(2) is used instead of **ipipe-shell-command**(2), this is useful when used in macros as it ensures that **grep** has finished before the command returns.

**rgrep** is similar to **grep** in that it uses **grep**(1) to search for all occurrences of *expression*, but **rgrep** also uses **find**(1) to search for *expression* in all files matching the *file-mask* in all directories from *base-path* down.

## NOTES

**xgrep** is a macro defined in `xfind.emf`.

**grep**(1) must be executable on the system before grep or rgrep can function, **find**(1) must also be available for rgrep to work. The **find**(1) command is defined by variable **%xfind-com**(5), the default when not defined is `find`.

## EXAMPLE

The **xgrep** command is generally set up in the startup files as follows:

```
;
; setup the next-error stuff including grep and compiling
;
set-variable $line-template "[0-9]+"
set-variable $file-template "[a-zA-Z:]*[0-9a-zA-Z\_.]+"
;
; Definitions for GNU grep utility.
;
set-variable %xgrep-com "grep -n "
0 add-next-line "*grep*"
add-next-line "*grep*" "%f:%l:"
```

## SEE ALSO

**%xfind-com**(5), **%xgrep-com**(5), **add-next-line**(2), **compile**(3), **find**(1), **xfind**(3), **get-next-line**(2), **grep**(1).

Ref: xgrep(3) File: m3mac001.3 Date: 2022/06/12

## NAME

xrdb-scheme - Load a XRDB scheme file from the user's config folder

## SYNOPSIS

**xrdb-scheme** "*[scheme-name...]*"

## DESCRIPTION

**xrdb-scheme** "*[scheme-name...]*" loads from the themes folder within the users config directory the given xrdb scheme file. The scheme name should be given without extension. If no argument is given, the user can interactively select the scheme from the command line.

## NOTES

The command allows to use different editor editor themes without creating individual scheme macro files. A web resource which currently has more than 400 color schemes and their xrdb files that are usable with MicroEmacs is:

```
https://github.com/mbadolato/iTerm2-Color-Schemes
```

The user has to download the raw xrdb files from that side and place them into a themes folder within the users MicroEmacs config folder. Below is a little example command line for Unix systems to download 3 color schemes:

```
mkdir ~/.config/jasspa/meuser/themes
wget -O ~/.config/jasspa/meuser/themes/mathias.xrdb \
  https://github.com/mbadolato/iTerm2-Color-Schemes/raw/master/xrdb/Mathias.xrdb
wget -O ~/.config/jasspa/meuser/themes/material-light.xrdb \
  https://github.com/mbadolato/iTerm2-Color-Schemes/raw/master/xrdb/Material.xrdb
wget -O ~/.config/jasspa/meuser/themes/material-dark.xrdb \
  https://github.com/mbadolato/iTerm2-Color-Schemes/raw/master/xrdb/Material%20Dark.xrdb
```

After you executed these commands you should be able to use the downloaded themes after executing: **Esc x** and **xrdb-scheme** and then pressing two times the TAB key. Feel free to browse that README on the **iTerm2-Color-Schemes** project page and then replace the wget code with the theme names.

Please note that to change the appearance of these schemes you can't use the **scheme-editor(3)** command, but you have to change the color values directly within the xrdb file.

## SEE ALSO

**scheme-editor**(3), **user-setup**(3).

Ref: xrdb-scheme(3) File: m3mac105.3 Date: 2025/09/26

## NAME

alias-path-setup - Define alternative path alias

## SYNOPSIS

**alias-path-setup**

## DESCRIPTION

**alias-path-setup** provides a dialog interface for declaring alternative alias paths. Alias path is a short name, conventionally then ~ is the home directory. The alias path setup allows other directory paths to be bound to a name i.e. "**me**" may be associated with "**/home/jon/merep/me/**". Given the alias definition then **me** expands to **/home/jon/merep/me/**.

**alias-path-setup** is accessible from the **user-setup**(3) **Miscellaneous** tab.

Multiple aliases may be defined and they are automatically accessible from the mode line when a file or directory name is entered.

## SEE ALSO

**user-setup**(3).

Ref: alias-path-setup(3) File: m3osd016.3 Date: 2022/06/12

## NAME

buffer-setup - Configure the current buffer settings

## SYNOPSIS

**buffer-setup**

## DESCRIPTION

**buffer-setup** provides a dialog interface for configuring the buffer major mode, indentation and buffer modes. **buffer-setup** may be invoked from the main *Tools* menu or directly from the command line using **execute-named-command**(2).

The **buffer-setup** dialog may be used to quickly reconfigure the attributes of a given buffer. The settings are only temporary and expire once the buffer has been closed or re-loaded. **buffer-setup**(3) should be used to associate user specific settings with different buffer types.

The following controls are available:

**Major Mode**
: Associates a file type with the content of the file and is used to modify the buffer highlighting and content assisted controls if the file type has not been recognized correctly. Changing the major mode will cause the new file type template to be loaded and may cause the buffer key bindings to change. The default major mode settings are set with **major-mode-setup**(3).

**Indent Width**
: Defines the indent width to use when the TAB key is pressed, or following a nested construct in a auto-indent buffer. This entry modifies the **$buffer-indent-width**(5) setting. The default indentation width is defined by the variable **$indent-width**(5).

**Tab Width**
: The width of a rendered TAB character in spaces, modification of this value immediately changes the layout. This modifies the buffer width tab variable **$buffer-tab-width**(5) the default is 8 (defined by **$tab-width**(5)). Where files have been edited with a non-standard tab width then modification of this value (to typically 2 or 4) may correct the problem. To correct the styling of the buffer to be editor independent then the command **tabs-to-spaces**(3) may be used to change all of the tab characters to spaces once the tab width has been correctly set.

**Fill Column**
: A text layout variable, defines the maximum length of the line before wrapping occurs when a text paragraph is filled. This modifies the buffer fill column variable **$buffer-fill-col**(5) the default is 78 (defined by **$fill-col**(5)).

**Fill Mode**
: A text layout variable, defines how paragraphs should be filled across multiple lines. This modifies the buffer fill mode variable **$buffer-fill-mode**(5) the default is **B** which causes automatic detection of paragraph layout, defaulting to full justification against the left and right margins (defined by **$fill-mode**(5)).

**Use Auto Indentation**
: Toggles automatic indentation on an off, this setting is only present if the buffer type supports a **indent**(2) scheme. Typically used for files such as lex and yacc which use

a special syntax combination with C.

**Use Hilighting**

Toggles hilighting on and off, this setting is only present if the buffer type supports a **hilight**(2) scheme.

**Buffer Modes**

The modes of the buffer, refer to Operating Modes for more information.

A notable mode switch here is **tab**(2m) which changes the behavior of the `TAB` key, when set tabs are simulated with the insertion of spaces, when disabled literal tab characters are used.

## HISTORY

Prior to the 2006 release **buffer-setup** was previously called **indent-setup** and dealt with the indentation settings only. The 2006 release introduced the concept of **Major Mode**.

## SEE ALSO

**buffer-setup**(3), **execute-named-command**(2), **hilight**(2), **indent**(2), **major-mode-setup**(3), **osd**(2), **tabs-to-spaces**(3) **$buffer-fill-col**(5), **$buffer-fill-mode**(5), **$buffer-indent-width**(5), **$buffer-tab-width**(5), **$fill-col**(5), **$fill-mode**(5), **$indent-width**(5), **$tab-width**(5).

Operating Modes.

## NAME

describe-word - Display a dictionary definition of a word

## SYNOPSIS

**describe-word** "*word*"

## DESCRIPTION

**describe-word** can be used to interface to an external dictionary to get a definition of a given word. When executed **describe-word** will use the current word under the cursor as the initial *word* or will prompt the user if the cursor is not in a word.

The command provides four methods of interfacing with an external dictionary and each language can be configured to use one or more dictionaries, allowing secondary sources to be used if the primary lookup failed. The first interface simply launches an external program which provides the definition in its own user interface, e.g. GoldenDict or MS Bookshelf. A lookup using this interface is always assumed to be successful so this should always be the last one for a language.

The second interface launches an external program which prints out the definition to `stdout`, MicroEmacs can then pull out the definition and display it in a ***describe-word*** buffer.

The third interface requires a URL, such as http:// or dict://, which provides the definition when loaded, and a clean-up macro the turns the definition into a readable form.

The last interface requires a dictionary text file, a regex search string and a clean-up macro. **describe-word** loads the dictionary file and searches for definitions of the given word, adding each one to the output buffer, it then calls the clean-up macro to format the definitions into a more readable form.

## NOTES

**describe-word** is a macro implemented in `word.emf`.

Due to the size and availability of dictionaries etc. MicroEmacs is released with describe-word set up to use the most appropriate dictionaries available at `dict://dict.org/`.

**describe-word** must be setup for each required language, which requires the setting of a **.describe-word.<lang-id>** variable. The variable must be a MicroEmacs list of dictionary interface configurations, each one being a sub-list to configure the interface to a dictionary source, each sub-list must take one of four forms:

### External

To interface with an external program the sub-list has the following items:

```
|<flags>|<source-name>|<not-used>|<command-line>|
```

Where:

**<flags>**

Is a list of option flags, one of which must be `x` to indicate an external source. Other flags supported by an external source are `i` to use an ipipe to launch, this is useful when needing to debug the interface, and `s` for a single, or shared, output buffer which is only used when ipipe mode is also being used.

**<source-name>**
Defines the name of the interface, or source, and is only used when ipipe mode is also being used.

**<not-used>**
The 3rd item is not currently used and should be left empty.

**<command-line>**
Defines the command-line required to launch the external dictionary program, the **<command-line>** should contain a `%w` token which is replaced with the given word, the token will typically also require surrounding `%"` tokens (i.e. `%"%w%"`) so that if the word contains a space it will be properly quoted in the generated command-line.

**Command**

To interface with an external program which prints the definition out to `stdout`, the sub-list has the following items:

```
|<flags>|<source-name>|<clean-up-macro>|<command-line>|
```

Where:

**<flags>**
Is a list of option flags, one of which must be `c` to indicate a command source. The only other flags currently supported by a command source is `s` for a single, or shared, output buffer, without this flag the buffer name includes the word being described.

**<source-name>**
Defines the name of the interface, or source, which is used in the details given in the first line of the output buffer.

**<clean-up-macro>**
The clean-up macro has two purposes, firstly if the macro aborts (i.e. calls **!abort**(4) etc.) it signals to **describe-word** that the output does not actually contain a definition, for example the output may simply be "`No definitions found for <word>`". The second purpose is to perform any clean-up and reformatting required to make the output more legible. **describe-word** provides a simple **describe-word-cmd** macro which simply aborts if the output contains one line with the line starting with "`No definitions found for `".

**<command-line>**
Defines the command-line required to launch the dictionary program, the **<command-line>** should contain a `%w` token which is replaced with the given word, the token will typically also require surrounding `%"` tokens (i.e. `%"%w%"`) so that if the word contains a space it will be properly quoted in the generated command-line.

**URL** To interface with an online dictionary source, the sub-list has the following items:

```
|<flags>|<source-name>|<clean-up-macro>|<url>|
```

Where:

**<flags>**
Is a list of option flags, one of which must be `u` to indicate a URL source. The only other flags currently supported by a URL source is `s` for a single, or shared, output buffer, without this flag the buffer name includes the word being described.

<**source-name**>
Defines the name of the interface, or source, which is used in the details given in the first line of the output buffer.

<**clean-up-macro**>
The clean-up macro has two purposes, firstly if the macro aborts (i.e. calls **!abort**(4) etc.) it signals to **describe-word** that the output does not actually contain a definition, for example the output may simply be a network error. The second purpose is to perform any clean-up and reformatting required to make the output more legible. **describe-word** provides a **describe-word-dct** macro for `dict://` URLs which aborts if the output contains less than three lines. It also reformats the output to improve the output in MicroEmacs.

<**url**> Defines the URL required to get a definition, the <**url**> should contain either a `%w` token, which is replaced with the given word in plain text, or `%e` which is replaced with the given word URL-encoded (also known as percent encoded).

**File** To interface with a dictionary text file, the sub-list has the following items:

```
|<flags>|<source-name>|<clean-up-macro>|<regex-search>|<dict-file>|
```

Where:

<**flags**>
Is a list of option flags, one of which must be `f` to indicate a file source. Other flags currently supported by a file source are `k` to keep the dictionary file loaded (improves performance at a cost of increased memory usage), `m` to indication that the dictionary may contain multiple definitions of the same word, and `s` for a single, or shared, output buffer, without this flag the buffer name includes the word being described.

<**source-name**>
Defines the name of the interface, or source, which is used in the details given in the first line of the output buffer.

<**clean-up-macro**>
The clean-up macro has two purposes, firstly if the macro aborts (i.e. calls **!abort**(4) etc.) it signals to **describe-word** that the output does not actually contain a definition. The second purpose is to perform any clean-up and reformatting required to make the output more legible. **describe-word** provides a **describe-word-sln** macro which attempts to convert a single line of definition text into a better laid out set of definitions.

<**regex-search**>
Defines the regex search string required to find the definition of a given word, it must contain a `%r` token which is replaced with the given word (but regex encoded, i.e. '.' to '\.' etc) and at least on group, the first of which must fully contain the definition.

<**dict-file**>
The file name of the dictionary, this can be an absolute file name or given without the path if the file is located in a directory in the MicroEmacs **$search-path**(5).

Each interface sub-list must be wrapped up in the main setup list, which must use a different list separator character.

Note that if spelling has also been setup for the current language, when **describe-word** fails to find a definition it will use the **spell**(2) interface to extract the base word (i.e. base of `words` is `word`) and search for a definition of the base word. When a language is configured to use multiple interfaces, the actual word and then the base is looked-up in the primary interface before moving to the secondary etc.

## EXAMPLE

The following example configures **describe-word** to use the online **dict://dict.org/ WordNet** English dictionary:

```
set-variable .describe-word.enus "!|u|WordNet|describe-word-dct|dict://dict.org/d:%w:wn|!"
```

WordNet is a good modern English dictionary, but it has less than 150K word definitions. To improve the likelihood of finding a definition, a secondary source can be added:

```
set-variable .describe-word.enus "!|u|WordNet|describe-word-dct|dict://dict.org/d:%w:wn|!|u||describe-wo
```

The changed of `wm` to `*` at the end of the `dict://` URL requests the dictionary server to search all available dictionaries. Alternatively, if the **dict**(1) utility is installed, it can be used instead, as follows:

```
set-variable .describe-word.enus "!|c||describe-word-cmd|dict %\"%w%\"|!"
```

Note the enclosing of `%w` with `%"` tokens to ensure the correct encoding/formatting of the command-line.
Alternatively, if the **GoldenDict**(1) utility is installed, the following can be used to simply launch the utility:

```
set-variable .describe-word.enus "!|x|GoldenDict||goldendict %\"%w%\"|!"
```

Note however that MicroEmacs has no way to detect whether a definition was found, so it has to assume it was. As a result this interface should be last as no further interfaces will be used.
If available, the use of a local dictionary text file can greatly improve performance, the dictionary file typically contains the word at the start of a line, followed by a delimiter, such as a tab character, and then the definition in the rest of the line. **describe-word** can be configured to use such a file as follows:

```
set-variable .describe-word.enus "!|fkms|My Dictionary File|describe-word-sln|^%r\t\\(.*\\)|mydict.txt|!
```

Where `mydict.txt` is the name of the dictionary file and is located in the MicroEmacs search path. This could be combined with other sources such as `dict.org` or **GoldenDict**, to create a comprehensive dictionary look-up:

```
set-variable .describe-word.enus "!|fkms|My Dictionary File|describe-word-sln|^%r\t\\(.*\\)|mydict.txt|!
```

The required variables should be added to the user setup file. Notes that, as some interfaces could use '`!`' or '`|`' characters in their configuration, MicroEmacs uses different list separator characters, typically `\f` place of `!` and `\b` instead of `|`.

## SEE ALSO

**spell-word**(3), **spell-buffer**(3), **find-word**(3).

## NAME

file-attrib - Set the current buffers system file attributes

## SYNOPSIS

**file-attrib**

## DESCRIPTION

**file-attrib** opens a dialog enabling the user to change the system properties of the current buffer's file. Top of the dialog give the current buffer name and its file name. The `Save Changes` button writes the current buffer out with any current edits and changes to its file attributes. The `Ok` button closes the file-attrib dialog, any changes made to the file attributes will be applied next time the buffer is written.

The type allow the changing between UNIX, MS Windows and DOS text file formats. UNIX has a single new line character ('`\n`') where as Windows and Dos have a double new line character ('`\r\n`'). Also a Dos text file is terminated with a `C-z` (0x1A) character which the other two do not. These attribute are set in MicroEmacs by using buffer modes **cr**(2m), **lf**(2m) and **ctrlz**(2m).

The central part of the dialog contains system dependent attributes which are defined as follows:

**UNIX Platforms**

Allow the setting of user, group and global, read, write and execute permissions, see man pages on **chmod**(1) for more information. This is a front end to setting the variable **$buffer-fmod**(5).

**Win32 Platforms**

Allow the setting of MS Windows file attributes, i.e. read-only, hidden, archive etc. Note that the directory attribute is displayed but cannot be altered. This is a front end to setting the variable **$buffer-fmod**(5).

**DOS Platform**

Allow the setting of MS Dos file attributes, i.e. read-only, hidden, archive etc. Note that the directory attribute is displayed but cannot be altered.

## NOTES

**file-attrib** is a macro implemented in `fattrib.emf`.

## SEE ALSO

**find-file**(2), **write-buffer**(2), **cr**(2m), **lf**(2m), **ctrlz**(2m), **$buffer-fmod**(5).

Ref: file-attrib(3) File: m3osd008.3 Date: 2022/06/12

## NAME

file-tool-setup - Define file-browser tools

## SYNOPSIS

**file-tool-setup**

## DESCRIPTION

**file-tool-setup** provides a dialog interface for declaring a set of actions to act on files shown in the **file-browser**(3) directory listing. Actions may be defined to interact with `ZIP` files for both creation and extraction, opening of `PDF` files with Adobe Acrobat, opening of Microsoft Word `DOC` files with Word or StarOffice etc. These options are shown in the **file-browser** pop-up menu under **Tools**.

The file tools may be configured from **user-setup**(3) in the **Miscellaneous** tab or invoked directly from the command line using **file-tool-setup**, when invoked then the File Tool Setup dialog is displayed. The dialog allows commands to be be matched against certain file types that are determined using a regular expression, if a filename matches the pattern then a command line or MicroEmacs command may be invoked to process the file(s). The command line includes *variables* which are filled in by MicroEmacs. The information is stored in the registry *user.erf* and is retained across sessions.

The **file-tool-setup** presents a list of tools, there is no limit to the number of tools which can be defined or to the number of tools that may map to a specific file type. The dialog has the following fields:

**Name** Declares the name of the tool as it appears in the **Tools** section of the **file-browser**(3) pop-up menu. A single menu short-cut key may be defined by prefixing the hot key with a `\H`.

**File Mask**

A regular expression that is used to search for files of a given type, this follows the standard MicroEmacs Regular Expression syntax. As an example 'all files' is defined as "`.*`", PDF files are defined as `.*\.pdf`, image files might be defined as a set of different extensions `.*\.\(png\|jpg\|jpeg\|bmp\|gif\)`, etc.

**Command**

Defines the command line or MicroEmacs command that is executed to process the action. The command line comprises escape characters using the percentage (`%`) character, these fields are filled in automatically by MicroEmacs and are defined as follows:

**%%** - Replaced with a single '`%`' character.

**%/** - Path character

Replaced with OS specific path char, i.e. '`/`' on UNIX, '`\`' on Windows etc.

**%;** - Command separator

Replaced with a '`;`' on UNIX or a '`&`' on Windows, used to execute two shell commands. T%he command to the left of the **%&** will be executed first and it's exit code is ignore, the command to the right is then always executed. Note that two ampersands ('**&&**') works in a similar way except

if the exit code of the first (left hand) command is non-zero, indicating an error, the other command is not executed.

**%f** - Filename

A replacement for the filename, typically passed in quotes to the command being executed as an argument. The filename is defined as a relative pathname from the current directory location. If multiple files are selected in the **file-browser** then the command line is executed multiple times, once for each file, unless the **%*** construct is used in conjunction with **%f** when multiple files may be specified with a single command invocation.

**%b** - Base Filename

The basename of the file, i.e. no pathname and no file extension.

**%e** - File Extension

The file extension of the filename.

**%p**, **%P** - File pathname

The pathname of the file without the filename. **%p** is the file path with '/' forward slash directory separators. **%P** is the same as **%p** but with '\' backslashes directory separators on Microsoft Windows and DOS.

**%i[***fh—prompt—default—initial***]** .. **%n** - User Input

Specifies that the user is to be prompted for input from the command line. Square brackets follow the **%i** which defines the command line prompt. The **%i[..]** is replaced with the user input and may be replicated in the command line using **%n** where **n** is a digit `1..9` where **%1** reproduces the first **%i**, **%2** reproduces the second, ... etc.

*f* specifies the input flags and *h* the data type to be entered including the history and semantics allowed, both may be omitted, but when only one is given it is assumed to be *h*. When given, *f* is a single character bitwise flag given in hexadecimal, where each bit has the following meaning:

`0x01` The *default* value will be specified and this will be returned by default. If this this bit is clear the default value is completely omitted so that the *initial* value effectively takes it's place.

`0x02` The *initial* value will be specified and this will be initial value given on the input line.

`0x04` Auto-complete using the initial value, usually used with bit `0x02`.

`0x08` Hide the input string, the characters in the current input string are all displayed as `'*'`s.

If **f** is not specified the flag will be set to 0 or 1 depending on whether a *default* value is given. *h* may have the following values:

`0` For a general string input using the general history.
`1` For an absolute file name, with completion and history.
`2` For a MicroEmacs buffer name, with completion and history.
`3` For a MicroEmacs command name, with completion and history.
`4` For a file name, with completion and history.
`5` For a search string, with history.
`6` For a MicroEmacs mode name, with completion and history.
`7` For a MicroEmacs variable name, with completion and history.
`8` For a general string using no history.

*prompt* is the prompt string presented on the command line.

*default* is the default value that should be used and must be supplied when bit 1 of the *f* flag is set. The **%** characters may be used and will be replaced with information from the filename e.g. **%b** would be replaced with the base file name.

*initial* specifies the initial value for the input buffer, it must be supplied when bit 2 of the *f* flag is set. The **%** characters may be used and will be replaced with information from the filename e.g. **%b** would be replaced with the base file name.

As an example of the user input, consider the following command line which is used to unzip a **.zip** file:

```
unzip "%f" -d %i[4|Unzip sub-dir|%b]
```

Given an input file of `/home/user/foo.zip` then the command line `unzip "/home/user/foo.zip" -d foo` is generated, given that the user accepted the defualt command line prompt of "`foo`" for the prompt "`Unzip sub-dir`"

Where the user input is required in multiple parts of the command line then the **%1** syntax may be used. The following example may be used to create a **.zip** archive from one or more selections in the file list.

```
rm -f "%i[14|Zip file name|%b].zip"%& zip -r %1.zip%*[
"%f"]
```

In this instance the user is prompted for the `Zip file name` and it is used twice in the command line, the default name is the base file name of the first file. Consider two files have been selected `foo.txt` and `bar.txt` where `foo.txt` is first. The generated command line is:

```
rm -f "foo.zip"; zip -r foo.zip "foo.txt" "bar.txt"
```

Note how the *basename* has been used as the default for the zip file name to create and this has been used in two places in the command line, once to delete the old archive if it already exists and once to form the new archive.

**%v[***variable***—***fh***—***prompt***—***default***—***initial***]** - Variable

Inserts the current value of **variable**, if **variable** just contains alpha numeric characters then it is prefixed "`.V`" to create a command variable (it is the user's responsibility to avoid variable name conflicts). The use of **%v** is similar to the use of **%i** except its value persists to the next use, so the default is only ever used once.

An initial value for the variable is required, which means the variable must already be defined or a *default* value must be specified. The arguments *fh* onwards are the same as **%i**'s.

**%v[***variable***—***default***]** - Variable

As above except the user is never prompted for a new value, if the **default** value is omitted the variable must already be define.

**%*[***argument***]** - Multiple file command line

Utilities that accept multiple arguments on the command line use a special construct **%***. By default a single command line invocation is made for each file, the **%*** construct makes a single command line invocation for multiple files.

The square bracket construct contains the expansion for each file specified in the **file-browser**, this will typically include the **%f** construct. For each argument then the brackets are expanded and concatenated together to form a command line, as such the bracket construct usually starts with a space character as this separates the files.

As an example then zip file construction will take multiple files on the command line to add to the archive:

```
rm -f "%i[4|Zip file name|%b].zip"; zip -r %1.zip%*[
"%f"]
```

***%"*... ***%"*** - Quote text

Using quote tags ensures the quoted part of the command-line is treated as a single argument. For example, if the tool:

```
unzip %f
```

is used for the file "Test Files.zip" the command-line becomes '`unzip Test Files.zip`' which will not work correctly, conversely using `"%f"` will not work for files with quotes (e.g. '2"bolt.def'). However, if

```
unzip %"%f%"
```

is used MicroEmacs works out the best way to quote the file name for it to be properly processed.

***MicroEmacs-Command*** - Internal command

If the command starts with a star character (`*`) then the command is treated as a MicroEmacs command and is executed within the editor. As an example consider handling a zip file using the **find-zfile**(3) command:

```
*find-zfile "%f"
```

For each file then **find-zfile** is invoked to read the zip archive.

**Default Open**

A flag that defines whether this is the default open action. When set then the command is automatically launched if the file is opened in the **file-browser** (usually by left clicking on the file name or pressing "`return`") . When clear then the tool has to be explicitly invoked.

**Hide Window**

Defines the visibility of the window associated with the command, for example, if the command-line uses a `cmd` shell to launch another process the cmd window should be hidden. This has no effect on internal MicroEmacs commands.

**Up/Down**

Changes the item order in the list/menu by moving it up or down.

**Add** Adds a new tool line to the list.

**Modify** Modifies the existing tool setting.

**Delete** Deletes the tool from the list.

**Close** Closes the tool menu.

**EXAMPLE**

Example tool definitions:

Word processor
: Name: StarOffice/OpenOffice/Microsoft Word
File Mask: `.*\.\(doc\|ppt\|xls\|odt\|rtf\)`
Command: `soffice %"%f%"`
Hide Window: yes
Default Open: no

Image Viewer
: Name: XnView/Gimp/Photoshop/PaintShop
File Mask: `.*\.\(png\|jpg\|jpeg\|bmp\|gif\)`
Command: `xnview %"%f%"`
Hide Window: yes
Default Open: yes

Postscript Viewer
: Name: Gnome Ghostview/Ghostview
File Mask: `.*\.ps`
Command: `ggv %"%f%"`
Hide Window: yes
Default Open: yes

Acrobat Viewer
: Name: Adobe Acrobat
File Mask: `.*\.pdf`
Command: `acroread %"%f%"`
Hide Window: yes
Default Open: yes

Unzip a zip Archive in current directory
: Name: Unzip in .
File Mask: `.*\.zip`
Command: `unzip %"%f%"`
Hide Window: no
Default Open: no

Unzip a zip Archive in a sub-directory
: Name: Unzip in sub-dir
File Mask: `.*\.zip`
Command: `unzip %"%f%" -d %"%i[4|Unzip sub-dir|%b]%"`
Hide Window: no
Default Open: no

Create a zip archive
: Name: Create Zip
File Mask: `.*`
Command: `rm -f %"%i[4|Zip file name|%b].zip%"; zip -r %"%1.zip%"%*[%"%f%"]`
Hide Window: no
Default Open: no

Add to a zip archive
: Name: Add to Zip
: File Mask: `.*`
: Command: `zip -r %"%i[4|Zip file name|%b]%"%*[ %"%f%"]`
: Hide Window: no
: Default Open: no

View archive file
: Name: View Archive
: File Mask: `.*\.\(zip\|tgz\|tar\.gz\|tar\|gz\|bz2\)`
: Command: `*find-zfile %"%f%"`
: Hide Window: no
: Default Open: yes

Convert files to UNIX
: Name: 2Unix
: File Mask: `.*`
: Command: `me @2unix %*[ %"%f%"]`
: Hide Window: no
: Default Open: no

Microsoft Windows - default action
: Name: Windows Default
: File Mask: `.*`
: Command: `cmd /c %"%f%"`
: Hide Window: yes
: Default Open: no
: Notes: This action runs the default action for the file type.

Microsoft Windows - Explorer
: Name: Explorer
: File Mask: `.*`
: Command: `explorer .%*[]`
: Hide Window: yes
: Default Open: no

## SEE ALSO

**file-browser**(3), **find-zfile**(3), **user-setup**(3), **%os-chars**(5).
Regular Expressions

Ref: file-tool-setup(3) File: m3osd017.3 Date: 2025/07/27

## NAME

file-type-setup - Define file type set

## SYNOPSIS

**file-type-setup**

## DESCRIPTION

**file-type-setup** provides a dialog interface for declaring a set of common files type categories and their associated file extensions. The file type list is used in two places, the main menu's `File => Quick Open` sub-menu and the main `File => Open` **File Type** list. In each case the file type "`All Files`" is automatically added.

**file-type-setup** may be invoked directly from the command line or more usually from the miscellaneous configuration item **File Types** in **user-setup**(3).

The user can add, remove and change the list of file types by using this dialog. An entry can be selected for editing or deletion by simply selecting it with the left mouse button. A new entry may be added by simply filling in the 3 entry boxes and selecting Add. Items in the Dialog are as follows.

`No.` The file type entry number. A new entry is always added to the end of the list, ignoring this value. The position of an existing entry can be changed by altering this field to the desired position and selecting the `Change` button to move it to its new position.

`Name` The file type name, the string printed in the sub-menus.

`File Mask List`
A comma ('`,`') separated list of file masks which match the file type, e.g. for C and C++ source files use "`*.c,*.cc,*.cpp`".

`Add` Adds a new entry to the list, only the **Name** and **FileMask List** fields are used, the **No.** field is ignored as the new entry is always added to the end of the list. The position can be altered by using the **Change** button.

`Modify`
Alters an existing file type entry, all 3 fields must be set.

`Delete`
Deletes the current entry number, only the **No.** entry is used.

## SEE ALSO

**buffer-setup**(3), **user-setup**(3), Top Main Menu.

Ref: file-type-setup(3) File: m3osd015.3 Date: 2024/10/13

## NAME

insert-symbol - Insert an ASCII character

## SYNOPSIS

**insert-symbol**

## DESCRIPTION

**insert-symbol** draws the ASCII character table to the screen, displaying decimal, hexadecimal and character notations in a tabular form. A character is selected using the mouse or cursor characters inserting the selected character into the current buffer at the current position.

## NOTES

**insert-symbol** is a macro defined in `osdmisc.emf`.

The dialog is created using **osd**(2).

## SEE ALSO

**insert-string**(2), **&atoi**(4), **osd**(2).

## NAME

line-scheme - Apply line annotation in current buffer

## SYNOPSIS

*n* **line-scheme**

## DESCRIPTION

**line-scheme** applies or removes line hilighting by assignment of the buffer line variable **$line-scheme**(5). The numerical argument *n* is the line scheme number to apply to the line or region when omitted (or specified as 1) then any line hilighting is removed.

If a region is defined then the command operates as follows:

Region contained on a single line
: When the region is contained on a single line then this is assumed to be a search string and all lines in the buffer that match the region text are hilighted with the color scheme *n*.

Region spans multiple lines
: When the region spans multiple lines then all of the lines of the region are hilighted with the color scheme *n*.

**line-scheme** is typically invoked from the pop-up menu (mouse right) to apply line hilighting annotation to a line or region. The user may select to apply a region hilight, remove the region hilight, remove all hilighting or perform a search using **osd-search**(3).

## NOTES

**line-scheme** is a macro implemented in `search.emf`.

## SEE ALSO

**delete-hilight-lines**(3), **osd-search**(3), **osd**(2), **$line-scheme**(5).

Ref: line-scheme(3) File: m3osd013.3 Date: 2022/06/12

## NAME

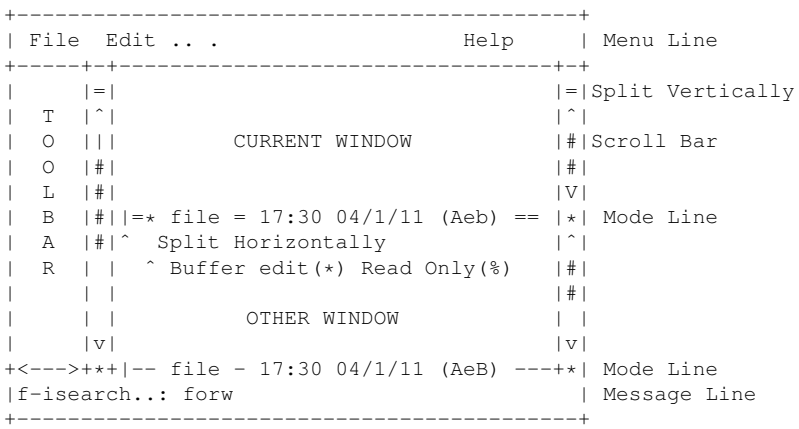
Main Menu - The top main menu

## SYNOPSIS

*n* osd
**toolbar**

## FRAME COMPOSITION

The frame composition presented by MicroEmacs is defined as follows:




**Menu Line**

The *Menu Line* appears at the top of the buffer. A left click of the mouse on the menu line shows a menu dialog for user selection.

**Window**

The windows appear at the center of the frame, typically with a scroll bar to the right of the window and a **Mode Line** at the bottom of the window. The mouse interactions with the window are defined as follows:

Left mouse in window - Selecting a region

Regions may be selected with the mouse. Select a position with the left mouse, keep the left button pressed move the mouse to the desired region end and release the left button. Theregion is hilighted, the text is copied the user does not need to perform any other selection operation to copy the text.

Right mouse in window - Pop up Dialog

A right mouse click in the window brings up a context sensitive pop-up menu. A right click on a spelling error, with **auto-spell**(3) enabled shows a list of possible corrections.

Right mouse on mode line - Pop up Dialog

A right mouse click on the **Mode Line** brings up a pop-up menu to control the buffer.

Left mouse on mode line - split and resize window

If the | character is selected on the **Mode Line** then the window is split into two horizontally. If the mode line is dragged with the mouse then the buffer is re-sized. If the window is dragged into another window then the other window is deleted.

Left mouse on scroll bar - Up/down action
: A left click on the scroll bar moves the window up/down. If the = component of the scroll bar is selected then the window is split into two vertically. Dragging the scroll-bar with the mouse horizontally causes the window to be re-sized. Selecting the $\star$ in the corner of the window allows the window to be resized both horizontally and vertically.

Right mouse on scroll bar - Scroll bar size
: A right click of the mouse on the scroll bar allows the scroll bar width to be altered.

Message Line
: The message line is used from command entry and messages. On entering a command such as **isearch-forward**(2) then the user enters text into the **Message Line** until the command completes. Messages are also displayed in the message line using **ml-write**(2).

Toolbar
: The toolbar window is displayed if toolbar mode is enabled in **user-setup**(3). The mouse interaction is a little different from a conventional mode line, a mouse right click on the toolbar **Mode Line** allows the tollbar to be configured. A Right click on the < and > characters of the toolbar mode line moves to the next or previous tool.

## MAIN MENU

The main menu is provided to give an easier access to parts of MicroEmacs functionality, the menu is not burnt into MicroEmacs but defined on start-up in `me.emf` and `osd.emf`. The **user-setup**(3) command can be used to set whether the menu is always visible and if the Alt-Hotkeys are enabled (i.e. '`A-f`' to open the **File** menu).
The main menu is **osd**(2) dialog number `0` so key bindings can be made which will open the main menu, an argument of `0` will simply open the main menu, an argument of `0x0n0000` will not only open the main menu but also the `n`th sub menu, e.g. to open the edit menu use:

```
0x020000 osd
```

Following is a brief description of the main menu items:

### File Menu

`New`
: Changes the current buffer to a new buffer.

`Open`
: Opens a dialog enabling the user to select files for opening into MicroEmacs. By default the dialog opens the selected file using command **find-file**(2), but if the view option is selected the **view-file**(2) command is used. The binary or encrypt options configure whether the files are to be loaded with **binary**(2m) or **crypt**(2m) modes enabled.

`Quick Open`
: Opens a sub-menu list all user file types (defined in **user-setup**(3)). Selecting one will open another sub-dialog list all files of that type in the current directory, selecting a file will open it using command **find-file**(2).

`Favorites`
: Opens a sub-menu enabling the user to add new favorite files, edit the existing list of favorite files, or select an existing favorite file in which case the file is opened using command **find-file**(2). The favorite file used to store the list is "**$MENAME.**`eff`" and is saved in the **$user-path**(5).

Each favorite file takes 2 lines in the file, the first is the text displayed in the dialog (note that character '\' must be protected with a '\' and the characters '\H' inserted before another character will make that character a Hot key), if the line starts with a '-' then a dividing line is inserted before it in the menu (use '\-' if the line should start with a '-'). The second line is the file name of the favorite file. If this line with a ':' character the rest of the line is a buffer name rather than a file name. If it starts with a '!' character, the rest of the line is processed using **execute-line**(2), this can be used to view help pages or execute processes such as compile etc.

Quicker access to the favorite menu, and the first 9 favorite files in particular, by enabling the **C-#** bindings, see the Start-up tab of **user-setup**(3).

`Close` Closes the current buffer, executes a dialog form of the command **delete-buffer**(2).

`Attributes`
Opens a dialog enabling the user to set the current buffers file attributes, i.e. read and write permissions. See command **file-attrib**(3) for more information.

`Save` Saves the current buffer, executes a dialog form of the command **save-buffer**(2).

`Save As`
Saves the contents of the current buffer as a new file, executes a dialog form of the command **write-buffer**(2).

`Save All`
Saves all changes buffers and session history, executes a dialog form of the command **save-all**(3).

`Reread File`
Reloads the contents of the current buffer from the file system, discarding any changes, executes **reread-file**(3).

`Print Setup`
Opens a dialog which enables the user to configure the printer driver, output location and page layout (executes command **print-setup**(3)).

`Print` Prints the contents of the current buffer to the printer, executes command **print-buffer**(2).

`File Browser`
Launches the file browser which enables the file system to be manipulated and traversed, executes command **file-browser**(3).

`FTP Browser`
Launches the FTP browser, an FTP client, executes command **ftp**(3).

`Load Session`
Loads a new session into MicroEmacs, the current session is saved and closed and the new session is loaded. The user is prompted from the command line to provide the name of the new session to load. See **read-session**(3).

`Exit` Exits the editor, prompting the user to save any unsaved buffers, executes command **save-buffers-exit-emacs**(2).

### Edit Menu

`Undo` Undoes the last edit in the current buffer (executes command **undo**(2)).

`Redo` Redo the last undo, only available immediately after an undo. This is also done via the **undo**(2) command.

`Undo All Changes`
Undo all edits in the current buffer until the last save or no more undo history is available. Executes the command **undo**(2) with a 0 numerical argument.

`Set Mark`
Sets the position of start of a selected region, executes command **set-mark**(2).

`Copy Region`
Copies a marked up region, executes command **copy-region**(2).

`Kill Region`
Deletes a marked up region and places the deleted text in the kill buffer which may be retrieved using **Paste**, executes command **kill-region**(2).

`Delete Region`
Deletes a marked up region but the deleted text is not saved in the the kill buffer and cannot be retrieved using **Paste**.

`Paste Region`
Inserts text from the clipboard into the buffer at the current cursor position, executes command **yank**(2).

`Cut Rectangle`
Deletes a selected rectangular region from the buffer, the vertical extents of the cut are determined by the column position of the start and end of the selected region, executes command **kill-rectangle**(2).

`Copy Rectangle`
Copies a selected rectangular region from the buffer, the vertical extents of the copy are determined by the column position of the start and end of the selected region, executes command **copy-rectangle**(3).

`Paste Rectangle`
Inserts a rectangle region into the buffer at the current cursor position (and for the subsequent lines at the given column position), executes the command **yank-rectangle**(2).

`Insert Symbol`
Dialog presented to select a character symbol for insertion, executes the command **insert-symbol**(3).

`Insert Date/Time`
Inserts the current date and time at the cursor position, a sub-menu is used to determine the format of the insertion string. The list of date formats can be customised by setting the **%date-formats** variable, see macro file **insdate.emf**.

`Insert File Name`
Inserts the name of the file at the current cursor position, executes the command **insert-file-name**(2).

`Insert File`
Inserts the contents of a file into the buffer at the cursor position, the user is prompted for the name of the file to insert. Executes the command **insert-file**(2).

`Draw Lines`
: Moves into line drawing mode, the cursor keys are used to control the line drawing, use **esc-h** for help, **return** to exit draw mode and **C-g** to quit draw mode without making changes. Executes the command **draw**(3).

`Toggle Insert Mode`
: Modifies character entry behavior, moves between overwrite and insertion mode, changes the buffer mode **over**(2m).

`Toggle View Mode`
: Toggles the buffer between view mode and edit mode, in view mode the contents of the buffer may not be altered. Change the buffer mode **view**(2m).

**Search Menu**

`Incremental Search`
: Initiates an incremental search, the user is prompted to enter the string to search for, see the command **isearch-forward**(2).

`Reverse Inc Search`
: Initiates a incremental search backward, the user is prompted to enter the string to search for, see the command **isearch-backward**(2).

`Find` Opens a dialogue for searching the buffer for a given string, see **osd-search**(3).

`Find Next`
: Finds the next string in the buffer using the last search string, uses **hunt-forward**(2).

`Find Previous`
: Finds the previous string in the buffer using the last search string, uses **hunt-backward**(2).

`Replace`
: Opens a dialogue for searching and replacing the buffer for a given string, see **osd-search**(3).

`Hilight Search`
: Opens a dialog which allows lines of the buffer to be searched and highlighted by changing the line color. Setting a line highlight is a temporary change, it does not effect any files and is lost when the buffer is deleted, see **osd-search**(3).

`Find Files`
: Initiates a search for files in the file system from the directory of the current buffer using **find**(3).

`Find in Files (grep)`
: Initiates a string search of the file system, in the directory of the current buffer using **grep**(3).

`Replace in Files`
: Initiates a string search and replace of the file system, in the directory of the current buffer using **replace-all-string**(3).

`Query Replace in Files`
: As with **Replace in Files** except prompts the user before each replace using **query-replace-all-string**(3).

`Set Bookmark`
: Places a bookmark in the buffer which provides a quick method of returning to the buffer position, the user is prompted for a single character to identify the position the buffer. Executes command **set-alpha-mark**(2).

`Goto Bookmark`
: Moves the cursor to a book mark position defined with **Set Bookmark**, the user is prompted for a single character that identifies the bookmark. Executes command **goto-alpha-mark**(2).

`Goto Fence`
: When the cursor is on a fence character (i.e. {, }, `(`, `)`, `[`, `]`, ...) then the other matching fence is located and the buffer position is changed to the other fence character. Executes command **goto-matching-fence**(2).

`Goto Line`
: Moves cursor to the specified line number or relative line position, executes a dialog form of the command **goto-line**(2).

`Goto Tag`
: Only available when a `tags` file is found in the current directory, on selection either the current tag is located and displayed in a pop-up window or a dialog is presented with a list of tags found within the tag file. On selection of a tag then the file is opened and the cursor position is moved to the definition of the tag. See command **find-tag**(2) for more information.

**View Menu**

`Narrow Out`
: Hides the selected region of the buffer, executes command **narrow-buffer**(2) with a numeric argument of `4`.

`Narrow To`
: Hides all of the buffer with the exception of the selected region, executes command **narrow-buffer**(2) with a numeric argument of `3`.

`Remove Single Narrow`
: Removes a single narrowed region of the buffer under the cursor. Executes command **narrow-buffer**(2) with a numeric argument of `2`.

`Remove All Narrows`
: Removes all of the narrowed regions in the buffer, executes command **narrow-buffer**(2) with a numeric argument of `1`.

`Buffer Info`
: Provides information on the buffer current cursor position on the mode line including line number, character under the cursor etc. See **buffer-info**(2).

`Word Count`
: Counts the number of words in the buffer and in the selected region (between point and mark), executes **count-words**(2).

`Occurrences`
: Finds all occurrences of a given string in the buffer. The user is prompted for the string to find. A second window opens and all occurrences of the string in the buffer are presented, selecting an item from this list, using the mouse, moves the cursor to the specified occurrence. See **occur**(3).

`Buffer`
: Lists the current buffers in the editor, selecting one changes the current buffer to the selection.

`Buffer (by type)`
: Similar to the File menu's Quick Open, this opens a sub-menu listing the the user's file types and for each file type a list of buffers matching that type is shown. Selecting a buffer will change the current buffer to the select.

`Main Menu`
: Hides or shows the Main Menu. If the main menu has been hidden then **F1** brings up the menu, the main menu may be re-activated then if necessary.

`Scroll Bars`
: Hides or shows scroll bars within the buffer.

`Tool Bar`
: Hides or shows the tool bar window, see **toolbar**(3).

`White Characters`
: Hides or shows the white space characters in the buffer.

**Format Menu**

`Restyle Buffer`
: Re-styles the layout of the buffer based on the major-mode setting. Executes the command **restyle-buffer**(3).

`Restyle Region`
: Re-styles the layout of a selected region of the buffer based on the major-mode setting. Executes the command **restyle-region**(3).

`Increase Indent`
: Increases the indentation of a line or region by a single single Tab stop position. Executes the command **indent-increase**(3).

`Decrease Indent`
: Decreases the indentation of a line or region by a single single Tab stop position. Executes the command **indent-decrease**(3).

`IQ Fill Paragraph`
: Reformat a paragraph using the current fill mode, handling embedded bullets with no line separation. Executes the command **ifill-paragraph**(3).

`Fill Paragraph`
: Reformat a paragraph using the current fill mode using the standard paragraph fill mode. Executes the command **fill-paragraph**(2).

`Fill All Paragraphs`
: Reformats all paragraphs in the buffer using the standard paragraph filling mode. Executes command **fill-paragraph**(2) with a very large positive numerical argument. Note that this only effects paragraphs from the current position onwards.

`Paragraph to Line`
: Converts the current paragraph to a single line. Paragraphs are typically converted to a single line before copying into an external program, such as an email client or word processor, as these often performs their own text wrapping. It is therefore important that the new line characters in MicroEmacs paragraphs are removed first.

`All Paragraphs to Lines`
: As with `Paragraph to Line` except it converts the current and all following paragraph. Note the conversion only starts from the current position, earlier paragraphs remain unchanged.

`Change Character Set ...`
: Converts the ASCII text from one character set to another, this allows diacritical marks to be maintained.

`Tabs to Spaces`
: Converts `TAB` characters to spaces whilst retaining the correct tab stop positions within the file. Note that if the tab stops appear to be incorrect then the tab width should be temporarily changed using **buffer-setup**(3) before converting to spaces.

`Clean Buffer`
: Cleans up the buffer by removing white space from the ends of all lines, see **clean**(3).

`Capitalize Word`
: Capitalizes the characters from the cursor position, or next word, to the end of the word using **capitalize-word**(2).

`Lower Case Word`
: Changes characters to lowercase from the cursor position, or next word, to the end of the word using **lower-case-word**(2).

`Upper Case Word`
: Changes characters to uppercase from the cursor position, or next word, to the end of the word using **upper-case-word**(2).

`Lower Case Region`
: Lowercase all characters in the current region using **lower-case-region**(2).

`Upper Case Region`
: Uppercase all characters in the current region using **upper-case-region**(2).

**Tools Menu**

`Current Buffer Tools`
: For some file formats MicroEmacs provides a file format specific set of tools, see the file type help page for more specific information.

`Spell Word`
: Spell checks a word entered on the command line, executes the command **spell-word**(3).

`Spell Buffer`
: Spell checks the buffer from the current cursor position onwards, executes command **spell-buffer**(3).

`Word Complete`
: Takes the incomplete word to the left of the cursor and attempts to complete the word by using the users current language dictionary, use `TAB` for a list of possible completions. Executes the command **expand-word**(3).

`Compare Windows`
: Compares the contents of two buffers, the comparison starts from the current cursor position in each buffer and the cursor position should be aligned before starting the comparison, executes the command **compare-windows**(2).

`Diff Changes`
: Compares the current buffer with the last backup version of the buffer, the buffer has to be saved before the comparison may be performed, executes the command **diff-changes**(3).

`Diff`
: Runs a **diff**(3) command to difference two files, the user is prompted each of the files before running the command. The results of the difference are presented in a new buffer window.

`Graphical Diff`
: Runs a graphics difference between two files where the files are presented side by side. The user is prompted for the two files to compare. See also **gdiff**(3) which uses a command line prompt for the files rather than a dialog.

`Compile`
: Starts a compilation or build process, executes the command **compile**(3).

`VCS`
: Opens the submenu of the VCS managing the current buffer, for more more help see **vcs**(3), and the help page for the VCS in use.

`Sort Lines`
: Performs an alphabetical case sensitive sort of the lines in a selected region, the lines are re-ordered in ascending alphabetic order. Executes the command **sort-lines**(2).

`Sort Lines (Ignore Case)`
: Performs an alphabetical case insensitive sort of the lines in a selected region, the lines are re-ordered in ascending alphabetic order. Executes the command **sort-lines**(2).

`Unique Lines`
: Removes duplicated adjacent lines in a region. The region should be sorted before **uniq**(3).

`User Setup`
: Runs a dialog interface to configure MicroEmacs, executes the command **user-setup**(3).

`Buffer Setup`
: Runs a dialog interface to change the session configuration of a buffer i.e. assignment of the major-mode etc. This command changes the current buffer only and does not affect other buffers. Executes the command **buffer-setup**(3).

`Major Mode Setup`
: Runs a dialog interface which allows the major mode to be changed. Executes the command **major-mode-setup**(3).

`Scheme Editor`
: Runs a dialog which allows the current highlighting scheme to be edited, modification of the scheme affects all buffers. Executes the command **scheme-editor**(3).

**Advanced Menu**

`Record Macro`
: Starts recording the key strokes to create a keyboard macro, see **start-kbd-macro**(2).

`Execute Macro`
: Executes a previously recorded keyboard macro, see **execute-kbd-macro**(2).

`Name Macro`
: Allows a keyboard macro to be named, see **name-kbd-macro**(2).

`Execute Command`
: Executes a named MicroEmacs command, the user is prompted to enter the command to be executed, see **execute-named-command**(2).

`Execute File`
: Executes a MicroEmacs macro file, typically used to load and install Macros into the editor, see **execute-file**(2).

`Calculate`
: Invokes a simple command line calculator, see **calc**(3).

`Notes` Invokes the **notes**(3) tool which may be used to organize and store miscellaneous information (i.e. short notes).

`Organizer`
: Invokes the **organizer**(3) tool which provides a calendar and address book.

`File Tool`
: Opens a sub-menu listing all tools applicable to the current buffer's file type, selecting a tool will launch the tool with the current buffer's file.

`Ipipe command`
: Start a incremental shell command, the command runs in the background and the results are collected in a new buffer window, using **ipipe-shell-command**(2).

`Shell` Creates a new shell command window outside of the editor using **shell**(2).

**Window Menu**

`Split Window V`
: Splits the current buffer window into two vertically (stacked), executes the command **split-window-vertically**(2).

`Grow Window V`
: Increases the height of the current buffer window by a single line, executes the command **change-window-depth**(2) with an argument of 1.

`Shrink Window V`
: Reduces the height of the current buffer window by a single line, executes command **change-window-depth**(2) with an argument of -1.

`Split Window H`
: Splits the current buffer window into two horizontally (side by side), executes command **split-window-horizontally**(2).

`Grow Window H`
: Increases the width of the current buffer window by a single character, executes the command **change-window-width**(2) with an argument of 1.

`Shrink Window H`
: Reduces the width of the current buffer window by a single character, executes the command **change-window-width**(2) with an argument of -1.

`One Window`
: Makes the current buffer the only window on the screen, the other buffers are removed from the screen, executes the command **delete-other-windows**(2).

`Delete Window`
: Removes the current buffer window from the screen, other windows are re-organized to occupy the space created by deleting the window, executes the command **delete-window**(2).

`Previous Window`
: Moves focus to the previous window using **previous-window**(2).

`Next Window`
: Moves focus to the next window using **next-window**(2).

`Create New Frame`
: Create an new external frame, only available on version which support multiple-window frames. Executes command **create-frame**(2).

`Close This Frame`
: Closes the current frame, only available on version which support multiple-window frames. The command will fail if this is the only frame, use File -> Exit to exit MicroEmacs, executes command **delete-frame**(2).

`Next Frame`
: Shows the next internal frame, see **next-frame**(2).

**Help Menu**

`Major Mode Help`
: Display the help information associated with the current buffer. For some major mode (file format) MicroEmacs provides a major mode specific help page providing details of key-bindings and tools specific to the current buffers file type.

`General Help`
: Starts the **osd-help**(3) dialog which allows the help pages to the searched.

`List Bindings`
: Lists the key bindings associated with the current buffer, the bindings are displayed in a separate window and include the buffer and global key bindings. Executes the command **describe-bindings**(2).

`List Buffers`
: Lists the buffers currently active within MicroEmacs, the list is presented in a new buffer window and shows the buffer name, size, associated files and current mode. Executes the command **list-buffers**(2).

`List Commands`
: Lists all of the available commands and key bindings in a new buffer window, executes the command **list-commands**(2).

`List Registry`
: Presents the registry hierarchy in a separate buffer window, executes the command **list-registry**(2).

`List Variables`
: Presents a list of all of the global variables and their assigned values in a new window, executes the command **list-variables**(2).

`Command Apropos`
: Produces a list of commands that contain a given string as part of their name, the user is prompted to enter the string to search for. Executes the command **command-apropos**(2).

`Describe key`
: Determines the command that is bound to a specific key, user is prompted for the key sequence, the command bound to that key is displayed. executes the command **describe-key**(2).

`Describe Variable`
: Determine the value of a MicroEmacs variable, user is prompted for the variable name. Executes the command **describe-variable**(2).

`Describe Word`
: A dictionary function that looks up the meaning of a word specified on the command line, executes the command **describe-word**(3).

`Help on Command`
: Provides help on a specific MicroEmacs command, user prompted for the command name. Executes the command **help-command**(2).

`Help on Variable`
: Provides help on a specific MicroEmacs variable, user prompted for the variable name. Executes the command **help-variable**(2).

`Games` Opens a sub-menu listing all available games, see Games for more information.

`Product Support`
: Opens on-line Contact information.

`About MicroEmacs`
: Version and status information on the editor, executes the command **about**(2).

## NOTES

The main menu is defined using **osd**(2) in macro files me.emf and osd.emf.

General user extensions to the main menu can be added to the user file `myosd.emf` which is executed once when the main menu is first opened. The macro file can add new items to any of the main sub menus and can delete most existing items (some are dynamically added when appropriate, these should not be deleted). See `osd.emf` for examples of how to add items to the menu.

New sub-menus should be added in the company or user setup files as this must be done at start-up. The content on the menu is not required until the main menu is used so populating the new sub-menu can be done in `myosd.emf`.

## SEE ALSO

**user-setup**(3).

## NAME

major-mode-help - Displays help page for current buffer

## SYNOPSIS

**major-mode-help**

## DESCRIPTION

**major-mode-help** opens a dialog giving the user a brief help page on tools available for the current buffer. The help page changes depending on the type of the current buffer.

## HISTORY

**major-mode-help** was introduced in the 2006 release, prior to this the command was known as **buffer-help**.

## SEE ALSO

**major-mode-setup**(3).

Ref: major-mode-help(3) File: m3osd011.3 Date: 2022/06/12

## NAME

major-mode-setup - Configures the current buffer settings

## SYNOPSIS

**major-mode-setup**

## DESCRIPTION

The **Major Mode** defines the operational mode of the buffer, controlling highlighting and additional commands that are available. The *Major Mode* is determined by the file type content and is determined from the file extension or *Magic String* in the first few lines of the file.

**major-mode-setup** provides a dialog interface to configuring the setup of the current buffer's file type within MicroEmacs. **major-mode-setup** may be invoked from the main *Tools* menu or directly from the command line using **execute-named-command**(2).

The changes made to a configuration in **major-mode-setup** are maintained in future MicroEmacs sessions by storing them within the user's setup registry file, "<*logname*>`.erf`". Note that not all file types may be supported by **major-mode-setup**, if not the help menu item will not be available.

The contents of the dialog change, depending on the features the current buffer's file type supports. These features are implemented and installed within the buffer's file hook. The following buttons are always present at the bottom of the dialog:

`Tools` Opens the **major-mode-tools-setup**(3) dialog for the current buffer's file type.

`Save` Saves the changes made to the configuration back to the users registry file, i.e. "<*LogName*>`.erf`" but does not re-initialize the current buffer. No changes made will effect the current buffer unless the **Apply** button is pressed. Buffers of the same type created after the save may inherrit some of the changes.

`Apply` Makes the current buffer reflect the changes made, dismissing the **major-mode-setup** dialog. This also performs the above '**Save**' operation. Some changes such as dialog creation changes, will only take effect when MicroEmacs is restarted.

`Cancel`
Quits major-mode-setup, if changes where not **Save**d or **Apply**'d they will be lost.

Following is a list of configurable features which may be available:

Create Help Page
Enables/disables the creation of a help page dialog for the tools available for the current file type.

Create Tools Menu
Enables/disables the creation of a file type specific sub menu located within the main menu's **Tools** sub-menu.

Use Author Mode
For file types which have an automatic formatter/viewer (currently only html) enabling this will simply load the file enabling the source code to be viewed and edited. When disabled files of this type will be automatically processed giving a more readable 'formatted' representation.

Insert New Template
: When creating a new buffer/file of this type, a default template will be inserted if this is enabled. When disabled the buffer will remain empty.

Fence Display
: Enables or disables the displaying of matching fences for this file type. Note that the way in which the matching fence is display is determined by the **Fence Display** option on the Platform page of **user-setup**(3); the **major-mode-setup** option is ignored if this option is set to "`Never Display`".

Setup Hilighting
: Creates and enables the token hilighting for the current file type.

Setup Auto Indent
: Enables automatic formating (indenting) for the current file type. The indentation rules are either the built in 'C' indentation or created explicitly using the **indent**(2) command. When enabled the **tab**(2m) is still adhered to, but the **indent**(2m) mode is ignored; when disabled the indent mode can be used.

Setup Auto Spell
: Enables the setting up of **auto-spell**(3). When enabled the auto-spell key bindings are created and auto-spell is enabled if enabled within the user-setup dialog.

Setup Folding
: Enables the setting up of section folding, when enabled the folding key bindings are created.

Add Abbreviations
: Adds the file type's abbreviation file to the buffer using **buffer-abbrev-file**(2)

Search Modes: Exact
: Enables/disables the **exact**(2m) mode over-riding the setting within the **user-setup**(3) dialog. If this setting is changed the setting within user-setup will be ignored for the current file type.

Search Modes: Magic
: Enables/disables the **magic**(2m) mode over-riding the setting within the **user-setup**(3) dialog. If this setting is changed the setting within user-setup will be ignored for the current file type.

Buffer Modes: Auto
: Enables/disables the **auto**(2m) mode.

Buffer Modes: Backup
: Enables/disables the **backup**(2m) mode.

Buffer Modes: Indent
: Enables/disables the **indent**(2m) mode.

Buffer Modes: Justify
: Enables/disables the **justify**(2m) mode.

Buffer Modes: Tab
: Enables/disables the **tab**(2m) mode over-riding the setting within the **user-setup**(3) dialog. If this setting is changed the setting within user-setup will be ignored for the current file type.

Buffer Modes: Time
: Enables/disables the **time**(2m) mode.

Buffer Modes: Undo
: Enables/disables the **undo**(2m) mode over-riding the setting within the **user-setup**(3) dialog. If this setting is changed the setting within user-setup will be ignored for the current file type.

Buffer Modes: Wrap
: Enables/disables the **wrap**(2m) mode.

## NOTES

**major-mode-setup** is a macro using **osd**(2), defined in `majormod.emf`. This macro configures the user settings of the `hkXXX.emf` files via **buffer-init**(3).

## SEE ALSO

**buffer-init**(3), **major-mode-help**(3), **major-mode-tools-setup**(3), **user-setup**(3). File Hooks.

## NAME

organizer - Calendar and address organizer

## SYNOPSIS

**organizer**

## DESCRIPTION

**organizer** is a calendar and address organizer, enabling notes to be stored against the calendar days; addresses may be archived into an address book.

**organizer** uses the MicroEmacs in-built registry to store information within a registry file called <*username*>`.eof`. **organizer** may be entered directly from the command line, or via the menu (via **Tools**).

**organizer** is displayed within a single osd dialog box, tab selections at the top of the window enable the different forms of information to be displayed. Navigation is typically performed using the mouse, where the mouse is absent then the `TAB` key may be used to move between the fields. The information presented is defined as follows:

### Month

Shows the calendar month, starting with the current month, the current day is hi-lighted and any notes that have been entered are displayed in the **Notes** entry box at the bottom of the page.

The default mode of operation is note entries for the current month, however specifying the <*year*> as the wild card '*' (star) enables annual events to be entered into the organizer. Annual events are automatically inserted into the calendar each year, typically used for birthdays etc.

The entry controls to the dialog are defined as follows:

**<-** Advances to the previous month.

**->** Advances to the next month.

<*Month*>
: A pull down dialog enabling month selection.

<*year*>
: A text entry field specifying the current year as a 4 digit number. A value of * is the wild card year for specifying annual events.

**Notes** A free form text entry box allowing a note to be attached to the currently selected day.

**Save** Saves the entry back to file.

**Month To Buffer**
: Dumps a view of the month to the currently active buffer, any notes are also dumped to the buffer.

**Close** Exits the **organizer**.

### Week

Shows the calendar week in the current buffer, the days of the week are shown in a column ordering. Note that selection of the week is typically performed from the **Month** view, moving

to the **Week** view (via the tab) selects the week appropriate to the previously selected day within the month view.

The entry controls on the dialog are defined as follows:

<- Advances to the previous week.

-> Advances to the next week.

<*year*>
A text entry field specifying the current year as a 4 digit number. The value of * for viewing and setting annual events is not valid in this view.

**Notes** A free form text entry box allowing a note to be attached to the currently selected day.

<*day*> Selecting a date in the *day* column changes the view to the **Day** view.

**Save** Saves the entry back to file.

**Week To Buffer**
Dumps a view of the week to the currently active buffer, any notes are also dumped to the buffer.

**Close** Exits the **organizer**.

**Note:** The start day in the week view may be configured to commence on a day other than Sunday from the **Setup** tab.

**Day**

Shows an extended view of the notes attached to the current day, day selection is typically performed from the **Month** or **Week** views. The entry controls on the dialog are defined as follows:

<- Advances to the previous day.

-> Advances to the next day.

<*year*>
A text entry field specifying the current year as a 4 digit number. A value of * is the wild card year for specifying annual events.

<*month*>
A pull down dialog enabling month selection.

<*day*> A text entry enabling the current day to be entered.

**Notes** A free form text entry box allowing a note to be attached to the currently selected day.

**Save** Saves the entry back to file.

**Day To Buffer**
Dumps a view of the day to the currently active buffer, any notes are printed in the buffer.

**Close** Exits the **organizer**.

### Lists

The lists pane provides support for multiple list generation and manipulation. Each list consists of zero or more ordered items each of which has a text field in which the user can enter information.

Entry to the dialog is defined as follows:

**List** Selects a list.

**New** Creates a new list.

**Lines Per Item**
Sets the number of lines to use when displaying a list item.

**New** Creates a new list item at the end of the current list.

**Up** Moves the currently selected item (left click on the item number) up the list.

**Down** Moves the currently selected item down the list.

**Insert** Inserts a new list item before the currently selected item.

**Delete** Deletes the currently selected item.

**Save** Saves the entry back to file.

**List To Buffer**
Dumps a view of the list to the currently active buffer.

**Close** Exits the **organizer**.

### Address

The address pane provides entry to the address book, enabling personal and business details to be retained against a single name, tabbed selection of **Work** or **Home** selects the information that is displayed. A search engine is provide to locate names within the database, and provision is made to save some text against a name. Entries in the database are, by default, organized by record number, sorting may be explicitly performed from the **Sort** button.

Entry to the dialog is defined as follows:

*<Record No>*
The identity number of the record, a value of * denotes that this is a new record that is being inserted.

<< Moves to the start of the database.

>> Moves to the end of the database, showing record *, a new entry may be entered.

< Moves to the previous record.

> Moves to the next record.

**Name** The name of the individual, entered as *fore-name* and *surname*.

**Nickname**
A pseudo name assigned to an individual.

**Partner**
Shown in the **Home** view only. The *forename* and *surname* of any partner.

**Chld** Shown in the **Home** view only in the **Extended Address Book Mode**. The names of any children (up to 3).

**DOB** Date of Birth, shown in the **Home** view only in the **Extended Address Book Mode**. The dates of birth of the parents, any children in addition to an anniversary date.

**Company**
Shown in the **Work** view only. The name of the company.

**Address**
The address of the individual/company.

**Tel/Fax/Mobile**
Telecommunication information.

**Email/WWW/FTP**
Electronic communication information.

**Notes** Notes associated with the individual.

**Save** Saves the address information to file.

**Dup** Duplicates the currently selected address entry, creating a new record card. Typically used to construct a similar entry for a different individual.

**Delete** Deletes the currently selected entry.

**Addr to Buffer**
Dumps the currently selected address to the current buffer.

**Close** Exits the organizer.

**Find** **find** provides access to a search engine, enabling addresses to be located in the address book.

**Search For**
The string to search for.

**In Field**
Pull-down menu allowing the selection of the field to be searched in.

**Match** Selects how strict the search should be; typically **Any Part** is used as this is the least in-exact search. The default mode is configured in the **Setup** tab.

**Case/magic**
Selects the search criteria. The default mode is configured in the **Setup** tab.

**First** Finds the first record that matches the search criteria

**Next** Finds the next record that matches the search criteria, from the currently displayed record.

**Reverse**
Searches in reverse order.

**Close** Exits the search

**Sort** **sort** provides a mechanism to re-sort the data base into a different order. The sort is performed on up to 3 different keys enabling conflicting primary sort fields to be resolved by the secondary sort criteria. The default sort order is <*Record No*>, <*None*>, <*None*>.

**Sort Keys**
The *Primary*, *Secondary* and *Tertiary* sort fields are selected by a pull down menu. The fields to be used for sorting are selected from the list.

**Sort** Performs the sort, based on the settings of the *Sort Keys*.

**Close** Exits the sort dialog.

### Setup

The **setup** pane configures a number of general settings of the organizer.

**Current Organizer File**
The full pathname of the organizer file. By default this is set to <*userpath*><*userName*>**.eof** and can be altered using **user-setup**(3).

**Change Name**
Allows the displayed name of the month and the day to be modified.

**First Day of the week**
Selects the first day of the week, this sets the first day to be displayed in the **Week** view and the first column in the **Month** view.

**Min New Year Days**
The number of days that must appear in the first week of the New Year for the week to be considered week 1. Modifying the value of this field modifies the week number.

The **Calendar** section allows the wordy representation of the calendar date to be modified. Typically used to modify the names to the native language.

**Change Month Name**
Select the existing month representation from the left-hand box and type in a new selection into the right-hand box.

**Change Week Day Name**
As *Change Month Name*, enables the day of the week representation to be modified.

**First Day Of The Week**
Selects the first day that appears in the **Week** view.

**Minimum Days of New Year in first week**
Specifies the number of days that must appear in the first week of the New Year for the week to be designated as week 1. This value allows the week number to be aligned with the calender weeks of standard diaries. The default value is 7 days; but may be reduced to 5 or 6 for typical alignment.

The **Address Book** section allows the operation of the address book to be modified.

**Use Extended Address Book**
The extended address book allows additional information to be added to the personal address book. The extended information is limited to the amount of personal information attributed to an individual, including *Date of Birth* and *Child* information.

**Import From File**

The **Import** from file allows the address book to be imported from a file. The import data format is a single line per entry, comma , separated. The field order is defined as follows, the * entries indicate the **Extended Address Book** fields:

*Record No*, *First Name*, *Surname*, *Nick Name*, *Selected*, *Notes*, *Partner First Name*, *Partner Surname*, *Home Address*, *Home Telephone*, *Home Fax*, *Home Mobile*, *Home E-Mail*, *Home WWW Page*, *Home FTP Site*, *Work Company*, *Work Address*, *Work Telephone*, *Work Fax*, *Work Mobile*, *Work E-Mail*, *Work WWW Page*, *Work FTP Site*, *Date-Of-Birth**, *Partner DOB**, *Date-Of-Marriage**, *Child1 Name**, *Child1 DOB**, *Child2 Name**, *Child2 DOB**, *Child3 Name**, *Child3 DOB**.

**Export To File**

Exports the address book to a file, the address book is exported in the current sort order, with the fields defined as above. The exported address book may then be imported into a 3rd party package i.e. Microsoft Access, etc.

The **Default Address Find Settings** section defines the default search criteria used in the address book search function.

**Whole/Start/Any Part**

Radio buttons determine how the search is performed on the string.

- **Whole** matches the whole string exactly.
- **Start** matches the first part of the string only (i.e. `Ab*`).
- **Any Part** finds entries that include the search string at any position within the data base search field.

**Case Insensitive**

Checked, matches the strings regardless of case. (default).

**Magic Mode**

Allows magic strings to be included in the search string.

## NOTES

**organizer** is a macro that is implemented in `organiz*.emf` files. Organizer uses **osd**(2) to create and manage the dialogs.

The maximum size of a text note is 1024 characters.

With an new address is created it is added to the end of the address list regardless of the current sort criteria.

**Organizer** replaces the original **Calendar** utility.

## SEE ALSO

**notes**(3), **user-setup**(3), **osd**(2).

## NAME

osd-help - GUI based on-line help

## SYNOPSIS

**osd-help**

## DESCRIPTION

**osd-help** provides a GUI front end to the on-line help manual, the dialog consists of 3 pages which are defined as follows:

### Contents

The contents page displays a list on contents similar to the **help**(2) high level help page. Selecting an item will display the help page in a buffer, selecting **Exit** will exit the dialog.

### Index

The index page gives a list of help items, the **Scope** menu can be used to narrow the index list to the required item type.

### Search

The search page provides a way of searching the on-line help for a given topic. Similarly to the Index page, the **Scope** menu is provided to narrow the search to the required area.

The search strings is considered to be made up of items separated by spaces, an item can be enclosed in quotes (’**"**’) so that the item can include a space. If the first letter of an item is a ’+’ the given item must be found in a page for it to match, if the character is a ’−’ the item must NOT be found on a page for it to match, or other items are considered optional. At least one item must be found on a page for it to be a match, the numbers to the right of each found page is the number of items found.

## NOTES

See Help! for help on the on-line help pages.

**osd-help** is a macro using **osd**(2), defined in osdhelp.emf.

## SEE ALSO

**help**(2).

Ref: osd-help(3) File: m3osd007.3 Date: 2022/06/12

## NAME

osd-search - Search and annotate the current buffer

## SYNOPSIS

**osd-search**

## DESCRIPTION

**osd-search** is a hidden macro which provides a method of searching for text patterns within the current buffer and annotating any matches through colored line hilighting. A selection of line colors are provided to allow different search patterns to be assigned their own color.

**osd-search** is generally used for annotating log files and alike, where indevidual lines are of interest in addition to the context about that line. The hilighting draws attention to the line, by providing a visual cue, allowing the contents of the file to be breifly scanned.

On invocation of **osd-search** from the menu **Search** -> **Hilight Search** a **osd**(2) dialog is presented to the user, search patterns and their associated hilighting assignment are selected through this interface. The dialog entries are defined as follows:

**Search for**
: The text dialog entry box allows the search pattern to be entered. This may be a regular expression or plain text.

**Color**
: The **Color** allows the line hilighting color scheme to be selected from a pop-up menu. The color **Remove** is special and allows previously applied line hilighting to be removed.

**Case Sensitive**
: A check box that allows the search to be case sensitive or insensitive. This modifies the **exact**(2m) mode.

**Magic Mode**
: A check box that enables/disables regular expression pattern matching. This modifies the **magic**(2m) mode.

**Below**
: Searches and hilights lines matching the search pattern from the current cursor position to the end of the buffer.

**Above**
: Searches and hilights lines matching the search pattern from the current cursor position to the top of the buffer.

**All**
: Searches and hilights lines matching the search pattern for the whole buffer.

**Clear All**
: Removes all line hilighting from the current buffer.

**First**
: Moves to the top of the buffer and hilights the first line that matches the search pattern.

**Next**
: Hilights the next line that matches the search pattern.

**Reverse**
: Hilights the previous line that matches the search pattern.

**Exit**
: Exits the hilighting search dialog.

## NOTES

**osd-search** is a macro implemented in `osdmisc.emf`.

## SEE ALSO

**delete-hilight-lines**(3), **line-scheme**(3), **osd**(2), **$line-scheme**(5).

Ref: osd-search(3) File: m3osd004.3 Date: 2022/06/12

## NAME

print-setup - Configure MicroEmacs's printer interface

## SYNOPSIS

**print-setup**

## DESCRIPTION

**print-setup** provides a dialog interface for configuring MicroEmacs's printing interface. **print-setup** may be invoked from the main *File* menu or directly from the command line using **execute-named-command**(2).

The **print-setup** dialog is broken down into three pages of configuration options, the following buttons are available at the bottom of each dialog:

`Print` Prints the current buffer using the current configuration.

`Exit` Quits **print-setup**, changes made to the configuration are saved.

The following pages appear in the dialog:

### Printer

The **Printer** page is used to configure the *type*, *style* and *location* of the printer, the items on this page are defined as follows:

`Driver`

Sets the printer type to be used, selecting this item creates a drop down list of available printer drivers. The drivers inform MicroEmacs which fonts and colors are available and how to enable/disable them, these are usually special character sequences. The following special drivers are defined:

Default Plain Text
: This driver does not use any special character sequences so the output it produces is plain text. This should work with most printers, but it does not support any colors or fonts.

HTML
: This is a virtual printer driver as no printer uses HTML directly. However the files produced by this driver can be loaded by a web-browser and rendered with full color and font support so provides an efficient way of testing printer schemes. In addition may be used to convert the text rendered in MicroEmacs into HTML content.

Windows
: This utilizes MicroEmacs's built-in Windows printer interface (Windows platforms only). When selected MicroEmacs communicates directly to the MS Printer Manager.

`Print Scheme`

Defines the color and font scheme to be used, selecting this item creates a drop down list of available printer schemes - choose the most appropriate one for your printer. The *Default Plain Text* scheme does not use any color or fonts and should work for all drivers. See the next item for scheme creation and editing.

`Edit` Opens the **scheme-editor**(3) dialog box to edit the currently selected printer scheme. The editor may be used to create and install new printer schemes.

`Destination`
Specifies the destination print output, when selected a drop down menu appears with the following items:

To buffer only
: Creates a "`*printer*`" buffer and prints to that buffer.

To file only
: Creates a new temporary file and prints to it.

To file & print
: Prints to a temporary file and then executes the `command-line` (see next item) to print the resultant file. This option is not available when using the Windows printer driver.

Direct to printer
: Output is sent directly to the printer. The option is only available when using the Windows driver.

`Command-line`
Sets the command-line required to print a generated print file (option not available when the Windows driver is selected as printing is performed directly by the MS Print Manager). The command-line should be a single shell command using "`%f`" to reference the name of the file to be printed, e.g. on UNIX systems **lp**(1) or **lpr**(1) may be used as follows:

```
lp -s %f
```

On MS-DOS machines this may be performed by copying the file to the `PRN` device, as follows:

```
copy %f PRN
```

`Page Layout`
Displays the current page configuration in the form:

*Columns*x*Rows Chars-Wide*x*Chars-High*
: the fields cannot be edited directly, the settings in **Page Setup** affect these values.

**Page Setup**

`Paper Size`
Defines the size of the printer paper, selecting this item produces a pop down menu listing of all available paper sizes. If the the Windows printer driver is being used this field cannot be selected and the **Edit** button is used instead.

`Page Size`
Defines the printable size of the page in terms of the number of characters which will fit on the paper (*width*x*height*). When selected a drop down menu lists all available sizes for the current paper size. This option is not available with the Windows driver and the **Edit** button should be used.

`Edit` (Windows only)
Opens a Windows printer dialog box allowing the user to specify the windows printer, paper size and character size, etc.

`No. of Columns` and `Rows`
: Defines the page subdivision in terms of sub-columns and rows, thereby creating sub-pages within a page.

`Print Line Numbers`
: When enabled, prints the line number at the left hand edge of each line.

`Print Split Line IDs`
: When enabled the last right hand text column is reserved for a split line identifier. Whenever a line is too long to fit on a single line it is split over two or more lines, if this option is enabled the right edge will be set to the split character (usually a '\' char) to clearly indicate that the line is split.

`Page Layout`
: As with the **Printer Page Layout**, displays the current page configuration, the field cannot be edited.

### Layout

`Margins`
: Configures the top, bottom, left and right margins of the page in terms of characters.

`Print Header`
: Defines whether a header should be printed and its format. The following escape sequence is interpreted:

| | |
|---|---|
| `%%` | Print a '`%`' character. |
| `%b` | Print the current buffer's name. |
| `%D` | Print the current day of the month. |
| `%f` | Print the current buffer's file name. |
| `%h` | Print the current hour. |
| `%M` | Print the current month of the year. |
| `%m` | Print the current minute of the hour. |
| `%p` | Print the current page number. |
| `%s` | Print the current seconds. |
| `%Y` | Print the current year as a 2 digit number. |
| `%y` | Print the current year as a 4 digit number. |

`Print Footer`
: Defines whether a footer should be printed at the end of the page and the format. The header escape sequence characters are used.

### NOTES

**print-setup** is a macro using **osd**(2), defined in `printstp.emf`.

The list of available printer drivers and print schemes is stored in the macro file `printers.emf`. To create a new printer driver a new configuration registry file (`erf` file - see `print*.erf` for examples) must be created and added to the printer driver lists within `printer.emf`.

The `printers.emf` file also contains a list of printing schemes, using the **Install** option of the **scheme-editor**(3). **scheme-editor**(3) automatically adds the new scheme to the print schemes list.

## SEE ALSO

**print-buffer**(2), **scheme-editor**(3), **osd**(2).

Ref: print-setup(3) File: m3osd012.3 Date: 2022/06/12

## NAME

scheme-editor - Color Scheme Editor

## SYNOPSIS

*n* **scheme-editor** [ "*scheme-file-name*" ]

## DESCRIPTION

**scheme-editor** is a color and font scheme editor that provides a dialog interface to configure the display schemes used by the editor. The schemes may be created or modified within the scheme editor and then committed to the configuration files for general use.

The editor can be used to create both screen and printer color/font schemes, they are typically stored in the `macros` directory and are executed as macro files at start up or when printing. The standard screen schemes are called `scheme`*X*`.emf` and printer ones `print`*X*`.emf`.

If the *scheme-file-name* is not explicitly given then the scheme currently being used will be edited (or the previously edited scheme if a numeric argument of `0` is given).

The **scheme-editor** is displayed within a single dialog box, tab selections at the top of the dialog box enable **color** and **scheme** creation and/or modification. Navigation is typically performed using the mouse, where the mouse is absent then the `TAB` key may be used to move between the fields. The information presented is defined as follows:

**File Name**
: The name of the color scheme to be modified. This is the name of the **scheme***X***.emf** file, omitting the file extension. See the **FILES** section below for a list of standard screen and printer scheme supplied with MicroEmacs.

**Type** Defines whether the scheme is a screen or printer type.

**Description**
: An ASCII description of the color scheme, used to identify the color scheme.

**Buffer Hilight**
: Available when scheme is a screen type. Defines whether buffer hilighting should be enabled, when *Completely Disable* all buffers are displayed character for character in the standard text scheme, this will ensure maximum update performance but some file formats such as the on-line help will become unreadable so this option is really selected. Similarly *Reformat Only* disables the majority of buffers, hilighting is only enabled when the file would be unreadable without it, such as the on-line help or man page files. The default *Fully Enabled* setting enables all buffer hilighting.

**Print Option**
: Available when scheme is a printer type. Defines what components of a scheme is to be used when printing.

### Colors

The **colors** tab allows the basic palette colors of the editor to be created and modified. The left-hand side of the dialog contains a scrolling window containing the existing color entries. The right-hand side of the dialog provides the controls to add and change the color assignment. The controls operate on the currently selected palette entry.

**Cursor** Defines the colour of the text cursor; the first colour, *Normal*, defines the colour to be used when the current buffer does not have **over**(2m) mode enabled. The second colour, *Over*, is only used when the current buffer has **over** mode enabled.

**Show in Dialogs**
Defines whether the cursor should be displayed in the main OSD dialogs, this improves usability of the dialogs when limited colors are available. Indirectly sets bit `0x20` of **$system**(5).

**Red/Green/Blue**
The color entries allow the currently selected palette color entry to be modified. The color values may be changed by direct numeric entry (0..255) or via the ^/v controls; the color is committed to the palette using the **Add** or **Change** button.

**Change** Commits changes to the currently selected color in the palette.

**Add** Creates and adds a new color entry into the palette. The new palette entry is created with a default color that may be subsequently modified.

### Schemes

The **schemes** tab allows the schemes to be edited. The left-hand side of the dialog contains a scrolling window of the available color palette (created from the **Colors** tab). The right-hand side of the window shows the variants of the scheme.

**Selection**
The **selection** item provides a pull-down menu containing gross scheme categories used by the editor.

**Scheme** A pull-down menu containing the schemes of the selection, modifying this entry shows the variants of the scheme in the **Normal**, **Current**, **Select** and **Sel-Cur** dialogs.

**Reset** Resets the current scheme back to the default *Standard Text* scheme.

There are 4 variants, or styles, for a single scheme; each style is comprised of a foreground and background color, and a row of toggle button to enable/disable fonts, defined as follows.

`B` - Bold.
`I` - Italic.
`L` - Light (typically not supported).
`R` - Reverse video (fore/back-ground swapped).
`U` - Underline.
`V` - Toggle reverse video when inverted.

The last mode **V** needs a little more explanation; commands such as **screen-poke**(2) are able to invert the color scheme, i.e. use the fore color for the background etc. Enabling this mode will toggle the reverse video mode (**R**) when this feature is used. The style displayed by a particular scheme depends upon the selection/current status of the text:

**Normal** The normal style, when the text object is not selected or current (i.e. out of focus).

**Current**
The style used when the text object is current (i.e. in focus)

**Select** The style used when the text object is selected (i.e. by the mouse) and is not current.

**Sel-Cur**

The style used when the text object is selected and is current.

Note that a printer scheme only uses the Normal style.

Setting of the **selection** and **scheme** shows the current scheme in the **Normal**, **Current**, **Select** and **Sel-Cur** dialogs. New colors are assigned by selecting a color in the palette area and making it current. The current color is applied by selecting the **Fore / Back** boxes of the scheme dialog. The assigned color is displayed in the text box *The big brown fox....*

### Controls

The controls at the bottom of the dialog apply the edits to the configuration files.

**Current**

Makes the changes to the palette and schemes current, they are applied to the current editing session but are not committed to file. This allows the palette changes to be used prior to commitment. Note that all modifications are lost if they are not saved and the editing session is terminated.

**Save** Saves the scheme modifications to file, effectively making the changes permanent. Note however that the scheme macro file will be saved in the first directory in the **$search-path**(5), regardless of the location of the original. For network systems this typically means that the changes will only effect the current user.

**Install** Installs the current color scheme into the configuration files, making the color scheme accessible to the **user-setup**(3) dialog.

**Exit** Quits the scheme editor without modifying the settings.

### FILES

`scheme.emf` - Defines the standard scheme variables, including the available scheme list, and associated text.
`schemed.emf` - Default white on black color scheme.
`schemej.emf` - Black on cream color scheme.
`schemevi` - Sandy shores.
`schemesf` - Sherwood Forest.
`schemebh` - Blue Hue.
`schemepd` - Plain Black on Cream.
`schemepl` - Plain White on Black.
`schemel` - Black on grey.
`schememd` - Microsoft Developer Studio Colors.
`printers.emf` - Defines the list of available printer schemes and drivers.
`printd` - Default plain print-out.
`printf` - Print using fonts.
`printepc` - Print using Epson base colors and fonts.

### NOTES

**scheme-editor** is a macro that is implemented in file `schemosd.emf`. The scheme editor uses **osd**(2) to create and manage the dialogs.

Only the Normal scheme style is used by printer schemes.

The setting of **Buffer Hilight** can effect the way buffer hooks are load so changing from one scheme to another with differing Buffer Hilight settings may not fully work. This can be rectified by restart MicroEmacs with the new scheme as default.

The current screen scheme can effect the printing due to the **Buffer Hilight** setting, e.g. if the screen scheme is set to completely disable hilighting then any print-out will also have no hilighting.

## SEE ALSO

**user-setup**(3), **add-color-scheme**(2), **print-scheme**(2), **osd**(2).

Ref: scheme-editor(3) File: m3osd005.3 Date: 2025/09/05

## NAME

shell-tool-setup - Define shell tools

## SYNOPSIS

**shell-tool-setup**

## DESCRIPTION

The **shell-tool-setup** dialog allows the user to configure up to 10 system commands, or tools, which can be executed via MicroEmacs Main Advanced Menu. The dialog configures the user's registry for the command **execute-tool**(3) to be used. The execution of a tool can also be bound to a key, see **execute-tool** for more information.

The top half of the dialog consists of the 10 Tools (`0-9`) configuration buttons. Selecting one of these selects the current tool to be configured, the current tool is shown by the title in the middle of the dialog.

The lower half of the dialog configures the currently selected tool, as follows:

`Tool Name`
: Sets the displayed name of the tool. The tool name is used in the buttons in the top half of this dialog and in the MicroEmacs Main Tools Menu.

`Command Line`
: Sets the system command-line to be launched whenever the tool is executed, the following special tokens may be used in the command-line which are substituted at execution:

  **%ff** The current buffer's full file name, including the path.

  **%fp** The current buffer's file path.

  **%fn** The current buffer's file name without the path.

  **%fb** The current buffer's file base name, i.e. the file name without the path or the extension.

  **%fe** The current buffer's file extension with the '.' (e.g. "*.emf*"), set to the empty string if the file name does not have an extension.

  Note that "**%ff**" is always the same as "**%fp%fn**" and "**%fp%fb%fe**". If any of these tokens are used, the tool will fail to execute if the current buffer does not have a file name.

`Run Concurrently`
: If enabled, when the tool is executed the command is launched and run concurrently, allowing the user to continue working in MicroEmacs during the tools execution. This option is not available for all versions on MicroEmacs and forces the output to be captured. Enabling this option will force the use of command **ipipe-shell-command**(2) to launch the tool.

`Reread Current Buffer`
: When enabled the current buffer is reloaded when the command completes, this option is only available when **Run Concurrently** is disabled.

`Capture Output`
: If enabled any output produced from the execution of the tool will be captured and inserted into a new buffer. When enabled the following two items, `Buffer` and `Hide`, may be specified. When disabled the command used to execute the tool is **shell-command**(2), otherwise the command used is either **pipe-shell-command**(2) or **ipipe-shell-command**(2) depending on the setting of `Run Concurrently`.

`Buffer`
: Specifies the buffer name the captured output should be dumped to, this option is only visible when `Capture Output` is enabled. The following special tokens may be used in the buffer name which are substituted at execution:

  **%fn** The current buffer's file name without the path, set to the buffer name if the current buffer does not have a file name.

  **%fb** The current buffer's file base name, i.e. the file name without the path or the extension. Set to the buffer name if the current buffer does not have a file name.

  **%fe** The current buffer's file extension with the '.' (e.g. "*.emf*"), set to the empty string if the current buffer does not have a file name or it does not have an extension.

  Note that "**%fn**" is always the same as "**%fb%fe**". Default buffer name when this field is left empty is "**command*", or "**icommand*" if `Run Concurrently` is enabled.

`Hide` When enabled the tool output capture buffer is hidden, this option is only visible when `Capture Output` is enabled.

Current Buffer - `Save` and `Prompt Before Saving`
: If the current buffer has been edited, enabling `Save Current Buffer` will automatically save the current buffer before executing the tool. This is particularly useful when the tool operates on the current buffer's file (e.g. compiles the file). If `Prompt` is also enabled the user will be prompted before the file is saved.

All Buffers - `Save` and `Prompt Before Saving`
: If `Save All Buffers` is enabled, all edited buffers will be automatically saved before executing the tool. This is particularly useful when the tool may operate on multiple files (e.g. compilation of a project). If `Prompt` is also enabled the user will be prompted before each file is saved.

## SEE ALSO

**execute-tool**(3), **user-setup**(3), Top Main Menu.

Ref: shell-tool-setup(3) File: m3osd018.3 Date: 2024/10/13

## NAME

spell-buffer - Spell check the current buffer
spell-word - Spell check a single word
spell-edit-word - Edits a spell word entry
find-word - Find a word using spelling dictionaries

## SYNOPSIS

**spell-buffer**
*n* **spell-word** ["*word*"] (**esc $**)
**spell-edit-word** ["*word*"]
**find-word** ["*word*"]

## DESCRIPTION

MicroEmacs provides an integrated spell checker with the following features:

- Different languages.
- Dialog control of the speller.
- Best guess capability.
- *Replace* and *Replace all*, *Ignore* and *Ignore All*
- Undo capability.
- Adding new words and endings to speller.
- Auto correct of commonly occurring mistakes.
- Word finder, allows words to be searched with wild cards.

**spell-buffer** spell checks the current buffer, from the current position, to the end of the buffer. On invocation, an **osd**(2) dialog is opened and any corrections are made through this interface. If an error dialog opens instead the current language is not setup, please see the Language setting in **user-setup**(3) and Locale Support.

The dialog provides the user with an interface from which a new spelling may be selected, in addition new words may be added to the spelling dictionary. The dialog entries are defined as follows:

**Word** The **word** entry contains the erroneous word, this is presented in a text dialog box which may be manually edited to correct. If the word is manually corrected, then it is spell checked prior to insertion, and a new guess list is created. The user may elect to replace the word, take one of the suggestions or re-edit the misspelled word.

**Meaning**
The meaning button provides a convenient interface to **describe-word**(3) for looking up the meaning of the current word. The **Insert** button within the describe-word dialog will replace the current word in the spell-buffer.

**Suggestions**

The suggestions entry contains a list of suggestions as to the correct spelling of the word. The list is ranked in order of the best match, typically the misspelled word appears at (or near) the top of the list, unless the word is unknown or there are gross errors in the spelling. Selecting the word in the list with a single click of the mouse selects the word as the replacement, the actual replacement is performed by the **Replace** or **Replace All** buttons. Alternatively, double selecting a guess word replaces the word.

**Language**

The **language** entry allows the user to select the current spelling language. The new language is chosen from the dialog box. The language may be changed at any time during the spell operation and is effective immediately. The **Ext** languages are extended dictionaries that contain additional words, it is recommended that all spelling is performed with the extended dictionaries (where available).

**Replace**

The **replace** button is activated when a new word has been edited or selected as a candidate for replacement. Selecting **replace** modifies the erroneous word in the buffer with the newly selected word.

**Replace All**

The **Replace All** button is similar to the **Replace** button, except that it automatically replaces any subsequent occurrences of the erroneous word with the newly selected word. The replacement words are retained for the MicroEmacs edit session and are lost when the editor is closed.

**Ignore** The **ignore** button requests that the speller ignore the erroneous word and continue to spell the buffer.

**Ignore All**

The **Ignore All** button is similar to the **Ingore** button, except that it automatically ignores the erroneous word thereafter. The ignore words are retained for the MicroEmacs edit session and are lost when the editor is closed.

**Add** **Add** adds the current erroneous word to the dictionary, thereafter the word is recognized as a valid word. **Add** should only be used for words which have no derivatives, it is generally better to add a new word through the **Edit** interface where a new *base* word may be specified with it's derivatives.

**Edit** The **Edit** button executes **spell-edit-word** giving the current erroneous word. This allows new words and auto-corrections to be defined as well as existing words to be altered, see full description below.

**Find** The **Find** button executes **find-word** giving the current word as a starting seed. This allows the user to search for the word using a simple search criteria, see full description below.

**Undo Last**

The **undo Last** button restores the user to the previous spelling so that it may be re-entered, any replacement text that was made is restored to it's original spelling.

**Close** Exits the speller and returns the user to the buffer.

**spell-word** checks a single word which is either supplied by the user, or if an argument is given, the word under (or to the left of) the cursor position. If the word is correct, a

simple message-line print-out is given, otherwise the main spell **osd** dialog is opened and the user may check the spelling within the context of the spell dialog as described above.

The default key binding of "`esc $`" supplies an argument forcing **spell-word** to check the current buffer word. **spell-word** is often used to check the spelling of a word outside of the context of the editor (i.e. when working on paper, or when doing at that prize crossword !!).

**spell-edit-word** allows words in dictionaries to be altered as well as new words and auto-corrections to be defined. On invocation, an **osd** dialog is opened and changes are made through this interface, defined as follows:

**Word** The **word** entry to be changed or added. If **spell-edit-word** was executed via spell-buffer **Edit** button, this will be set to the current word.

**No word set**
The word entry is empty, most of the functionallity will not be available until a word is entered.

**New Word**
To add a new word, the derivatives of the new word should be selected using the prefix and suffix options. Note that not all derivatives are listed, only one example derivative of each spell rule is given.

**BE CAREFUL WITH THE CASE OF THE BASE WORD:** new words that are entered are case sensitive, as a general rule the *word* in the **Word** text box should be edited to it's base form and should be presented in lower case characters (unless it is a proper name, in which case it should be capitalized, or is an abbreviation or acronym when it might be upper case).

When the appropriate derivatives of the new word have been selected, it may be added to the dictionary using the **Add** button. This adds the word to the users personal dictionary. Please note that if there are numerous standard words missing then check that an *extended* dictionary (designated by **Ext** in the language) is being used, the extended dictionaries more than double the repertoire of words available.

Words added to the dictionary may be subsequently removed using the **Delete** button, typing the existing word in the **Word** entry and selecting **Delete** button removes the existing word.

**Auto-Correct**
Selection of the **Auto-Correct** button allows a replacement word to be entered in the **To** text entry. Selecting **Add** adds the automatic correction to the speller. Thereafter, whenever the erroneous word is encountered the replacement word is always used to replace it, without user intervention.

Entering an exiting *auto-correct* word into the dialog and selecting **Delete** removes an existing auto-correct entry.

**Close** Exits the **Edit** dialog.

**find-word** opens the word finder dialog. This allows the user to search for a word using a simple search criteria. (This is ideal for cheating at crosswords !!). The word to be searched for is entered into the **Word Mask** and may use wild cards * to

represent any number of characters, **?** to represent an unknown character and **[..]** for a range of characters.

For example, searching for `t?e?e` presents the list *theme*, *there* and *these*. Searching for `t*n` lists all of the words beginning with `t` and ending in `n`. See **$find-words**(5) for a full discription of the format used by search engine.

The words that match are returned in the scrolling dialog, and may be selected with the mouse (or cursor keys). The **Insert** button inserts the selected word into the current buffer or into the **Word** entry if executed from the **spell-buffer** dialog. Note that the list presented is limited to 200 words, selecting **next** gets the next 200 words, and so on. The **Close** button exits the dialog.

## NOTES

The words added to the speller during a MicroEmacs session are saved when the editor is closed. The user is prompted to save the dictionary, if the dictionary is not saved then any words added are lost.

All *ignore* words accumulated during a spell session are lost when the editor is closed. In order to retain *ignore* words, it is suggested that they are added to the personal dictionary rather than be ignored.

The personal spelling dictionary is typically called <*user*><*type*>.edf, and is stored in the default user location. The dictionary names are specified in the **user-setup**(3) dialog.

**find-word** may claim to have found more words than are actually listed. The use of derivatives in the spell algorithm allows a single word to be present several times. **find-word** counts each occurrence but it is only listed once.

## SEE ALSO

**user-setup**(3), Locale Support, **osd**(2), **spell**(2), **describe-word**(3), **$find-words**(5).

Ref: spell-buffer(3) File: m3osd002.3 Date: 2025/08/28

## NAME

user-setup - Configure MicroEmacs for a specific user

## SYNOPSIS

**user-setup**

## DESCRIPTION

**user-setup** provides a dialog interface to enable the user to configure the editor. **user-setup** may be invoked from the main *Help* menu or directly from the command line using **execute-named-command**(2). **user-setup** configures the user's setup registry file, "<*logname*>`.erf`" which is used by MicroEmacs to initialize the environment to a user's preference.

Note, if your screen is too small to display the whole dialog, it may be moved using any key bound to the scroll commands such as **scroll-up**, e.g. `A-up`, `C-z`, `A-down`, `C-v`, `A-left` etc. For systems without mouse support, the `tab` key may be used to move between fields.

On all pages the following buttons are available at the bottom of the dialog and have the following effect:

`Save` Saves the changes made to the users registry file, i.e. "<*Log-Name*>`.erf`" but does not re-initialize MicroEmacs. Some changes, such as color scheme changes, only take effect when the **Current** button is used or when MicroEmacs is restarted.

`Current`
Makes the current user and the changes made Current to this MicroEmacs session, dismissing the **user-setup** dialog and reinitializing MicroEmacs. This also saves the registry file out!

`Cancel`
Quits **user-setup**, if changes where not **Save**d or made **Current** they will be lost.

The following pages, which appear in the dialog, are defined as follows:

### Start-up

`User Name`
Documents the user name currently being configured, this entry cannot be changed in user-setup, see **$user-name**(5) for help on changing the MicroEmacs user name.

`User Path`
Documents the location of the current user's MicroEmacs setup files, this entry cannot be changed in user-setup, see **$user-path**(5) for help on changing the user path.

`Setup File`
Sets the personal user setup macro file name which is executed at start-up. A user macro file should contain all personal settings such as preferred key bindings etc. See Setting Up A User Profile for more information. The **Edit** check box can be used to enable/disable the automatic loading of the setup file ready for editing when the **Current** button is used.

`Company File`
Sets the company setup macro file name which is executed at start-up. A company macro file should contain all company wide standard settings such as `%company-name`,

`%company-url` and `%company-license`. The value given should not include the `.emf` extension. The value can be a space separated list of company file names, in which case only the first file is executed on start-up, unless the first name is just "–" which no file is executed, see **change-company**(3), **insert-template**(3) and Setting Up a Company Profile for more information.

`Emulation`
: Sets an emulation mode which changes the behaviour on MicroEmacs to emulate another editor/program; this is done by executing a macro file at start-up. An emulation macro file should contain the macro code required to simulate the environment of the other editor. MicroEmacs 2026 is released with four emulation modes, GNU Emacs (see macro file `meemacs.emf`), CUA (Windows style key bindings and interface, see `mecua.emf`), MicroEmacs v3.8 (`meme3_8.emf`) and NEdit v5 which is at best a demonstration of what can be achieved via the emulation support (defined in `menedit.emf`). See Compatibility for more information on emulation.

`Rebind Home Keys`
: When enabled the following key bindings are created to ease frustration, particularly for MS users:

  `home` Bound to beginning-of-line instead of beginning-of-buffer.

  `end` Bound to end-of-line instead of end-of-buffer.

  `C-home` Bound to beginning-of-buffer.

  `C-end` Bound to end-of-buffer.

`MS Shift Region`
: Enables/disables cursor key manipulation with the shift key similar to the conventional Microsoft region selection. When enabled, pressing the shift key in conjunction with the cursor movement keys selects a region which is hilighted. Once the region is selected then the `<DELETE>` or `<BACKSPACE>` key erases the selected region. This also enables a similar behaviour with the Mouse **Drag region** driver, see below.

`C-# to Favorites`
: When enabled, `C-0` opens the favorites menu and `C-1` to `C-9` opens favorite file with the hotkey of given number. `C-c 0` adds the current file to the end to the favorite file list, `C-c 1` to `C-c 9` adds the current file as a favorite file with the given number as a hotkey, replacing the previous file if it exists. See **Favorites** within the File menu section of the main menu for more information on favorites.

### Locale Setup

`Keyboard`
: Configures MicroEmacs to the user's keyboard. Accent character generation keys present on foreign keyboards cannot be automatically supported on Windows platforms. MicroEmacs must be informed of the keyboard being used to correctly interpret the keys. If a required keyboard is not supported please see FAQ38 on how to setup the keyboard, also see Locale Support.

`Language`
: Sets the user language, this sets the word (or letter) characters and if available sets up **spell**(2) with appropriate spelling rules and dictionaries. For more information on adding support for a language see Locale Support.

**NOTES**

Earlier versions MicroEmacs had "(Ext)" languages which use extended language dictionaries, vastly increasing the word list. New versions automatically test for and use these dictionaries if available. More recently these dictionaries have been merged into the main dictionary.

In earlier versions a personal dictionary name could be set in the next field, this option was removed on Oct 2001. Instead a personal dictionary for each language is automatically created for you, any words or auto-corrected words will be added to the current language's personal dictionary. The name of dictionary is "lsp<*lang-id*>.edf" where "<*lang-id*>" is the 4 letter MicroEmacs language name (e.g. "enus" for American), simply rename any existing personal dictionary to this new name.

Auto Save Dictionaries
: Enables auto-saving of any changed dictionaries on exit. If this is disabled the user is prompted to save for each changed dictionary.

Enable Auto-Spell
: Enables Auto Spell Checking in file types which support this feature (usually text based files such as **txt**(9) or **nroff**(9) files etc). Auto spell detects word breaks as you type and checks the spelling of every completed word hilighting any erroneous words in the error color scheme (usually red). When auto-spell is enable, right clicking on an erroneous word will add an extra **Auto Spell** sub-menu to the opened context menu, the sub-menu lists suggested corrections and other useful options; this menu can also be opened by executing the **auto-spell-menu**(3) command (usually bound to "C-f7"). The feature can be manually enabled and disabled by invoking the **auto-spell**(3) command.

**General**

Full Name
: This should be set to the user's name and is used in a variety of places, e.g. by **insert-template**(3) to set the "Created By" field in a template.

Organizer File
: Sets the organizer file base name, defaults to the **Log Name**. When notes and addresses are stored using **organizer**(3) the file "<*Organizer*>.eof" is used.

Global Modes
: Sets the initial state of the global **quiet**(5) global variable.

Search Modes
: Sets the initial state of the global search modes **exact**(2m) and **magic**(2m). This indirectly executes **global-mode**(2) to set the required modes.

Buffer Modes
: Sets the initial state of the global modes **auto**(2m), **backup**(2m), **tab**(2m) and **undo**(2m), any buffers created will inherit the state of these modes. However, as changing these modes directly effects only the global modes, any existing buffers (including ones re-created using the -c command-line option, see **me**(1)) will not be effect by the setting of these modes. For them to take effect, the buffers should be reloaded. These modes can be changed on a per file type basis using the command **buffer-setup**(3), also some file hooks override these global settings, such as the **makefile**(9) hook which overrides the **tab** mode. This indirectly executes **global-mode**(2) to set the required modes.

`Keep Undo`
: If this is enabled the undo history is kept after a save allowing the **undo**(2) command to back-up changes beyond the last save. When clear the undo history is discarded after the buffer is saved. This indirectly sets bit 0x8000 of the **$system**(5) variable.

`Auto-Save Time`
: Sets the length of time in seconds between buffer auto-saves, a setting of `0` or an empty string disables auto-saving. The default setting is 300 seconds or 5 minutes. This indirectly sets the **auto-time**(5) variable and the **autosv**(2m) global mode.

`Hide Backups`
: Enables hiding MicroEmacs generated backup files. On Windows and Dos platforms the Hidden file attribute is used to hide the file, whereas on UNIX the backup file name is prepended with a '`.`'. This indirectly sets bit 0x100000 of the **$system**(5) variable.

`Main Menu - Enable`
: Enables the top main menu bar.

`F-Type Buffer Sel`
: When enabled the main **File** menu's **Buffer** selection sub-menu changes from being a simple complete list of buffers to being a list of **File Types** (as defined by the **File Types** section below). Selecting one of these file types opens a further sub-menu listing all buffers of this type; this option is useful when the current session of MicroEmacs has a large number of buffers.

`Alt Action - Esc Prfx`
: If enabled the Alt key acts as a prefix 1 modifier key when no direct binding exists. By default '`A-n`' is not bound, with this bit set the key is inferred to '`esc n`' which is bound to **forward-paragraph**. Note that global, local and menu hot-key bindings override these. This indirectly sets bit 0x4000 of the **$system**(5) variable.

`Alt Action - Main Menu Hot-keys`
: If enabled the main menu Alt hot-key bindings are enabled. These are dynamic bindings automatically generated from the main menu. Typically the first item in the main menu is "`File`" with a hot key of '**F**', with this enabled '`A-f`' will open this menu item. Note that global and local key bindings override these. This indirectly sets bit 0x2000 of the **$system**(5) variable.

`Abbrev Setup`
: Configures which abbreviation expansion methods are enabled by default when the **expand-abbrev-handle**(3) is executed. **Accent** enables **expand-iso-accents**(3), **Look-back** enables **expand-look-back**(3) and **Dict'n** enables **expand-word**(3).

`Tab To Indent`
: Sets the **tab**(2) behavior in a buffer which has an indentation method. This indirectly sets bits `0x1000` and `0x200000` of the **$system**(5) variable.

`Show Modes`
: Selects which modes are to be displayed on the mode-line whenever a "`%e`" token is used in the **$mode-line**(5) variable. This indirectly sets the **$show-modes**(5) variable.

**Platform - UNIX XTerm Setup**

Only present on UNIX platforms using the X interface, see below for the Console setup.

`Font Name`
: Sets the X font name to be used. This indirectly executes **change-font**(2) with the given font name. e.g.

  `"-*-fixed-medium-r-normal--15-*-*-*-c-90-*"`

  On systems which support FreeType fonts, if the font name does not start with a '-' then a FreeType font is assumed (including the default "" empty string), otherwise the font is assumed to be an X11 core font.

`Display Char Set`
: Selects the display character set being used by the system to render the MicroEmacs window, dependent on the **Font** being used. The setting of this option effects the configuration of MicroEmacs's internal character maps (using command **set-char-mask**(2)) enabling the character sets of foreign languages to be correctly supported. It also changes the definition of variables **$box-chars**(5) and **$window-chars**(5) to their best values for the given font.

`Use Font Styles`
: When enabled the bold, italic, light and underline characteristics of the font will be used depending on their availability and the Color Scheme being used. This indirectly sets bit `0x10` of the **$system**(5) variable.

`Minimize Font Height`
: Some fonts, for some sizes, report a font height that is typically 1 pixel smaller than the sum of the font ascent and descent. When this occurs and this option is selected, MicroEmacs will use the smaller font height value, however this can lead to an unacceptable amount of debris left in the window during normal use. By default this option is not selected, minimising rendering issues. This option has no effect when the two height calculations agree.

`Choose Font`
: If the current system supports FreeType fonts, MicroEmacs will open a font selection dialog, the **Font Name** drop-down lists all FreeType monospaced fonts found on the system that support the current language (see **General** tab). Clicking the **Core Fonts** will close the dialog and run the XTerm command **xfontsel**(1) which opens a dialog allowing the user to select an X11 core font, the current font name is passed back to MicroEmacs if the **quit** button is press, the font name is not altered if **xfontsel** is exited using the title bar close button.

`Draw White Spaces`
: Enables the drawing of visible white spaces, i.e. space, tab and new-line characters. This indirectly sets bit `0x80000` of the **$system**(5) variable.

`Enable Toolbar`
: Enables the Toolbar - configurable, managed windows giving easy access to many features and tools. When enabled the toolbar is opened and displayed at start-up, when disabled the toolbar is not displayed but it can be opened (or closed) by simply executing the **toolbar**(3) command.

`Client Server`
: The client/server enables the file based external macro command driver to be enabled - see Client-Server. This by default is disabled, when enabled it is used by command-line options **-m** and **-o**.

`DOS File Names`
: DOS has a restricted 8.3 file naming system (i.e. "`BBBBBBBB.XXX`"), if this option is enabled the MicroEmacs will adhere to this system for auto-save and backup file names whenever possible. See **$auto-time**(5) for more information on the naming convention used. This indirectly sets bit 0x400 of the **$system**(5) variable.

`# Backups`
: This option only has an effect when **DOS File Names** is disabled. Setting this to a number greater than zero enables multiple backup files to be created, the number determined by this value. If set to zero (or less) then only a single backup file is created. This indirectly sets the **$kept-versions**(5) variable.

`Ignore Files`
: Sets a list extensions of files to be ignored in file completion, e.g. MicroEmacs backup files ( ). This indirectly sets the **$file-ignore**(5) variable.

`System Clipboard`
: Sets the required interaction with the external system clipboard; this sets bits **0x800000**, **0x4000000** and **0x8000000** of the **$system**(5) variable.

`Cursor Blink Rate`
: Sets the cursor blink period in millisecond. The first entry box sets the cursor visible time, a setting of zero disables blinking. The second box sets the hidden time. A visible time of 600 and hidden time of 200 gives a reasonable blink cycle. This indirectly sets the **$cursor-blink**(5) variable.

`Fence Display`
: Sets the preferred method of displaying a matching fence, a fence is one of the following brackets:

```
{...}   (...)   [...]
```

  Jumping to the opening fence only occurs when the closing brace is typed, whereas the drawing of matching fences occurs whenever the cursor is on an open fence or one character past the close fence. When this option is set to "`Never Display`" the **buffer-setup**(3) setting is ignored.

`Scroll Bars`
: Selects the scroll bar support required. When Splitter is enabled, the first character of the scroll bar and mode-line is a split character used for splitting the window into two using the mouse. This indirectly sets the **$scroll-bar**(5) variable.

`Horizontal Scroll`
: Selects the horizontal scrolling method used with the **scroll-left**(2) and **scroll-right**(2) commands. This indirectly sets the **$scroll**(5) variable.

`Vertical Scroll`
: Selects the vertical scrolling method used with the **forward-line**(2) and **backward-line**(2) commands. This indirectly sets the **$scroll**(5) variable.

`Color Scheme`
: Sets the color scheme setup macro file name which is executed at start-up. MicroEmacs by default comes with 4 color schemes. Color schemes can be created and altered using the **scheme-editor**(3) dialog.

### Platform - UNIX Console Setup

Only present on UNIX platforms when using the termcap interface, all the Console platform settings are kept independent of the X interface settings.

`Display Char Set`
: See **Platform UNIX Setup** above.

`UTF-8 encode term I/O`
: Enable this option if your terminal is using UTF-8 encoding. This can usually be determined by disabling the option then executing **insert-symbol**(3), if many of the characters in the second half of the table are drawn as a question mark within a diamond (<?>) the terminal is using UTF-8 encoding and the option should be enabled.

`Extend Char Set`
: When enabled MicroEmacs replaces the display of characters `0x00` to `0x1f` with specific Unicode characters, typically not available in the current charset, which are useful for variables **$box-chars**(5) and **$window-chars**(5) greatly improving the look of **osd**(2) dialogs, the scroll bars etc. This option is only supported when UTF-8 encoding is enabled.

`Termcap Color`
: This option determines whether Termcap based colors should be used. These are typically the standard eight colors and may not be supported on all terminals. If this option is disabled Termcap fonts (such as bold) are used instead to create a primitive hi-lighting. This indirectly sets bit 0x004 of the **$system**(5) variable.

`Use Font Styles`
: See **Platform UNIX Setup** above.

`Draw White Spaces`
: See **Platform UNIX Setup** above.

`Enable Toolbar`
: See **Platform UNIX Setup** above.

`Client Server`
: See **Platform UNIX Setup** above.

`DOS File Names`
: See **Platform UNIX Setup** above.

`# Backups`
: See **Platform UNIX Setup** above.

`Ignore Files`
: See **Platform UNIX Setup** above.

`Cursor Blink Rate`
: See **Platform UNIX Setup** above.

`Fence Display`
: See **Platform UNIX Setup** above.

`Scroll Bars`
: See **Platform UNIX Setup** above.

`Horizontal Scroll`
: See **Platform UNIX Setup** above.

`Vertical Scroll`
: See **Platform UNIX Setup** above.

`Color Scheme`
: See **Platform UNIX Setup** above.

**Platform - Windows Window Setup**

Only present on Microsoft Windows window-based machines, see below for the Console setup.

`Font Name`
: Sets the windows font name and size. This indirectly executes **change-font**(2) with the given font name. MicroEmacs may only use a Fixed Mono Font, either an OEM font as used by the MS-DOS command line, or the more conventional ANSI fonts. The fonts are selected using the **Change Font** button which invokes a dialog to allow the available fonts to be selected. True-Type mono fonts such as `Courier New` or `Lucida Console` are typically used.

`Weight & Size`
: Allows the size and weight of the font to be selected, specified as *weight*, *width* and *height*. The *weight* is typically 4, this corresponds to a regular weighting, 7 is bold. *width* is the width of the font in pixels, this may be 0 when the height is specified as negative. *height* is the height of the font, typically a negative value (where the *width* is 0), which produces a proportionally sized font, values of in the range -11 .. -14 generally produce reasonably sized fonts. The *hight* and *width* may be specified as positive values and allow explicit font dimensions to be specified, generally used to achieve a precise font size requirement.

`Use Font Styles`
: See **Platform UNIX Setup** above.

`Display Char Set`
: See **Platform UNIX Setup** above.

`Choose Font`
: Opens a windows dialog allowing the user to select a font, the selection is used to configure the above font fields.

`Draw White Spaces`
: See **Platform UNIX Setup** above.

`Capture Alt Space`
: Used to enable/disable the capture and interpretation of the '`A-space`' key sequence. If this key sequence is not captured by MicroEmacs it is passed back to Windows which opens the top left window menu, allow keyboard access to Window commands like Maximize.

`Enable Toolbar`
: See **Platform UNIX Setup** above.

`Client Server`
: See **Platform UNIX Setup** above.

`DOS File Names`
: See **Platform UNIX Setup** above. Note that some early version of Windows '95 have problems with extensions. Service release 2 fixed these problems - if you experience problems then return to 8.3 filename mode - note that MicroEmacs will still store longer file names, only the backup naming convention changes.

`# Backups`
: See **Platform UNIX Setup** above.

`Ignore Files`
: See **Platform UNIX Setup** above.

`Cursor Blink Rate`
: See **Platform UNIX Setup** above.

`Fence Display`
: See **Platform UNIX Setup** above.

`Scroll Bars`
: See **Platform UNIX Setup** above.

`Horizontal Scroll`
: See **Platform UNIX Setup** above.

`Vertical Scroll`
: See **Platform UNIX Setup** above.

`Color Scheme`
: See **Platform UNIX Setup** above.

**Platform - Win32 Console Setup**

Only present on Windows platforms when using the console interface, all the Console platform settings are kept independent of the Window interface settings.

`Display Char Set`
: See **Platform UNIX Setup** above.

`Draw White Spaces`
: See **Platform UNIX Setup** above.

`Enable Toolbar`
: See **Platform UNIX Setup** above.

`Client Server`
: See **Platform Win32 Setup** above.

`DOS File Names`
: See **Platform Win32 Setup** above.

`# Backups`
: See **Platform UNIX Setup** above.

`Ignore Files`
: See **Platform UNIX Setup** above.

`Cursor Blink Rate`
: See **Platform UNIX Setup** above.

`Fence Display`
: See **Platform UNIX Setup** above.

`Scroll Bars`
: See **Platform UNIX Setup** above.

`Horizontal Scroll`
: See **Platform UNIX Setup** above.

`Vertical Scroll`
: See **Platform UNIX Setup** above.

`Color Scheme`
: See **Platform UNIX Setup** above.

### Platform - DOS Setup

Only present on DOS machines.

`Graphic Mode #` and `Double Lines`
: Sets the DOS graphics mode number and whether the number of text lines can be doubled. This indirectly executes **change-font**(2) with the given font name.

`Display Char Set`
: See **Platform UNIX Setup** above.

`Double Lines`
: In some graphic modes the number of text lines can be doubled via a dos terminal call, this option is worth trying if more visible text lines are required.

`Draw White Spaces`
: See **Platform UNIX Setup** above.

`Enable Toolbar`
: See **Platform UNIX Setup** above.

`Ignore Files`
: See **Platform UNIX Setup** above.

`Cursor Blink Rate`
: See **Platform UNIX Setup** above.

`Fence Display`
: See **Platform UNIX Setup** above.

`Scroll Bars`
: See **Platform UNIX Setup** above.

`Horizontal Scroll`
: See **Platform UNIX Setup** above.

`Vertical Scroll`
: See **Platform UNIX Setup** above.

`Color Scheme`
: See **Platform UNIX Setup** above.

### Mouse

The mouse device creates keys in a similar way to regular keyboard keys and, like keyboard keysm they must be bound before they are used. MicroEmacs does not have the mouse functionality hard coded into the editor, it provides a macro interface to the mouse for ultimate flexibility and a set of default functionality which can be bound to the mouse in a variety of ways.

All the mouse controlling macros are stored in `mouse.emf` and `mouseosd.emf` although some buffers have local functionality over-rides, such as **file-browser**(3). The user can expand

the range of mouse functionality but how this is achieved is beyond the scope of this documentation.

The **user-setup** dialog allows the user to configure the mouse to use the default functionality, as follows:

`Enable Mouse`
: Enables or disables the mouse, when disabled the mouse can not be used and will not generate any key events. This does not apply to UNIX Termcap systems as the mouse cut and paste operation is performed by the Xterm. This indirectly sets bit 0x010 of the **$mouse**(5) variable.

`Number Buttons`
: Sets the number of buttons on the mouse, may be 1, 2 or 3. MicroEmacs usually obtains the correct number for the system, but sometimes this can be wrong. This entry can be used to correct this problem. For one button mice, the button is considered to be the `left` mouse button, two button mice have an `left` and `right` button. This indirectly sets the **$mouse**(5) variable.

`Swap Buttons`
: If enabled then the `left` and `right` buttons are swapped, i.e. when the left button is pressed it executes the right button bindings. This indirectly sets bit 0x020 of the **$mouse**(5) variable.

`Simulate 3 Buttons`
: If enabled then pressing the `left` and `right` buttons together with generate a middle button press event, this feature is for people with a 2 button mouse who want more. The two buttons must be pressed or release within 10 millisecond of each other.

  The following four fields determine which mouse button binding the user wishes to view and change:

`Button`
: The mouse button, `Left`, `Right` or `Middle` for the normal buttons and `Whell Up` or `Whell Down` for the pilot wheel events.

`Modifier - Shift`
: The action of the mouse can be different for every modifier key setting, if this is enabled then the binding being modified is for the **Button** being pressed with the **Shift** key held down.

`Modifier - Control`
: If enabled then modifying the action when the **Button** is pressed with the **Control** key held down.

`Modifier - Alt`
: If enabled then modifying the action when the **Button** is pressed with the **Alt** key held down.

  The following two fields determine the functionality of the button defined by the previous four fields:

`Bound To`
: The function to be performed. The functions available depend on the type of button being bound, the following is a list of functions available for normal buttons:

  `<not bound>`
  : The Button is not bound.

`Drag region`
: **set-mark**(2) is called at the pick location, until the button is dropped, the area of text between this point and the current mouse position is hi-lighted. When the mouse button is dropped, if the drop position is the same as the pick then the double click is tested for, if a double click is entered then the **Select Word** function is executed, otherwise the cursor is simply moved to the drop position. If the pick and drop position are different then the enclosed text is copied to the kill buffer using **copy-region**(2). Note this behaviour is altered by the setting of **MS Shift Region** on the **Start-Up** page.

`Copy Rectangle`
: One corner of a rectangle is defined on the pick and the diagonally opposite corner is defined when the button is dropped, the text enclosed within the rectangle is copied to the kill buffer using **copy-rectangle**(3).

`Kill Rectangle`
: The same as `Copy Rectangle` except the text is killed using **kill-rectangle**(2).

`Select Word`
: Also executed from a double click bound to **Drag Region**, **Select Word** copies the word under the mouse into the kill buffer using **copy-region**(2), unless a double click is entered in which case the whole line is copied.

`Default Pan`
: While the mouse button is pressed the current buffer pans with any mouse movement.

`MS Pan`
: MicroSoft style Pan; while the mouse button is pressed the current buffer pans vertically according to the mouse position relative to the point where the button was pressed.

`Find Buffer`
: Opens a buffer selection dialog, the buffer selected is displayed in the window.

`Find Tag`
: Executes **find-tag**(2) with the word currently under the mouse.

`Find ME Help`
: Executes **help-item**(2) with the word currently under the mouse.

`Undo`
: Simply executes **undo**(2) without moving the cursor to the position of the mouse. Subsequent calls to this binding will undo multiple edits.

`No Move Yank`
: Simply executes **yank**(2) without moving the cursor to the position of the mouse.

`Replace Yank`
: Simplar to "`No move yank`" except when the is a current region (typically defined by "`Drag region`" above), in which case the region is first deleted.

`Move to Yank`
: Moves the cursor to the current position of the mouse and executes **yank**(2).

`Reyank`
: Executes **reyank**(2) without moving the cursor. Note, to enable this functionality some sanity checks have had to be removed, as a result it should not be misused as seeming bizarre things can occur.

`Rectangle Yank`
: Moves the cursor to the current position of the mouse and executes **yank-rectangle**(2).

`Collapse Current`
: Toggles the collapse status of the current block, only applicable in buffers supporting **collapse-current**(3), such as c and emf files.

`Collapse All`
: Toggles the collapse status of the whole buffer, opening or closing all found blocks. Only applicable in buffers supporting **collapse-all**(3), such as c and emf files.

`Main Menu`
: Simply opens the main menu from any where on the screen.

`Context Menu`
: Opens a context sensitive menu dependent on the position of the mouse, i.e. opens the main menu if over it, opens a different menu when executed on the mode-line etc.

  The following is a list of functions available for pilot wheel events:

`Not bound`
: The Button is not bound.

`Scroll Up 1 Line` ....
: Scrolls the current buffer by the specified amount.

`Handle Scroll Bars`
: When enabled, if the button is pressed with the mouse on the main menu, a scroll bar or mode-line the standard action is performed, such as opening the main menu or scrolling up or down the window etc. The **bound To** command is only called if the mouse is in a main window. If disabled, the **Bound To** command is always called.

`Defaults`
: Rests the mouse configuration to the default settings.

### Miscellaneous

This tab gives easy access to setup dialogs of other features:

`Alias Path`
: Dialog to configure alias paths, for more information see **alias-path-setup**(3).

`ClearCase`
: Dialog to configure the IBM ClearCase interface, for more information see **cc-setup**(3).

`CVS` Dialog to configure the Concurrent Versions System (CVS) interface, for more information see **cvs-setup**(3).

`File Tools`
: Dialog to configure file tools which can be used in the file browser, for more information see file-tool-setup 3 .

`File Types`
: Dialog to configure file type list which is used in the main menu's **Open** and **Quick Open** items, for more information see file-type-setup 3 .

`Find` Dialog to configure the **find**(3), **grep**(3) and **replace-string-in-files**(3) commands, see **find-setup**(3).

`Session`
: Dialog to configure the current session, for more information see **session-setup**(3).

`Shell Tools`
: Dialog to configure the shell tools, these are an alternative to **File Tools** as they are typically used to operate on the current buffer's file. For more information see **shell-tool-setup**(3).

`Subversion`
: Dialog to configure the Subversion (SVN) interface, see **svn-setup**(3).

## NOTES

**user-setup** is a macro using **osd**(2), defined in `userstp.emf`.

## SEE ALSO

User Profiles, Company Profiles, Installation, Top Main Menu, **buffer-setup**(3), **change-company**(3), **change-scheme**(3), **scheme-editor**(3), **alias-path-setup**(3), **cc-setup**(3), **cvs-setup**(3), **find-setup**(3), **shell-tool-setup**(3), **$user-name**(5), **$user-path**(5).

Ref: user-setup(3) File: m3osd000.3 Date: 2025/10/20

## NAME

2unix - Convert text files to UNIX format files

## SYNOPSIS

**me** "@2dos" <*files*>
**me** "@2mac" <*files*>
**me** "@2unix" <*files*>
**me** "@2win" <*files*>

## DESCRIPTION

The start-up file `2unix.emf` may be invoked from the command line as a filter to convert all text files, which may be in MS-DOS, Windows, Macintosh format into UNIX text format.

Each file specified on the command line is interrogated and the line ending modified to UNIX.

`2dos`, `2win` and `2mac` similarly operate to provide conversion from any of these formats to MS-DOS, Macintosh or Windows file formats.

## EXAMPLE

When using ME as a command line filter it is advisable to control the backup command line options. Typically the backups and auto-save are disabled so that no backup files are generated i.e.

```
me -a -B "@2dos" files...
```

Performs a conversion from UNIX to MS-DOS and disables the generation of auto-save files and backups.

NE may be used to run the filters, by default NE runs without backup and auto-save mode enabled and the flags should NOT be used.

```
ne "@2dos" files...
```

## NOTES

`MS-DOS` files are defined to use a line ending of `\r\n`, the file is terminated with `ctrl-z`.

Macintosh files are defined to use a line ending of `\r`, there is no special end of file character.

Microsoft Windows files are defined to use a line ending of `\r\n`, there is no special end of file character.

UNIX files are defined to use a line ending of `\n`, there is no special end of file character.

The unix2dos.emf, dos2unix.emf and unix2win.emf were renamed to 2dos.emf, 2unix.emf and 2win.emf when 2mac.emf was added in October 2006 as the 'from' format is not important and number of combinations became unmanageable.

## SEE ALSO

**me**(1), **start-up**(3), **filetype**(3f), **auto**(2m), **cr**(2m), **lf**(2m), **ctrlz**(2m).

Ref: 2unix(3f) File: m3fcl002.3 Date: 2022/06/12

## NAME

benchmrk - Benchmark MicroEmacs macro processor speed

## SYNOPSIS

**me** "@benchmrk"

## DESCRIPTION

The start-up file `benchmrk.emf` may be invoked from the command line to time the macro processor variable manipulation times.

This macro suite was developed to optimize the macro processor performance, and allows comparable analysis of various macro variable manipulations. The macro is not important in it's own right and is not likely to be useful. Running it will provide an in-site into the speed of the macro language and should indicate to the user what are good and bad things to be doing.

As an aside, as MicroEmacs interprets the macro code the it is important that the processing operates at a reasonable speed. Most extensions offering additional functionality are likely to be added to MicroEmacs by way of a macro implementation - this allows speedy development of new features. Obviously core changes do occur when we find that we cannot support certain new requirements, or when the macro code becomes too convoluted. In these cases, new commands are added to help us solve the problem. However, recent evolution of the code has indicated that the core set is now reasonably complete.

## SEE ALSO

**start-up**(3).

Ref: benchmrk(3f) File: m3fcl003.3 Date: 2022/06/12

## NAME

ctags - Generate a C tags file

## SYNOPSIS

**me** "@ctags" [-*v%tag-option*=<*flags*>] [*files*]

## DESCRIPTION

The start-up file `ctags.emf` may be invoked from the command line to generate a **tags** file for C and C++ source and header files.

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. This is a good alternative on Microsoft platforms where a utility such as **ctags**(1) is not typically available. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all C and C++ source and header files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value <*flags*> can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag '`m`' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

`v` Add global variables to the tag file. (i.e. variables marked with *extern*).

`e` Add enumerated variables to the tag file (i.e. *enum* members).

`s` Add structure, type definitions and classes to the tag file (i.e. *stuct*, *typedef* and *class*).

The generated tags file includes `#define` and C++ class names.

## NOTES

This function is invoked from menu

**Tools -> C Tools -> Create Tags File**

when the user requests a tags file to be generated.

The user setup file "`myctags.emf`" is executed by ctags during start-up, this file can be used to over-ride any of the ctags configuration variables (see below).

The following variables are set within "`ctags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, default value is "`:*.[cC]:*.cpp:*.cc:*.h:*.hpp:`".

**%tag-ignoredir**

A list of directories to be ignored when recursive option is used, default value is ":SCCS/:CVS/:.git/:.svn/:".

These variables can be changed using the -v command-line option or via the "myctags.emf" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **c**(9).

Ref: ctags(3f) File: m3fcl004.3 Date: 2025/07/18

## NAME

ehftools - Generate a MicroEmacs help file

## SYNOPSIS

**me** "@ehftools" *.htm

## DESCRIPTION

The start-up file `ehftools.emf` may be invoked from the command line to generate a MicroEmacs help file from a set of HTML files (with the extension `.htm`).
The MicroEmacs documentation suite of tools has not been officially released as part of the distribution. For reference, the sequence of operations that are performed from the command line or shell script are:

```
make meehf.hts
hts2html -l .htm meehf.hts
mv me.htm me/1.htm
cd me
me "@ehftools" *.htm
```

## NOTES

The *nroff* to HTML generator leaves the special markers `<!-- XI: %s -->` in the generated HTML code which contain the hypertext link information.

## SEE ALSO

**start-up**(3), **ehf**(8).

Ref: ehftools(3f) File: m3fcl005.3 Date: 2022/06/12

## NAME

emftags - Generate a MicroEmacs macro tags file

## SYNOPSIS

**me** "@emftags" [*-v%tag-option=<flags>*] [*files*]

## DESCRIPTION

The start-up file `emftags.emf` may be invoked from the command line to generate a **tags** file for MicroEmacs macro files, **emf**(8).

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all MicroEmacs macro files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value *<flags>* can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag '`m`' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

## NOTES

This function is invoked from menu

**Tools -> Emf Tools -> Create Tags File**

when the user requests a tags file to be generated.

The user setup file "`myemftags.emf`" is executed by emftags during start-up, this file can be used to over-ride any of the emftags configuration variables (see below).

The following variables are set within "`emftags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, default value is "`:*.emf:`".

**%tag-ignoredir**
A list of directories to be ignored when recursive option is used, default value is "`:SCCS/:CVS/:.git/:.svn/:`".

These variables can be changed using the -v command-line option or via the "`myemftags.emf`" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **emf**(8).

Ref: emftags(3f) File: m3fcl006.3 Date: 2025/07/18

## NAME

filetype - Determine file type UNIX, DOS or Windows

## SYNOPSIS

**me** "@filetype" <*files*>

## DESCRIPTION

The start-up file `filetype.emf` may be invoked from the command line as a filter to determine the file type (MS-DOS, Windows or UNIX). Each file specified on the command line is interrogated and the line ending is reported.

When the command completes then the `*scratch*` buffer contains a list of the files and their file type.

## SEE ALSO

**start-up**(3), **dos2unix**(3f), **auto**(2m), **cr**(2m), **lf**(2m), **ctrlz**(2m).

Ref: filetype(3f) File: m3fcl012.3 Date: 2022/06/12

## NAME

gdiff - Command line graphical file difference

## SYNOPSIS

**me** "@gdiff" "*version1*" "*version2*"

## DESCRIPTION

MicroEmacs may be executed from the command line to invoke the *Graphical Difference* **gdiff**(3) macro, showing the difference(s) between two files.

The editor is invoked in **gdiff** mode and shows the difference between the two files on the command line. The two files are presented side by side and the lines are matched together, the windows are effectively locked together, scrolling one window causes the other window to scroll.

The changes are shown as color coded regions of highlighted text. Each block (or line) has to be accepted or rejected by the user, when all blocks have been accepted then the buffer may be written out merging the changes from both files depending on the accept/reject status. Blocks which have been accepted are typically shown in green, the rejected lines are shown in red.

### Key Mapping

The keyboard mapping in **gdiff** mode is defined as follows:

**esc-h** Help information.

**C-up/p** Advance to the previous difference.

**C-down/n**
Advance to the next difference.

**esc-up**/P
Advance to the previous unassigned difference.

**esc-down**/N
Advance to the next unassigned difference.

**g** Globally select all of the differences from the current window as assigned.

**G** Globally un-select all of the assigned differences.

**r/space/return/left-mouse**
Selects the current difference, only operates within a highlighted difference region.

**R** Un-selects the current difference, only operates within a highlighted difference region.

**l** Selects the current line in a highlighted region for assignment.

**L** Unselects the current line in a highlighted region for assignment.

**save-buffer**
Saves the current buffer, all of the assignment changes are merged and written out to the existing buffer.

**write-buffer**
: Saves the current buffer after merging all of the and assignment changes and prompts the user for the name of a new buffer.

**left/right**
: Scrolls the buffer left or right.

**kill-buffer/C-x C-c**
: Quits **gdiff**, the user is prompted to save changes.

## NOTES

The macro is defined in file `gdiff.emf`.

## SEE ALSO

**gdiff**(3), **start-up**(3), **xdiff**(3).

Ref: gdiff(3f) File: m3fcl001.3 Date: 2022/06/12

## NAME

info - Display Info pages.
pydoc - Display Python module documentation.
rdoc - Display R help pages.

## SYNOPSIS

**me** "@info" <*topic*>
**me** "@pydoc" <*modulname*>
**me** "@rdoc" <*helppage*>

## DESCRIPTION

The start-up files `info.emf`, `pydoc.emf` or `rdoc.emf` may be invoked from the command line as interactive documentation browser for info pages, Python module help pages or R help pages to be displayed in MicroEmacs as a more sophisticated documention tool than the standard Unix tools.

## EXAMPLE

Here are some examples how to use these command line arguments to display specific help pages within ME as a command line filter, i.e.

```
me @info coreutils
me @pydoc argparse
me @rdoc mean
me @rdoc cluster::pam
```

## NOTES

The start-up code is stored in the files info.emf, pydoc.emf and rdoc.emf were added in December 2025 and July 2026.

## SEE ALSO

**me**(1), **start-up**(3), **info**(3), **mdview**(3f), **py-doc**(3), **r-doc**(3).

Ref: info(3f) File: m3fcl018.3 Date: 2026/08/26

## NAME

javatags - Generate a C tags file from Java sources.

## SYNOPSIS

**me** "@javatags" [*-v%tag-option=<flags>*] [*files*]

## DESCRIPTION

The start-up file `javatags.emf` may be invoked from the command line to generate a **tags** file for java source files.

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. This is a good alternative on Microsoft platforms where a utility such as **ctags**(1) is not typically available to process Java files. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all Java source files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value *<flags>* can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag '`m`' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

`v` Add global variables to the tag file. (i.e. variables marked with *extern*).

`s` Add classes definitions to the tag file (i.e. *class*).

The generated tags file includes `#define` and C++ class names.

## NOTES

This function is invoked from menu

**Tools -> C Tools -> Create Tags File**

when the user requests a tags file to be generated.

The user setup file "`myjavatags.emf`" is executed by javatags during start-up, this file can be used to over-ride any of the javatags configuration variables (see below).

The following variables are set within "`javatags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, default value is "`:*.java:*.jav:`".

**%tag-ignoredir**

A list of directories to be ignored when recursive option is used, default value is ":SCCS/:CVS/:.git/:.svn/:".

These variables can be changed using the -v command-line option or via the "myjavatags.emf" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **java**(9).

Ref: javatags(3f) File: m3fcl011.3 Date: 2025/07/18

## NAME

jstags - Generate a Javascript tags file

## SYNOPSIS

**me** "@jstags" [*-v%tag-option=<flags>*] [*files*]

## DESCRIPTION

The start-up file `jstags.emf` may be invoked from the command line to generate a **tags** file for Javascript files, **js**(9).

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all Javascript files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value *<flags>* can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag 'm' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

## NOTES

This function is invoked from menu

**Tools -> Javascript Tools -> Generate Tags File**

when the user requests a tags file to be generated.

The user setup file "`myjstags.emf`" is executed by jstags during start-up, this file can be used to over-ride any of the jstags configuration variables (see below).

The following variables are set within "`jstags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, default value is "`:*.js:`".

**%tag-ignoredir**
A list of directories to be ignored when recursive option is used, default value is "`:SCCS/:CVS/:.git/:.svn/:`".

These variables can be changed using the -v command-line option or via the "`myjstags.emf`" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **js**(9).

Ref: jstags(3f) File: m3fcl016.3 Date: 2025/08/24

## NAME

luatags - Generate a Lua tags file

## SYNOPSIS

**me** "@luatags" [*-v%tag-option=<flags>*] [*files*]

## DESCRIPTION

The start-up file `luatags.emf` may be invoked from the command line to generate a **tags** file for Lua files, **lua**(9).

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all Lua files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value *<flags>* can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag 'm' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

## NOTES

This function is invoked from menu

**Tools -> Lua Tools -> Generate Tags File**

when the user requests a tags file to be generated.

The user setup file "`myluatags.emf`" is executed by luatags during start-up, this file can be used to over-ride any of the luatags configuration variables (see below).

The following variables are set within "`luatags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, default value is "`:*.lua:`".

**%tag-ignoredir**
A list of directories to be ignored when recursive option is used, default value is "`:SCCS/:CVS/:.git/:.svn/:`".

These variables can be changed using the -v command-line option or via the "`myluatags.emf`" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **lua**(9).

Ref: luatags(3f) File: m3fcl017.3 Date: 2025/08/24

## NAME

mdview - Browse Markdown pages.

## SYNOPSIS

**me** "@mdview" <*markdownfile*>

## DESCRIPTION

The start-up file `mdview.emf` may be invoked from the command line as interactive Markdown file browser for Wiki like organized Markdown files so that these files are displayed in MicroEmacs in a sophisticated way with links to other Markdown files displayed as clickable links and using a navigation header with links to home, next, previous, help etc.

## EXAMPLE

Here is how to use this command line filter to display specific markdown files ME as a command line filter, i.e.

```
me @mdview wiki/home.md
```

## NOTES

The start-up code is stored in the file mdview.emf, the filehook code in the file hkmdview.emf. The macro was added in August 2026.

## SEE ALSO

**me**(1), **start-up**(3), **info**(3f), **pydoc**(3f), **rdoc**(3f).

Ref: mdview(3f) File: m3fcl019.3 Date: 2026/08/26

## NAME

ntags - Generate a nroff tags file

## SYNOPSIS

**me** "@ntags" <*files*>

## DESCRIPTION

The start-up file `ntags.emf` may be invoked from the command line to generate a **tags** file for nroff files.

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all nroff files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value <*flags*> can contain any number of the following flags:

| | |
|---|---|
| `a` | Append new tags to the existing tag file, note that if also using flag 'm' multiple 'tags' to the same item may be created. |
| `m` | Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support. |
| `r` | Enables recursive mode, any sub-directory found within any given directories will also be processed. |

## NOTES

This function is invoked from menu

**Tools -> Nroff Tools -> Create Tags File**

when the user requests a tags file to be generated.
The tags are generated from the nroff macro:

```
.XI <name> ......
```

which indicates an index entry, where <*name*> is the tag name. <*name*> may be delimited by double quotes if any whitespace is present in the string.

This is the macro definition used in the MicroEmacs documentation system. The `ntags.emf` file should be edited and shadowed in the user directory if some other search criteria is used for nroff files. This macro file should provide a good starting point for any other search.

The user setup file "`myntags.emf`" is executed by ntags during start-up, this file can be used to over-ride any of the ntags configuration variables (see below).

The following variables are set within "`ntags.emf`" and are used to control the process:

**%tag-option**
: Tags options flag, default value is "". See above for more information.

**%tag-filemask**

A list of source file masks to be processed when a directory is given, default value is ”`:*.nrs:*.[1-9]:*.n:`”.

**%tag-ignoredir**

A list of directories to be ignored when recursive option is used, default value is ”`:SCCS/:CVS/:.git/:.svn/:`”.

These variables can be changed using the -v command-line option or via the ”`myntags.emf`” file

## SEE ALSO

**find-tag**(2), **start-up**(3), **nroff**(9).

Ref: ntags(3f) File: m3fcl007.3 Date: 2025/07/18

## NAME

perltags - Generate a Perl tags file

## SYNOPSIS

**me** "@perltags" [*-v%tag-option=<flags>*] [*files*]

## DESCRIPTION

The start-up file `perltags.emf` may be invoked from the command line to generate a **tags** file for Perl files, **pl**(9).

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is ". /", i.e. process the current directory. If a directory name is given (such as the default ". /") all Perl files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value *<flags>* can contain any number of the following flags:

| | |
|---|---|
| `a` | Append new tags to the existing tag file, note that if also using flag 'm' multiple 'tags' to the same item may be created. |
| `m` | Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support. |
| `r` | Enables recursive mode, any sub-directory found within any given directories will also be processed. |

## NOTES

This function is invoked from menu

**Tools -> Perl Tools -> Generate Tags File**

when the user requests a tags file to be generated.

The user setup file "`myperltags.emf`" is executed by perltags during start-up, this file can be used to over-ride any of the perltags configuration variables (see below).

The following variables are set within "`perltags.emf`" and are used to control the process:

**%tag-option**
: Tags options flag, default value is "". See above for more information.

**%tag-filemask**
: A list of source file masks to be processed when a directory is given, default value is "`:*.pl:*.pm:`".

**%tag-ignoredir**
: A list of directories to be ignored when recursive option is used, default value is "`:SCCS/:CVS/:.git/:.svn/:`".

These variables can be changed using the -v command-line option or via the "`myperltags.emf`" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **pl**(9).

Ref: perltags(3f) File: m3fcl014.3 Date: 2025/07/23

## NAME

phptags - Generate a PHP tags file

## SYNOPSIS

**me** "@phptags" [*-v%tag-option=<flags>*] [*files*]

## DESCRIPTION

The start-up file `phptags.emf` may be invoked from the command line to generate a **tags** file for PHP files, **php**(9).

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all PHP files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value *<flags>* can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag '`m`' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

## NOTES

This function is invoked from menu

**Tools -> PHP Tools -> Generate Tags File**

when the user requests a tags file to be generated.

The user setup file "`myphptags.emf`" is executed by phptags during start-up, this file can be used to over-ride any of the phptags configuration variables (see below).

The following variables are set within "`phptags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, the default value is:

"`:*.php:*.php3:*.php4:*.php5:*.phtml:*.inc:`".

**%tag-ignoredir**
A list of directories to be ignored when recursive option is used, default value is:

"`:SCCS/:CVS/:.git/:.svn/:`".

These variables can be changed using the -v command-line option or via the "`myphptags.emf`" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **php**(9).

Ref: phptags(3f) File: m3fcl015.3 Date: 2026/02/17

## NAME

printall - Formatted print job

## SYNOPSIS

**me** "@printall" <*files*>

## DESCRIPTION

The start-up file `printall.emf` may be invoked from the command line to generate a print job for each file specified on the command line.

Given a list of <*files*>, the files are loaded into the editor, and then printed through MicroEmacs printing formatter. This is an alternative to **cgrind**(1) or some other syntax smart *pretty print* filter.

The operation of this macro assumes that the printer is functioning correctly.

## BUGS

As a guess, I would probably bet that this does not work very well on Windows as a dialog is invoked for the print.

## SEE ALSO

**start-up**(3).

Ref: printall(3f) File: m3fcl008.3 Date: 2026/06/19

## NAME

tcltags - Generate a Tcl/Tk tags file

## SYNOPSIS

**me** "@tcltags" <*files*>

## DESCRIPTION

The start-up file `tcltags.emf` may be invoked from the command line to generate a **tags** file for Tcl/Tk files.

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all Tcl/Tk files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value <*flags*> can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag '`m`' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

## NOTES

This function is invoked from menu

**Tools -> Tcl Tools -> Create Tags File**

when the user requests a tags file to be generated.

The following variables are set within "`tcltags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, default value is "`:*.tcl:*.tk:`".

**%tag-ignoredir**
A list of directories to be ignored when recursive option is used, default value is "`:SCCS/:CVS/:.git/:.svn/:`".

These variables can be changed using the -v command-line option or via the "`mytcltags.emf`" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **tcl**(9).

Ref: tcltags(3f) File: m3fcl009.3 Date: 2025/07/18

## NAME

textags - Generate a LaTeX/BibTeX tags file

## SYNOPSIS

**me** "@textags" <*files*>

## DESCRIPTION

The start-up file `textags.emf` may be invoked from the command line to generate a **tags** file for LaTeX and BibTeX files.

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all LaTeX files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value <*flags*> can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag '`m`' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

## NOTES

This function is invoked from menu

**Tools -> LaTeX Tools -> Create Tags File**

when the user requests a tags file to be generated.

The following variables are set within "`textags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, default value is "`:*.tex:*.bib:`".

**%tag-ignoredir**
A list of directories to be ignored when recursive option is used, default value is "`:SCCS/:CVS/:.git/:.svn/:`".

These variables can be changed using the -v command-line option or via the "`mytextags.emf`" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **tex**(9).

Ref: textags(3f) File: m3fcl010.3 Date: 2025/07/18

## NAME

vbtags - Generate a Visual Basic tags file

## SYNOPSIS

**me** "@vbtags" [-*v%tag-option*=<*flags*>] [*files*]

## DESCRIPTION

The start-up file `vbtags.emf` may be invoked from the command line to generate a **tags** file for Visual Basic files, **vb**(9).

Given a list of *files* a tags file `tags` is generated in the current directory, which may be used by the **find-tag**(2) command. If no *files* are specified the default file list is "`./`", i.e. process the current directory. If a directory name is given (such as the default "`./`") all Visual Basic files within the directory will be processed.

The value of variable **%tag-option** is used to control the tag generation process, its value <*flags*> can contain any number of the following flags:

`a` Append new tags to the existing tag file, note that if also using flag 'm' multiple 'tags' to the same item may be created.

`m` Enable multiple tags. This enables the existence of 2 tags with the same tag name, but typically with different locations. See help on **find-tag**(2) for more information on multiple tag support.

`p` Add prototype declarations (i.e. lines marked with *Declare* and a function or subroutine name).

`r` Enables recursive mode, any sub-directory found within any given directories will also be processed.

`v` Add global variables to the tag file. (i.e. variables marked with *Public* or *Private*).

## NOTES

This function is invoked from menu

**Tools -> Visual Basic Tools -> Generate Tags File**

when the user requests a tags file to be generated.

The user setup file "`myvbtags.emf`" is executed by vbtags during start-up, this file can be used to over-ride any of the vbtags configuration variables (see below).

The following variables are set within "`vbtags.emf`" and are used to control the process:

**%tag-option**
Tags options flag, default value is "". See above for more information.

**%tag-filemask**
A list of source file masks to be processed when a directory is given, default value is "`:*.bas:*.cls:*.asp:`".

**%tag-ignoredir**

A list of directories to be ignored when recursive option is used, default value is ":SCCS/:CVS/:.git/:.svn/:".

These variables can be changed using the -v command-line option or via the "myvbtags.emf" file

## SEE ALSO

**find-tag**(2), **start-up**(3), **vb**(9).

Ref: vbtags(3f) File: m3fcl013.3 Date: 2025/07/18

## NAME

!bell - Sound audio alarm

## SYNOPSIS

**!bell** [*n*]

## DESCRIPTION

**!bell** provides a warning (audible or visual) to alert the user of a problem. **!bell** honors the **$quiet**(5) setting, as such if **quiet** mode is disabled an audible warning is given, otherwise a visual warning is given to the user (usually the message "*[BELL]*" in the bottom left hand corner of the message line).

The optional numerical argument *n* over-rides the current setting of **quiet**, a value of `0` specifies a quiet bell, `2` an audible one, when omitted the default is `1` honoring the setting of **quiet** mode.

**!bell** is generally used in conjunction with !abort, the !bell function warning the user and the !abort function to quit the macro.

## EXAMPLE

The following macro example checks for incoming mail and is taken from `mail.emf`. If any mail has arrived an audible warning is assured by toggling the **quiet** mode.

```
;
; Mail checker
define-macro mail-check
    !if &seq &set %vm-mail-src &reg &spr "/history/%s/mail-src"  $platform ""
        ml-write "[Incoming mail file not setup! Execute mail-setup to configure]"
        !abort
    !endif
    600000 create-callback mail-check
    ml-write &spr "Checking for mail in %s..." %vm-mail-src
    set-variable #l0 &cond &gre &stat "s" %vm-mail-src 0 "M" "-"
    !if &not &seq &mid $mode-line 2 1 #l0
        set-variable #l1 &rig $mode-line &cond &seq &mid $mode-line 2 1 "%" 4 3
        set-variable $mode-line &cat &cat &lef $mode-line 2 #l0 #l1
        screen-update
        !if &seq #l0 "M"
            ; use an argument to !bell so it toggles it back to its original state
            !bell 0
            !bell 2
        !endif
    !endif
    0 ml-write
!emacro
```

## SEE ALSO

**!abort**(4), **abort-command**(2), **$quiet**(5).

Ref: !bell(4) File: m4fil013.4 Date: 2026/06/19

## NAME

!emacro - Terminate a macro definition
!ehelp - Terminate a help definition

## SYNOPSIS

define-macro *macro-name*
*... macro body ...*
**!emacro**
define-help *item-name*
*... help body ...*
**!ehelp**

## DESCRIPTION

**!emacro** terminates the storage of an open macro, (opened with **define-macro**(2)). Only the lines between **define-macro** and the **!emacro** directive comprise the new macro *macro-name*.

Similarly **!ehelp** terminates the storage of an open help definition, (opened with **define-help**(2)). Only the lines between **define-help** and the **!ehelp** directive comprise the new help text for item *item-name*.

**!emacro** and **!ehelp** may not be used in any other context.

## EXAMPLE

For example if a file is being executed contains the text:

```
;
; Read in a file in view mode, and make the window red
;
define-macro view-a-file
    find-file @ml"File to view: "
    1 buffer-mode "view"
    set-variable $buffer-bcol %red
!emacro

define-help view-a-file
    This is the help text for the macro view-a-file.
!ehelp

ml-write "[view-a-file macro has been loaded]"
```

then only the lines between the **define-macro** command and the **!emacro** directive are stored in macro *view-a-file* and the lines between the **define-help** command and the **!ehelp** directive are stored as help for *view-a-file*. The ml-write line is executed when the file is loaded, and the message will appear on the message line, this does not however form part of the macro or help.

## SEE ALSO

Operating Modes, **define-macro**(2), **define-help**(2).

Writing MicroEmacs Macros
Macro Glossary
Macro Debugging - A Crash Course
MicroEmacs Macro Language at a Glance

Ref: !emacro(4) File: m4fil006.4 Date: 2026/02/21

## NAME

!force - Ignore command or macro status

## SYNOPSIS

**!force** [*n*] *command*

## DESCRIPTION

**!force** ignores the return status of a *command* while executing a macro. When MicroEmacs executes a macro, if any command fails, the macro is terminated at that point. If a line is preceded by a **!force** directive, execution continues whether the command succeeds or not. **$status**(5) may be used following **!force** to determine if the command failed or not.

A double **!force** can be used to catch a user termination (via the **abort-command**(2) bound to `C-g`). A macro command aborted by the user will be terminated even with a single **!force** directive, but not with two. See the example below.

When specifying a numerical argument with a *command*, it is placed after the **!force** directive and before the *command* i.e.

```
!force 1 forward-char
```

## EXAMPLE

The following example shows how **!force** is used in conjunction with **$status**.

```
;       Merge the top two windows

push-position           ;remember where we are
1 next-window           ;go to the top window
delete-window           ;merge it with the second window
!force pop-position     ;This will continue regardless
!if $status
    ml-write "Call PASSED"
!else
    ml-write "Call FAILED"
!endif
```

The following example creates an infinite loop that can only be broken out of by a user abort. The calling macro catches this by using a double **!force** and continues. This concept is used by commands which take a considerable amount of time yet cannot be simply aborted by the user such as the spell-checker's best guess list generator.

```
define-macro infinite-loop
    set-variable #l0 1
    !while 1
        ml-write &cat "In loop, C-g to exit: " &pinc #l0 1
    !done
!emacro

define-macro catch-abort
    !force !force infinite-loop
    ml-write "You will see this"
!emacro
```

## SEE ALSO

**$status**(5).

Ref: !force(4) File: m4fil007.4 Date: 2026/02/17

## NAME

!goto - Unconditional labeled jump
!tgoto - Conditional labeled jump

## SYNOPSIS

**!goto** *label*
...
*****_label_
**!tgoto** *condition label*
...
*****_label_

## DESCRIPTION

Flow can be controlled within a MicroEmacs macro using the **!goto** directive. It takes as an argument a *label*. A *label* consists of a line starting with an asterisk (*) and then an alphanumeric label. Only labels in the currently executing macro can be jumped to, trying to jump to a non-existing label terminates execution of a macro. *labels* may be located at any position within the macro (forwards or backwards from the **!goto**).

A conditional jump may be implemented with a **!tgoto**, this takes an additional argument *condition*, which may be a literal numeric value, a variable or an evaluated expression (see Variable Functions). If the *condition* evaluates to TRUE (or non-zero) then the branch is taken and control continues from the *label*.

**!tgoto** is an ideal replacement for **!while**(4) and **!repeat**(4) where nested loops are required.

## EXAMPLE

For example, create a block of DATA statements for a BASIC program:

```
        insert-string "1000 DATA "
        set-variable %linenum 1000
*nxtin
        screen-update           ;make sure we see the changes
        set-variable %data @ml"Next number: "
        !if &equal %data 0
            !goto finish
        !endif
        !if &greater $curcol 60
            2 backward-delete-char
            newline
            set-variable %linenum &add %linenum 10
            insert-string &cat %linenum " DATA "
        !endif
        insert-string &cat %data ", "
        !goto nxtin
*finish
        2 backward-delete-char
        newline
```

Not that any of us are writing basic programs these days !!

## NOTES

**!goto** and **!tgoto** are expensive operations because a symbolic name lookup is performed in the macro file. For time critical macros then the **!jump**(4) and **!tjump**(4) directives should be used

as these do not perform a symbolic name search. The *jump* equivalents are source sensitive since a line displacement rather than a *label* is used - this makes them a little dangerous to use.

## SEE ALSO

Variable Functions, **!if**(4), **!jump**(4), **!repeat**(4), **!return**(4), **!tjump**(4), **!while**(4).

Ref: !goto(4) File: m4fil009.4 Date: 2026/02/21

## NAME

!if, !elif, !else, !endif - Conditional statements

## SYNOPSIS

**!if** *condition*
*... condition body ...*
[**!elif** *condition*
*... condition body ...* ]
[**!else**
*... condition body ...* ]
**!endif**

**!iif** *condition body*

## DESCRIPTION

The conditional directives allow statements to be executed only if a condition specified in the directive is met, as follows:

- Every line following the **!if** directive, until the first **!elif**, **!else** or **!endif** directive, is only executed if the expression following the **!if** directive evaluates to a `TRUE` value (non-zero).
- If the **!if** evaluates to `FALSE` and a **!elif** directive is next then the expression following the **!if** is evaluated and following statements are executed if `TRUE`.
- If no **!if** or **!elif** is found to be `TRUE` and a **!else** is found then the statements following it are executed.

The *condition* may be any logical condition as evaluated by the variable functions (e.g. **&equal**(4)) returning `TRUE` or `FALSE`. An integer value, non-zero evaluates `TRUE`, zero evaluates to `FALSE`. A non-numerical argument, such as a string is always `FALSE`.

The *conditional body* may be any **MicroEmacs** function, macro or directive with the exception of **define-macro** and **!emacro**. All directives that alter the execution of the macro are handled correctly within the **!if** statement (e.g. !goto, !return etc).

**!iif** provides an optimal way to write an **!if** statement with exactly one *body* line and no **!elif** or **!else** statements.

## EXAMPLE

The following macro segment creates the portion of a text file automatically. (yes believe me, this will be easier to understand then that last explanation....)

```
!if &sequal %curplace "timespace vortex"
    insert-string "First, rematerialize\n"
!endif
!if &sequal %planet "earth"     ;If we have landed on earth...
    !if &sequal %time "late 20th century"  ;and we are then
        ml-write "Contact U.N.I.T."
    !elif &sequal %time "pre 20th century"
        ml-write "start praying for a miracle"
    !else
        insert-string "Investigate the situation....\n"
```

```
        insert-string "(SAY 'stay here Sara')\n"
    !endif
!else
    set-variable %conditions @ml"Atmosphere conditions outside? "
    !if &sequal %conditions "safe"
        insert-string &cat "Go outside......" "\n"
        insert-string "lock the door\n"
    !else
        insert-string "Dematerialize..try somewhere else"
        newline
    !endif
!endif
```

The first **!if** statement in the above example can be optimised to:

```
!iif &sequal %curplace "timespace vortex"  insert-string "First, rematerialize\n"
```

## NOTES

There is a convention in **MicroEmacs** macro code of using 2 spaces to separate the *condition* from the *body* of an **!iif** statement making it easier to locate the start of the *body*.

## SEE ALSO

Variable Fuctions, **!goto**(4), **&equal**(4), **!return**(4), **$status**(5).

Ref: !if(4) File: m4fil008.4 Date: 2026/02/21

## NAME

!jump - Unconditional relative branch
!tjump - conditional relative branch

## SYNOPSIS

**!jump** *offset*
**!tjump** *condition offset*

## DESCRIPTION

Flow can be controlled within a MicroEmacs macro using the **!jump** directive. It takes as a numerical argument *offset*. The *offset* is a signed relative displacement, it may be a literal numeric value, a variable or an evaluated expression (see Variable Functions). The displacement to jump starts from the current **!jump** line. (i.e. **0 !jump**) would loop forever as it jumps to itself). Negative *offset* branches backwards, positive *offset* forwards.

A conditional relative branch, with a numerical displacement is specified using **!tjump**. This has an additional argument *condition* which is evaluated and if TRUE (Non-zero) then the branch is taken. The *condition* may be a variable or an evaluated expression.

**!jump** and **!tjump** are fast equivalents of **!goto**(4) and **!tgoto**(4), respectively. **!jump** should be used with care as these calls are source sensitive and unexpected results may be obtained if the *offset*'s are specified incorrectly.

## WARNING

Comments are not counted as valid lines within the relative displacement, these are stripped out when the macro is loaded. When using a relative branch ensure that ONLY the *code* lines are counted.

## EXAMPLE

For some seriously dirty macro tricks then the **!jump** directive becomes very useful. The following example is taken from the **Metris** macro (which is packed with goodies if you can find time to work out what it does !!). The following example uses the random number generator **$random**(5) to generate a random number which scaled and used as a **!jump** offset, thereby creating a *switch* type statement.

```
0 define-macro met-select-piece
    !jump &mul 5 &add 1 &div &mod $random 71 10
    set-variable :met-np1 " X "  ; 1st 3 lines are dummies to get offset right
    set-variable :met-np1 " X "
    set-variable :met-np1 " X "
    set-variable :met-np1 " X "
    set-variable :met-np1 " X "
    set-variable :met-np2 "XX "
    set-variable :met-np3 " X "
    set-variable :met-ncol %lyellow
    !return
    set-variable :met-np1 "XX "
    set-variable :met-np2 "XX "
    set-variable :met-np3 "   "
    set-variable :met-ncol %yellow
    !return
    set-variable :met-np1 "X  "
    set-variable :met-np2 "XX "
    set-variable :met-np3 " X "
    set-variable :met-ncol %lmagenta
```

```
    !return
    set-variable :met-np1 "  X"
    set-variable :met-np2 " XX"
    set-variable :met-np3 " X "
    set-variable :met-ncol %lgreen
    !return
    set-variable :met-np1 " X "
    set-variable :met-np2 " X "
    set-variable :met-np3 " XX"
    set-variable :met-ncol %magenta
    !return
    set-variable :met-np1 " X "
    set-variable :met-np2 " X "
    set-variable :met-np3 "XX "
    set-variable :met-ncol %green
    !return
    set-variable :met-np1 " X "
    set-variable :met-np2 " X "
    set-variable :met-np3 " X "
    set-variable :met-ncol %lblue
    !return
    set-variable :met-np1 " X "
    set-variable :met-np2 " X "
    set-variable :met-np3 "X X"
    set-variable :met-ncol %lred
!emacro
```

## SEE ALSO

Variable Fuctions, **!goto**(4), **!if**(4), **!repeat**(4), **!return**(4), **!tgoto**(4), **!while**(4).

Ref: !jump(4) File: m4fil032.4 Date: 2026/06/19

## NAME

!nmacro - Execute line as if not in a macro

## SYNOPSIS

**!nmacro** *command*

## DESCRIPTION

**!nmacro** causes *command* to be executed as if it were initiated from the command line by the user, rather than from the macro context. When MicroEmacs executes a macro, by default any input the command requires is expected on the same line immediately following the command. If a line is preceded by a **!nmacro** (or **!nma**) directive, the command is executed as if it was invoked from the command line by the user, as such, the rest of the line is ignored and all input is obtained directly from the user, as per normal command interaction.

## EXAMPLE

The following example is taken from macro file `meme3_8.emf` and shows how to add a buffer mode.

```
; Add a buffer mode
define-macro add-mode
    ; Has the require mode been given as an argument, if so add it
    !force 1 buffer-mode @1
    !if &not $status
        ; No - use 1 buffer-mode to add a mode
        !nma 1 buffer-mode
    !endif
!emacro
```

The first line checks that the mode to add has not already been given as a macro argument, e.g. by executing the following line

```
buffer-add-mode "view"
```

If this line fails then the argument was not specified and must be obtained from the user as normal.

## NOTES

Individual arguments may be obtained from the user using the **@mn**(4) interactive macro variables.

## SEE ALSO

**@mn**(4).

Ref: !nmacro(4) File: m4fil025.4 Date: 2026/06/19

## NAME

!return, !abort - Exit macro

## SYNOPSIS

**!return** [*n*]
**!abort** [*n*]

## DESCRIPTION

The **!return** directive causes the current macro to exit with a `TRUE` status, either returning to the caller (if any) or to interactive mode. If an argument *n* is specified then the return status is determined by the value of *n*.

**!abort** has the same effect as **!return** only always returning a `FALSE` status to halt the execution of any calling macro. If an argument *n* is given to **!abort** the bell is also rung, the valid values of `n` are the same as for the **!bell**(4) directive.

## EXAMPLE

The following example checks the current language and warns if it has not be set, i.e. `Default`.

```
;       Check the current language

!if &not &seq %language "Default"
    !return
!endif
ml-write "Warning - you have not setup the Language - use user-setup"
```

The following example is shows the logic of the **!return** directive:

```
; !return example
define-macro i-will-return
    ml-write "you will see me"
    !return
    ml-write "you wont see me"
!emacro

define-macro test-return
    ml-write "you will see me"
    i-will-return
    ml-write "you will see me"
!emacro
```

Similarly, for the **!abort** directive

```
; !abort example
define-macro i-will-abort
    ml-write "you will see me"
    !abort
    ml-write "you wont see me"
!emacro

define-macro test-abort
    ml-write "you will see me"
    i-will-abort
    ml-write "you wont see me"
!emacro
```

For the last two examples above, all the "**will**"s are displayed and none of the "**wont**"s are.

## SEE ALSO

**define-macro**(2), **!bell**(4), **!if**(4), **!goto**(4).

Ref: !return(4) File: m4fil010.4 Date: 2026/02/21

## NAME

!while, !repeat, !break, !continue, !done, !until - loop

## SYNOPSIS

**!while** *condition*
... loop body ...
[**!continue** — **!break**]
... loop body ...
**!done**

**!repeat**
... loop body ...
[**!break**]
... loop body ...
**!until** *condition*

## DESCRIPTION

The six loop directives can be mixed and matched as required, a loop must have one of the start loop directive and an end loop directive; it can optionally have as many **!break** and **!continue** directives as needed.

The **!repeat** directive always executes the loop contents; whereas the **!while** directive allows statements within the loop to be executed only if the given *condition* is `TRUE`, if the condition is `FALSE` then execution is continued on the line after the loop end statement (**!done** or **!until**).

The **!done** directive always returns control back to the loop start directive (**!while** or **!repeat** statement), whereas **!until** only loops back to the start if the given *condition* is `FALSE`, exiting the loop otherwise.

A **!continue** may be used in the loop, this immediately returns control to the loop start line (**!while** or **!repeat** statement), skipping the rest of the loop content and end statement (including any **!until** condition). A **!break** can be used to exit a loop, execution resumes on the line after the end loop statement.

Up to six loops can be nested within a single function/executed buffer at any one time. If more are required then a sub function could be defined and called or the **!goto**(4) and **!jump**(4) directives could be used in conjunction with the **!if**(4) statement to construct loops.

## EXAMPLE

For example, the following macro segment fills to the fill column with spaces.

```
!while &less $curcol $buffer-fill-col
    insert-string " "
    !if &equal %example "1"     ; Silly to show continue
        !continue               ; Goto !while
    !elif &equal %example "2"   ; Silly to show continue
        !break                  ; Goto line after !done
    !endif
    ml-write "You wont see me if %example = 1 or 2"
!done
ml-write "LoopA has exited"

set-variable #l1 10
```

```
!repeat
    10 ml-write &spr "LoopB: %d" #l1
!until &les &dec #l1 1 1
100 ml-write &spr "LoopB end: #l1 should = 0 : %d" #l1

set-variable #l1 10
!repeat
  10 ml-write &spr "LoopC: %d" #l1
  !iif &les &dec #l1 1 1 !break
!done
100 ml-write &spr "LoopC end: #l1 should = 0 : %d" #l1
```

## SEE ALSO

**!if**(4), **!goto**(4), **!jump**(4).

Ref: !while(4) File: m4fil011.4 Date: 2026/02/21

## NAME

&abs, &add, &sub, &mul, &div, &mod, &neg, &inc, &dec, &pinc, &pdec - Numeric macro operators

## SYNOPSIS

**&abs** *num1*
**&add** *num1 num2*
**&sub** *num1 num2*
**&multiply** *num1 num2*
**&divide** *num1 num2*
**&mod** *num1 num2*
**&negate** *num*
**&inc** *variable increment*
**&dec** *variable decrement*
**&pinc** *variable increment*
**&pdec** *variable decrement*

## DESCRIPTION

The numeric operators operate on variables or integers to perform integer computations, returning the integer result of the operation. The contents of the variables are interpreted as signed integers typically with a dynamic range of -2^31 <= *num* <= 2^31-1.

The operators may all be abbreviated to their three letter abbreviation (i.e. **&multiply** may be expressed as **&mul**). In all cases the first argument is completely evaluated before the second argument.

**&abs** *num1*
: Returns the absolute value of *num1* i.e. if *num1* is positive it returns *num1*, else *-num1*

**&add** *num1 num2*
: Addition of two numbers *num1* and *num2*. i.e. *num1* + *num2*

**&sub** *num1 num2*
: Subtract the second number *num2* from the first *num1* i.e. *num1* - *num2*.

**&multiply** *num1 num2*
: (Signed) Multiply *num1* by *num2*. i.e. *num1* * *num2*. **&mul** is the three letter abbreviation.

**&div** *num1 num2*
: Divide the first number *num1* by the second *num2*, returning the integer result. i.e. *num1* / *num2*. Note that an error will be thrown if *num2* is 0.

**&mod** *num1 num2*
: Divide the first number *num1* by the second *num2*, returning the integer remainder. i.e. *num1* % *num2*. Note that an error will be thrown if *num2* is 0.

**&negate** *num*
: Negate the integer (multiply by -1) i.e. *-num*. **&neg** is the three letter abbreviation.

Expression evaluation is prefix. Operators may be nested using a pre-fix ordering, there is no concept of brackets (in-fix notation). The expression `(2 * 3) + 4` is expressed as:

```
&add &mul 2 3 4
```

conversely `2 * (3 + 4)` is expressed as:

```
&mul 2 &add 3 4
```

The pre/post incrementing and decrementing operators provide a mechanism for stepping through indexed information without incurring the overhead of providing multiple statements to perform assignment operations. The *variable* argument MUST be the name of a variable, it cannot be an expression or an indirection. The *increment* may be any integer expression (including another auto (dec)increment). Note that *variable* is re-assigned with it's new value within the operator, therefore use with care when performing multiple (dec)increments within the same statement line. The four operators are defined as follows:

**&inc** *variable increment*
: Pre-increment the *variable* by *increment*, returning the incremented value i.e. *variable* += *increment*.

**&dec** *variable decrement*
: Pre-decrement the *variable* by *decrement*, returning the decrement value i.e. *variable* -= *decrement*.

**&pinc** *variable increment*
: Post-increment the *variable* by *increment*, returning the pre-increment value i.e. *variable*++., where the ++ value is determined by *increment*. The return value is the value of *variable* as passed by the caller, the next reference to *variable* uses the *variable*+*increment* value.

**&pdec** *variable decrement*
: Post-decrement the *variable* by *decrement*, returning the pre-decrement value i.e. *variable*–, where the – value is determined by *decrement*.

## EXAMPLE

Add two numbers together and assign to a variable:

```
set-variable %result &add %num1 %num2
```

Increment `%result` by 1 and add to `%result2`

```
set-variable %result  &add %result 1
set-variable %result2 &add %result2 %result
```

The previous example could have used the increment operators to achieve the same result in a single operation e.g.

```
set-variable %result2 &add %result2 &inc %result 1
```

## SEE ALSO

Variable Functions, **&great**(4).

## NAME

&and, &or, &not, &equal, &sequal - Logical macro operators

## SYNOPSIS

**&and** *log1 log2*
**&or** *log1 log2*
**&not** *log*

**&equal** *num1 num2*
**&great** *num1 num2*
**&less** *num1 num2*

## DESCRIPTION

The logical testing operators perform comparison tests, returning a boolean value of `TRUE` (1) or `FALSE` (0).

The functions may all be abbreviated to their three letter abbreviation (i.e. **&great** may be expressed as **&gre**). In all cases the first argument is completely evaluated before the second argument. Logical operators include:

**&and** *log1 log2*
: `TRUE` if the logical arguments *log1* and *log2* are both `TRUE`.

**&or** *log1 log2*
: `TRUE` if either one of the logical arguments *log1* and *log2* are `TRUE`.

**&not** *log*
: Logical NOT. Returns the opposite logical value to *log*.

The numerical logical functions operate with integer arguments:

**&equal** *num1 num2*
: `TRUE`. If numerical arguments *num1* and *num2* numerically equal. Abbreviated form of the function is **&equ**.

**&great** *num1 num2*
: `TRUE`. If numerical argument *num1* is greater than *num2*. Abbreviated form of the function is **&gre**.

**&less** *num1 num2*
: `TRUE`. If numerical argument *num1* is less than *num2* Abbreviated form of the function is **&les**.

Evaluation of the logical operators are left to right, the leftmost argument is fully evaluated before the next argument. The operator ordering is prefix notation (see **&add**(4) for an example of prefix ordering).

## EXAMPLE

Test for integers in the range greater than 12:

```
!if &great %i 12
    ...
```

Test for integers in the range 8-12, inclusive

```
!if &and &great 7 &less 13
    ...
```

## NOTES

MicroEmacs always evaluates all arguments operators BEFORE the result is obtained, this differs from most programming languages. Consider the following example:

```
!if &and &bmod "edit" &iseq @mc1 "Save buffer first [y/n]? " "nNyY" "y"
    save-buffer
!endif
```

This would not not work as the user may expect, the user would be prompted to save every time regardless of whether the buffer has been changed. Instead the following should be used:

```
!if &bmod "edit"
    !if &iseq @mc1 "Save buffer first [y/n]? " "nNyY" "y"
        save-buffer
    !endif
!endif
```

## SEE ALSO

Variable Functions, **&add**(4), **&sequal**(4), **&sin**(4), **&cond**(4).

Ref: &and(4) File: m4fil003.4 Date: 2022/06/12

## NAME

&ato, &gmod, &bmo, &ind, &inw, &exi - Miscellaneous functions

## SYNOPSIS

**&atoi** *char*
**&itoa** *num*
**&gmode** *mode*
**&bmode** *mode*
**&nbmode** *buffer mode*
**&inword** *char*
**&indirect** *str*
**&exist** *str*

## DESCRIPTION

These are a selection of miscellaneous functions providing tests and exchanging of information. The functions may all be abbreviated to their three letter abbreviation (i.e. **&indirect** may be expressed as **&ind**). In all cases the first argument is completely evaluated before the second argument.

**&atoi** *char*
: Converts the given character *char* to it's ASCII number which is returned. (see **&itoa**). Abbreviated command is **&ato**.

**&itoa** *num*
: Converts an integer *num* to it's ASCII character representation which is returned to the caller. Abbreviated command is **&ito**.

**&gmode** *mode*
: Returns `1` if the given mode *mode* is globally enabled. Allows macros to test the global mode state (see Operating Modes). Abbreviated command is **&gmo**.

**&bmode** *mode*
: Returns `1` if the mode *mode* is enabled in the current buffer. Allows macros to test the state of the buffer mode. Abbreviated command is **&bmo**.

**&nbmode** *buffer mode*
: Returns `1` if the mode *mode* is enabled in buffer *buffer* . Allows macros to test the state of a buffer mode other than the current. Abbreviated command is **&nbm**.

**&inword** *char*
: `TRUE`. If the given character *char* is a 'word' character, see **forward-word**(2) for a description of a 'word' character. Abbreviated command is **&inw**.

**&indirect** *str*
: Evaluate *str* as a variable. The *str* argument is evaluated and takes the resulting string, and then uses it as a variable name. i.e. a variable may reference another variable which contains the data to be referenced. Abbreviated command is **&ind**.

**&exist** *str*
: Tests for the existance of *str* which may be a buffer, variable or a command/macro name, returning `TRUE` if the buffer, variable or command does currently exist. Note

that when testing for the existence of a buffer the buffer name must be encased in ':'s, e.g. "`:*scratch*:`" to distinguish it from a check command/macro name. Abbreviated command is **&exi**.

## EXAMPLE

The **&exi** function is etremely useful in initializing, for example:

```
!if &not &exi %my-init
    ; %my-init is not yet defined so this is the first call
    set-variable %my-init 1
    .
    .
```

Or in all the file hooks a user defined extension is checked for and executed if defined:

```
define-macro fhook-c
    .
    .
    ; execute user extensions if macro is defined
    !if &exi my-fhook-c
        my-fhook-c
    !endif
!emacro
```

The **&ind** function deserves more explanation. **&ind** evaluates its string argument *str*, takes the resulting string and then uses it as a variable name. For example, given the following code sequence:

```
; set up reference table

set-variable  %one      "elephant"
set-variable  %two      "giraffe"
set-variable  %three    "donkey"

set-variable  %index "%two"
insert-string &ind %index
```

the string "`giraffe`" would have been inserted at the point in the current buffer.

The **&bmode** invocation allows a calling macro to determine the buffer mode state (see Operating Modes). Consider the following example which is a macro to perform a case insensitive alphabetic sort using the **sort-lines**(2) function. **sort-list** sorts according to the state of the **exact**(2m) mode, hence the macro has to determine the buffer state in order to be able to do the sort.

```
define-macro sort-lines-ignore-case
    set-variable #l0 &bmod exact
    -1 buffer-mode "exact"
    !if @?
        @# sort-lines
    !else
        sort-lines
    !endif
    &cond #l0 1 -1 buffer-mode "exact"
!emacro
```

The **&inword** function is shown in the following example. In this case the mouse is positioned over a word. The **&inword** function is used to determine if the cursor is on a valid word character, if so the cursor is placed at the start of the word.

```
define-macro mouse-control-drop-left
    set-cursor-to-mouse
    !if &inword @wc
        backward-word
        set-mark
        forward-word
```

```
    !else
        ...
    !endif
    copy-region
    set-cursor-to-mouse
!emacro
```

## SEE ALSO

Operating Modes, Variable Functions, **&sprintf**(4), **&equal**(4).

Ref: &atoi(4) File: m4fil005.4 Date: 2026/02/17

## NAME

&band, &bor, &bnot, &bxor - Bitwise macro operators

## SYNOPSIS

**&band** *num1 num2*
**&bor** *num1 num2*
**&bxor** *num1 num2*
**&bnot** *num*

## DESCRIPTION

The bitwise operators perform bit operations on numeric values returning a numerical result of the operation.

The functions may all be abbreviated to their three letter abbreviation (i.e. **&band** may be expressed as **&ban**). In all cases the first argument is completely evaluated before the second argument.

**&band** *num1 num2*
: Bitwise AND of *num1* and *num2* i.e. *num1* & *num2*.

**&bor** *num1 num2*
: Bitwise (inclusive) OR of *num1* and *num2* i.e. *num1* — *num2*.

**&bxor** *num1 num2*
: Bitwise (exclusive OR) XOR of *num1* and *num2* i.e. *num1* ^ *num2*.

**&bnot** *num*
: Bitwise NOT operator of *num*, inverts the state of all bits i.e. *num*.

Evaluation of the bitwise operators are left to right, the leftmost argument is fully evaluated before the next argument. The operator ordering is prefix notation (see **&add**(4) for an example of prefix ordering).

## SEE ALSO

Variable Functions, **&add**(4), **&and**(4), **&negate**(4), **&or**(4).

Ref: &band(4) File: m4fil021.4 Date: 2026/06/19

## NAME

&bstat - Retrieve current buffer statistic

## SYNOPSIS

**&bstat** <*stat*>

## DESCRIPTION

**&bstat** returns the specified <*stat*> on the current buffer. Valid <*stat*> values are:

**o** Returns a non-zero value if the buffer is out of date (i.e. the original file has changed and is newer than the buffer.

## EXAMPLE

The following example uses **&bstat** to determine if the buffer is out of date and write a simple message:

```
!if &bstat o
  ml-write "Buffer out of date"
!endif
```

## SEE ALSO

Variable Functions, **find-file**(2), **&stat**(4).

Ref: &bstat(4) File: m4fil037.4 Date: 2022/06/12

## NAME

&cat, &lef, &rig, &mid, &len, &slo, &trb - String macro operators

## SYNOPSIS

**&cat** *str1 str2*
**&lef** *str len*
**&right** *str index*
**&mid** *str index len*
**&len** *str*
**&slower** *str*
**&supper** *str*
**&trboth** *str*
**&trleft** *str*
**&trright** *str*

## DESCRIPTION

The string operators operate on character strings (**%** or **$** variables), performing general string manipulation, returning a string result.

The operators may all be abbreviated to their three letter abbreviation (i.e. **&right** may be expressed as **&rig**). In all cases the first argument is completely evaluated before the second argument.

**&cat** *str1 str2*
: Concatenate two string *str1* with *str2* to form a new string. i.e. *str1str2*

**&lef** *str len*
: Return *len* leftmost characters from *str*. If *str* length is shorter than *len* then the string itself is returned. A *len* of zero returns the empty string. A negative *len* returns all but the 0-*len* rightmost characters from *str*.

**&rig** *str index*
: Returns the rightmost characters of string *str* from index *index*. This function causes some confusion, consider **&lef** and **&rig** to be the string equivalents of their integer counterparts &div and &mod; **&rig** returns the remainder of the equivalent **&lef** function. Invocation with *index* set to zero returns *str*. A negative *index* returns the 0-*index* rightmost characters from *str*.

**&mid** *str index len*
: Extracts a sub-string from string *str*, starting at position *index* of length *len*. A negative *index* extracts the string starting from 0-*index* from the end of *str* with a length of *len*.

**&len** *str*
: Returns the integer length of the string (number of characters).

**&slower** *str*
: Returns the given string with all upper case characters converted to lower case.

**&supper** *str*
: Returns the given string with all lower case characters converted to upper case.

**&trboth** *str*
: Returns the given string trimmed of white spaces (i.e. ' ', '\t', '\r', '\n', '\Cl' and '\Ck') from both sides of the string.

**&trleft** *str*
: Returns the given string trimmed of white spaces from the left side of the string only.

**&trright** *str*
: Returns the given string trimmed of white spaces from the right side, or end, of the string only.

Evaluation of the strings is left to right, the leftmost argument is fully evaluated before the next argument. The operator ordering is prefix notation (see **&add**(4) for an example of prefix ordering).

## EXAMPLE

Concatenate two strings `abc` and `def` together:

```
set-variable %result &cat "abc" "def"
```

To concatenate three strings `abc`, `def` `ghi` together:

```
set-variable %result &cat "abc" &cat "def" "ghi"
```

or, a slightly different ordering:

```
set-variable %result &cat &cat "abc" "def" "ghi"
```

Retrieve the leftmost character of a string variable, modify the variable to contain the remainder.

```
set-variable %foo "abcdef"
set-variable %c   &lef %foo 1
set-variable %foo &rig %foo 1
```

Where `%c` = "`a`"; `%foo` = "`bcdef`" following evaluation.
To retrieve the characters `cde` into variable `%result` from the string "`abcdef`" use:

```
set-variable %result &mid "abcdef" 2 3
```

To retrieve the rightmost character from the string:

```
set-variable %foo "abcdef"
; using negative argument
set-variable %result &rig %foo -1
; using &len
set-variable %result &rig %foo &sub &len %foo 1
; using &mid to do the same thing
set-variable %result &mid %foo -1 1
set-variable %result &mid %foo &sub &len %foo 1 1
```

To get an input string from the user which is free of spaces at the start and end:

```
set-variable %result &trb @ml "Enter string"
```

## NOTES

The original **MicroEMACS** "**&rig** *str n*" function returns the last *n* characters from the string *str* this differs from the definition of **&rig** in this implementation. As most string decomposition is performed left to right, and to make **&lef** and **&rig** complement each other, the indexing of the function has been modified.

## SEE ALSO

Variable Functions, **&sin**(4), **&sequal**(4), **&lget**(4), **&sprintf**(4).

Ref: &cat(4) File: m4fil002.4 Date: 2022/06/12

## NAME

&cbind, &kbind, &nbind - Command/key binding operators

## SYNOPSIS

**&cbind** *key*
**&nbind** *key*
**&kbind** *n command*

## DESCRIPTION

**&cbind** returns the command bound to the given key sequence, **&cbind** can be abbreviated to **&cbi**. If the key is not bound then **&cbind** returns the string "*ERROR*".

**&nbind** returns the numerical argument associated with the given key binding, **&nbind** can be abbreviated to **&nbi**. If the key is not bound then **&nbind** returns the string "*ERROR*", if the binding has no argument then an empty string (`""`) is returned.

**&kbind** returns a key sequence bound to the given *command* with the given numerical argument *n*, use an empty string (`""`) for no argument. If no binding can be found then **&kbind** returns an empty string (`""`).

## EXAMPLE

The following example waits for the user to press a key, then prints what command the key is bound to.

```
ml-write "Enter key: "
set-variable #l0 @cgk
ml-write &spr "%s is bound to %s" #l0 &cbin #l0
```

## NOTES

In March 2001 **&kbind** was renamed **&cbind** and a new **&nbind** and **&kbind** added.

## SEE ALSO

Variable Functions, **global-bind-key**(2).

Ref: &cbind(4) File: m4fil026.4 Date: 2026/06/19

## NAME

&cond - Conditional expression operator

## SYNOPSIS

**&cond** *log expr1 expr2*

## DESCRIPTION

The conditional expression **&cond** provides an alternative way to write !if-!else-!endif constructs, e.g.:

```
!if &gre %a %b
    set-variable %z %a
!else
    set-variable %z %b
!endif
```

may be replaced with a conditional expression, breaking down the components then

*log* is **&gre %a %b**
*expr1* is **%a**
*expr2* is **%b**

rewriting the expression we get:

```
set-variable %z &cond &gre %a %b %a %b
```

This is far more concise, albeit a little less readable, but does improve the performance of macros as there is less information to interpret.

The **&cond** operator accepts three fields, ALL fields are evaluated although only one of the results *expr1* or *expr2* is used. The *log* field is a logical value, if it is non-zero (TRUE) then the result of the *expr1* evaluation is used, otherwise the result of *expr2* is used.

It should be noted that the conditional expression may be used in any construct i.e. **&add**(4), **&cat**(4), etc. the *expr* arguments may be strings, numbers or booleans the resultant value of the *expr* arguments is simply returned to the calling expression.

## SEE ALSO

Variable Functions, **&add**(4), **&great**(4).

Ref: &cond(4) File: m4fil022.4 Date: 2022/06/12

## NAME

&fabs, &fadd, &fsub, &fmul, &fdiv, &fsqrt - Floating point macro operators

## SYNOPSIS

**&fabs** *num*
**&fadd** *num1 num2*
**&fsub** *num1 num2*
**&fmul** *num1 num2*
**&fdiv** *num1 num2*
**&fsqrt** *num*

## DESCRIPTION

The floating point numeric operators operate on variables or literals (strings in the form of numbers) to perform floating point computations, returning double precision floating point numbers in scientific format. The contents of the variables are interpreted as floating point numbers with a 15 digits of precision and a range of +/-1.7e308.

The operators may all be abbreviated to their three letter abbreviation (i.e. **&fmultiply** may be expressed as **&fmu**). In all cases the first argument is completely evaluated before the second argument.

**&fabs** *num*
: Returns the absolute value of *num* i.e. if *num* is positive it returns *num*, else *-num*

**&fadd** *num1 num2*
: Addition of two numbers *num1* and *num2*. i.e. *num1* + *num2*

**&fsub** *num1 num2*
: Subtract the second number *num2* from the first *num1* i.e. *num1* - *num2*.

**&fmul** *num1 num2*
: (Signed) Multiply *num1* by *num2*. i.e. *num1* * *num2*.

**&fdiv** *num1 num2*
: Divide the first number *num1* by the second *num2*. i.e. *num1* / *num2*. Note that `1.#INF` is returned if *num2* is 0.

**&fsqrt** *num*
: Calculates the square root of *num*. Note that `-1.#INF` is returned if *num* is less than 0.

Expression evaluation is prefix. Operators may be nested using a pre-fix ordering, there is no concept of brackets (in-fix notation). The expression `(2 * 3) + 4` is expressed as:

```
&fadd &fmul 2 3 4
```

conversely `2 * (3 + 4)` is expressed as:

```
&fmul 2 &fadd 3 4
```

The number `123.4` can also be expressed as `1.234e2` and `0.001234` as `1.234e-3`.

## EXAMPLE

Add multiply a number by pi and print in a readable form:

```
ml-write &spr "%.10g" &fmul %num1 3.141592654
```

### NOTES

To format the results of these functions in a more usable form use commands '%f' and '%g' of the **&sprintf**(4) function.

### SEE ALSO

Variable Functions, **&add**(4), **&sprintf**(4).

Ref: &fabs(4) File: m4fil012.4 Date: 2022/06/12

## NAME

&find - Find a file on the search path
&which - Find a program on the path

## SYNOPSIS

**&find** <*basename*> <*extension*>
**&which** <*progname*>

## DESCRIPTION

**&find** searches for a named file <*basename*><*extension*> on the MicroEmacs search path defined by the variable **$search-path**(5), initialized by the system from the environment variables **$MEINSTALLPATH**(5) etc.

Each path component defined in **$search-path** is prepended to the constructed file name and it's existence is tested. If the file exists, then the FULL path name of the file is returned to the caller, otherwise the string "ERROR" is returned.

<*basename*>
: The base name of the file, excluding any extension.

<*extension*>
: The extension of the file name, this must be specified with the extension delimiter, typically dot ('.'). An empty string (i.e. '""') may be specified if no extension is required. **<extension>** can also be a '—' character separated list of extensions, see **&lset**(4) for more information on MicroEmacs list format and functions.

  **&which** searches for the given executable program <*progname*> on the system program search path defined by the environment variable **$PATH**. If the program is found, then the FULL path to the program is returned, otherwise the string "ERROR" is returned.

  Note that on dos and Windows only the extensions `.bat`, **.cmd**, `.com` and `.exe` are checked for.

## USAGE

**&find** is typically used with **insert-file**(2) and **find-file**(2) within macro scripts, and is used to locate user specific files.

## EXAMPLE

The following example uses **&find** to locate the uses 'C' template file. Given a **$search-path** setting of `/usr/bob/emacs:/usr/local/microemacs`:

```
insert-file &find "c" ".etf"
```

Would insert the file `/usr/bob/emacs/c.etf` if it existed, else the file `/usr/local/microemacs/c.et` if it exists.

## SEE ALSO

Variable Functions, **find-file**(2), **insert-file**(2), **$search-path**(5), **$MEINSTALLPATH**(5), **$MEUSERPATH**(5).

Ref: &find(4) File: m4fil024.4 Date: 2025/01/09

## NAME

&ldel, &lfind, &lget, &linsert, &lset - List manipulation functions

## SYNOPSIS

**&ldel** *list index*
**&lfind** *list value*
**&lget** *list index*
**&linsert** *list index value*
**&llength** *list*
**&lset** *list index value*

## DESCRIPTION

The list manipulation functions perform operations on specially formatted strings called lists. A list is defined as:

```
"|value1|value2|.....|valueN|"
```

Where '|' is the dividing character, this is not fixed to a '|', but is defined by the first character of the string. Following are all valid lists.

```
"|1|2|3|4|5|"
"X1X2X3X4X5X"
"\CAHello\CAWorld\CA"
"??"
```

The functions may all be abbreviated to their three letter abbreviation (i.e. **&linsert** may be expressed as **&lin**). In all cases the first argument is completely evaluated before the second or third argument.

**&ldel** *list index*

Creates a new list from deleting item *index* from *list*. If *index* is out of *list*'s range (0 < index <= # items in list) then *list* is returned unchanged.

**&lfind** *list value*

Returns the index whose item is the same as *value* in *list*. If *value* is not found in *list* then "0" is returned.

**&lget** *list index*

Returns the value of item *index* in *list*. If *index* is out of *list*'s range (0 < index <= # items in list) then an empty string is returned.

**&linsert** *list index value*

Creates a new list from inserting *value* into *list* at point *index*, thereby pushing item *index* to *index*+1 etc.

If *index* is 0, *value* is inserted at the beginning of the list.

If *index* is -2, *value* is inserted alphabetically (case sensitively) into the list.

If *index* is -3, *value* is inserted alphabetically (case insensitively) into the list.

Otherwise, if *index* is `-1` or greater that the number of items in *list* then *value* is inserted at the end of the list.

**&llength** *list*

Returns the list length, i.e. the number of items in the given list.

**&lset** *list index value*

Creates a new list from setting *index* of *list* to *value*. If *index* is out of *list*'s range (0 < index <= # items in list) then **&lset** behaves like **&linsert** and inserts alphabetically or at the end of the list.

## EXAMPLE

The following example moves item 4 in a list to position 2:

```
set-variable #l0 &lget %list 4
set-variable #l1 &ldel %list 4
set-variable %list &lins #l1 2 #l0
```

The following example is taken from vm.emf, it firstly checks where the user has entered a vm command, if not then the key is execute as normal, otherwise the appropriate vm command is executed.

```
define-macro vm-input
    set-variable #l2 @cck
    set-variable #l3 @cc
    !if &not &set #l0 &lfi "|esc h|delete|space|return|A|a|C|c|....|z|" #l2
        !if &not &seq #l3 "ERROR"
            execute-line &spr "!nma %s %s" &cond @? @# "" #l3
            !return
        !endif
        ml-write &spr "[Key \"%s\" not bound - \"esc h\" to view help]" #l2
        !abort
    !endif
    set-variable #l1 &lget "|%osd-vm-help osd|vm-del-windows|scroll-down|....
                        ....vm-goto-list|vm-Archive-box|vm-archive-box|....
                        vm-cut-all-data|0 vm-extract-data|...|vm-forward|" #l0
    execute-line #l1
!emacro
```

## NOTES

An error will be thrown if a list function is given an empty sting as the list, this is because the character \x00 cannot be used as the separator char.

## SEE ALSO

Variable Functions, **&mid**(4), **&cat**(4).

Ref: &ldel(4) File: m4fil029.4 Date: 2024/08/22

## NAME

&opt - MicroEmacs optional feature test

## SYNOPSIS

**&opt** *str*

## DESCRIPTION

This function can be used to test the availability of optional features in the current session of MicroEmacs. Some features, like spelling checker support, are a compilation option, other options like mouse support may also be unavailable on some platforms. The **&opt** function can be used by macros to check that required base functionality is available.

The function returns `1` in the given feature "*str*" is supported, otherwise it returns `0` if the feature is unknown or not supported in the running version.

Optional components of MicroEmacs are enabled/disabled at compile time, most options are configured by `MEOPT_`<*NAME*> `#define`'s within the source file `emain.h`. Following is a complete list of options, giving the **opt** string and `#define` label:

abb - MEOPT_ABBREV
: Abbreviation functionality (see **expand-abbrev**(2)).

cal - MEOPT_CALLBACK
: Callback and idle event handling (see **create-callback**(2)).

cli - MEOPT_CLIENTSERVER
: Client/server support (see Client-Server).

col - MEOPT_COLOR
: All color support (making hilighting redundent etc, see **add-color**(2)).

cry - MEOPT_CRYPT
: File encryption (see **crypt**(2m) mode).

deb - MEOPT_DEBUGM
: Macro debugging (see **$debug**(5)).

dir - MEOPT_DIRLIST
: Generate directory listing when loading a directory (see **find-file**(2) and **file-browser**(3)).

ext - MEOPT_EXTENDED
: Miscellaneous more advanced commands and features such as **append-buffer**(2), **generate-hash**(2) and **screen-poke**(2).

fen - MEOPT_FENCE
: Fence matching (see **user-setup**(3) and **fence**(2m)).

fra - MEOPT_FRAME
: Multiple frames (Internal or external, see opt "**mwf**" and command **create-frame**(2)).

has - MEOPT_CMDHASH
: Use a hash table for rapid command name lookup, not to be confused with **generate-hash**(2) which is an extended command.

hil - MEOPT_HILIGHT
: Hilight and user definable indentation rules (see **hilight**(2) and **indent**(2)).

hsp - MEOPT_HSPLIT
: Horizontal window splitting (see **split-window-horizontally**(2)).

ipi - MEOPT_IPIPES
: Interactive pipes (see **ipipe-shell-command**(2)).

ise - MEOPT_ISEARCH
: Incremental search (see **isearch-forward**(2)).

lbi - MEOPT_LOCALBIND
: Buffer, message-line and OSD local binding overrides (see **buffer-bind-key**(2)).

mag - MEOPT_MAGIC
: Regular expression search engine (see **magic**(2m) mode).

mjm - MEOPT_MAJORMODE
: File major-mode auto-detection and configuration (see **major-mode**(2)).

mou - MEOPT_MOUSE
: Mouse support (see **$mouse**(5)).

mwf - MEOPT_MWFRAME
: Multiple window frame support (see opt "**fra**").

nar - MEOPT_NARROW
: Buffer narrowing (see **narrow-buffer**(2)).

nex - MEOPT_FILENEXT
: Location list stepping (see **get-next-line**(2)).

osd - MEOPT_OSD
: On Screen Display GUI support (see **osd**(2)).

pri - MEOPT_PRINT
: Printing support (see **print-buffer**(2)).

rcs - MEOPT_RCS
: File Revision Control Support (see **$rcs-co-com**(5)).

reg - MEOPT_REGISTRY
: Internal registry and history support (see **read-registry**(2)).

scr - MEOPT_SCROLL
: Window scroll-bar support.

soc - MEOPT_SOCKET
: FTP and HTTP URL support via sockets (see **find-file**(2)).

spa - MEOPT_SPAWN
: External process launching (see **shell-command**(2)).

spe - MEOPT_SPELL
: Spelling checker support (see **spell**(2)).

ssl - MEOPT_OPENSSL
: FTPE, FTPS and HTTPS Secure URL support via OpenSSL (see **find-file**(2) and **%http-flags**(5)).

tag - MEOPT_TAGS
: Tags support (see **find-tag**(2)).

tim - MEOPT_TIMSTMP
: File timestamping on save (see **time**(2m) mode).

typ - MEOPT_TYPEAH
: Input detect or 'type-ahead' for background processing support.

und - MEOPT_UNDO
: Undo support (see **undo**(2)).

wor - MEOPT_WORDPRO
: Word-processor style commands like **fill-paragraph**(2) (see **forward-paragraph**(2)).

## EXAMPLE

The following example checks for URL support and if not available it pops up an error:

```
!if &not &opt "soc"
    osd-dialog "Opt Test" "Error: No URL support!" "  &OK  "
!endif
```

## NOTES

The output of command **about**(2) includes a full list of all available optional features.

For secure URL (https) support to work the binary must be built with **MEOPT_OPENSSL** and the appropriate `libssl` and `libcrypto` runtime libraries must be available; **&opt** does not confirm the availability of the libraries.

## SEE ALSO

**about**(2), Building MicroEmacs.

Ref: &opt(4) File: m4fil035.4 Date: 2026/01/17

## NAME

&reg - Retrieve a registry value (with default)

## SYNOPSIS

**&reg** *root default*

## DESCRIPTION

**&reg** retrieves the value of a node defined by *root* from the registry, this is an absolute path.

The *default* value is the value of the node to return if the registry node does not exist.

## EXAMPLE

The following example is taken from `me.emf` and uses the registry to retrieve some of the default configuration files:

```
; Load in the color setup
!force execute-file &reg &spr "/history/%s/color" $platform "color"
; execute company setup
!if &not &seq &set #l0 &reg "/history/company" ""
    !force execute-file #l0
!endif
```

## SEE ALSO

**copy-registry**(3), **delete-registry**(2), **find-registry**(2), **list-registry**(2), **mark-registry**(2), **read-registry**(2), **set-registry**(2). **erf**(8).

Ref: &reg(4) File: m4fil028.4 Date: 2022/06/12

## NAME

&rep, &irep, &xrep, &xirep - Replace string in string functions

## SYNOPSIS

**&rep** *str1 str2 str3*
**&irep** *str1 str2 str3*
**&xrep** *str1 str2 str3*
**&xirep** *str1 str2 str3*

## DESCRIPTION

These functions search for *str2* in *str1*, replacing it with *str3*, returning the resultant string.

The functions may all be abbreviated to their three letter abbreviation (i.e. **&xirep** may be expressed as **&xir**). In all cases the first argument is completely evaluated before the second and third arguments.

**&rep** *string search replace*
: Searches for the *search* string in the given *string* using a simple case sensitive exact match algorithm. Any occurrences are removed from *string* and *replace* is inserted in its place. Either of the 3 input strings can be the empty string (`""`).

**&irep** *string search replace*
: **&irep** is identical to **&rep** except a case insensitive search algorithm is used.

**&xrep** *string regex-search regex-replace*
: **&xrep** can be used to access the more powerful regular expression searching capabilities. The function is similar to **&rep** except it takes a regex search string and the replacement string may also refer to all or part of the matched string. See Regular Expressions for information on the *regex* format.

**&xirep** *string regex-search regex-replace*
: **&xirep** is identical to **&xrep** except a case insensitive regex search is used.

## EXAMPLE

The following example turns a UNIX format file name (using a '/' to divide directories - like MicroEmacs) into an windows format name (using a '\'):

```
set-variable #l0 &rep #l0 "/" "\\"
```

The following example replaces one or more white spaces in the variable with a single space, this is an easy way to remove unnecessary spaces:

```
set-variable #l0 "This   is   not   so    spacey    after   xrep"
set-variable #l0 &xrep #l0 "\\s +" " "
ml-write #l0
```

## SEE ALSO

Operating Modes, Variable Functions, **&sequal**(4), **&sin**(4).

Ref: &rep(4) File: m4fil000.4 Date: 2022/06/12

## NAME

&seq, &iseq, &sle, &sgre, &xseq, &xiseq - String logical macro operators

## SYNOPSIS

**&sequal** *str1 str2*
**&isequal** *str1 str2*
**&sless** *str1 str2*
**&sgreat** *str1 str2*
**&xsequal** *str1 regex*
**&xisequal** *str1 regex*

## DESCRIPTION

The string logical testing operators perform string comparison tests, returning a boolean value of `TRUE` (1) or `FALSE` (0).

The functions may all be shortened to their three letter abbreviation (i.e. **&sequal** may be expressed as **&seq**). In all cases the first argument is completely evaluated before the second argument. String logical operators include:

**&sequal** *str1 str2*
: `TRUE` if the two strings *str1* and *str2* are the same. Abbreviated form of the function is **&seq**.

**&sless** *str1 str2*
: `TRUE` if string *str1* alphabetically less than *str2*. Abbreviated form of the function is **&sle**.

**&sgreat** *str1 str2*
: `TRUE` if string *str1* alphabetically larger than *str2*. Abbreviated form of the function is **&sgr**.

**&isequal** *str1 str2*
: `TRUE` if the two strings *str1* and *str2* are the same ignoring letter case. Abbreviated form of the function is **&ise**.

**&xsequal** *str1 regex*
: `TRUE` if the string *str1* matches the *regex* (case sensitive). Abbreviated form of the function is **&xse**. See Regular Expressions for information on the *regex* format.

**&xisequal** *str1 regex*
: `TRUE` if the string *str1* matches the *regex* (case insensitive). Abbreviated form of the function is **&xis**. See Regular Expressions for information on the *regex* format.

Evaluation of the string logical operators are left to right, the leftmost argument is fully evaluated before the next argument. The operator ordering is prefix notation (see **&add**(4) for an example of prefix ordering).

## EXAMPLE

Test for variable **$buffer-bname**(5) is equal to `*scratch*`:

```
!if &seq $buffer-bname "*scratch*"
    ...
```

The following example tests a character is in the range `a-z`:

```
!if &not &and &sle %c "a" &sgr %c "z"
    ...
```

The following example inserts the string "`c`" into the alphabetically order string list **%test-list**:

```
set-variable %test-list "|a|b|d|e|"
set-variable %test-insert "c"

set-variable #l0 1
!while &and &not &seq &lget %test-list #l0 "" ...
        ... &sle &lget %test-list #l0 %test-insert
    set-variable #l0 &add #l0 1
!done
set-variable %test-list &lins %test-list #l0 %test-insert
```

The first test on the **!while &and** conditional checks that the current item in the list is not an empty string (""). If it is the end of the list has been reached.
The following example tests the current buffers file name for a ".c" extension:

```
!if &xse $buffer-fname ".*\\.c"
    ...
```

Note the '\' character is needed to protect the second '`.`', i.e. so that it does not match any character and the second '\' is required as the string is first parsed by the macro interpreter which changes it to "`.*\.c`" which is then interpreted as a regex.

## SEE ALSO

Variable Functions, **&sin**(4), **&slower**(4), **&rep**(4), **&add**(4), **&equal**(4), **&cond**(4), Regular Expressions.

Ref: &sequal(4) File: m4fil030.4 Date: 2022/06/12

## NAME

&set - In-line macro variable assignment

## SYNOPSIS

**&set** <*var*> <*expr*>

## DESCRIPTION

**&set** performs an in-line macro variable assignment assigning a variable <*var*> the value of the expression <*expr*>, returning the evaluated result to the caller. <*expr*> may be numeric, boolean or a string expression.

**&set** is typically used for defining (and simultaneously using) indices e.g. as with **add-color**(2) or **add-color-scheme**(2). This is a short-hand of **set-variable**(2).

## EXAMPLE

The following example uses**&set** to define new colors (see `color.emf`):

```
; Standard colors
add-color &set %white     0 200 200 200
add-color &set %black     1 0   0   0
add-color &set %red       2 200 0   0
add-color &set %green     3 0   200 0
add-color &set %yellow    4 200 200 0
add-color &set %blue      5 0   0   200
add-color &set %magenta   6 200 0   200
add-color &set %cyan      7 0   200 200
```

## SEE ALSO

Variable Functions, **&inc**(4), **set-variable**(2).

Ref: &set(4) File: m4fil023.4 Date: 2022/06/12

## NAME

&sin, &isin, &rsin, &risin - String in string test functions

## SYNOPSIS

**&sin** *str1 str2*
**&isin** *str1 str2*
**&rsin** *str1 str2*
**&risin** *str1 str2*

## DESCRIPTION

These functions test for the existence of *str1* in *str2*, returning the position of the string in *str2* or 0 if not found.

The functions may all be abbreviated to their three letter abbreviation (i.e. **&risin** may be expressed as **&ris**). In all cases the first argument is completely evaluated before the second argument.

**&sin** *str1 str2*
: Returns `0` if string *str1* does not exists in string *str2*. Otherwise the function returns the character position + 1 of the location of the first character of the first occurrence of *str1*.

**&isin** *str1 str2*
: Returns `0` if case insensitive string *str1* does not exists in string *str2*. Otherwise the function returns the character position + 1 of the location of the first character of the first occurrence of *str1*.

**&rsin** *str1 str2*
: Returns `0` if string *str1* does not exists in string *str2*. Otherwise the function returns the character position + 1 of the location of the first character of the last occurrence of *str1*.

**&risin** *str1 str2*
: Returns `0` if case insensitive string *str1* does not exists in string *str2*. Otherwise the function returns the character position + 1 of the location of the first character of the last occurrence of *str1*.

## EXAMPLE

The **&sin** and similar functions are useful for two different purposes. Consider the following example, this utilizes **&sin** in two different contexts. `!while &not &sin @wc " \t\n"` is a test for the end of the number, i.e. a white space character (<tab>, <SPACE> or <NL>). The invocation `set-variable #l1 &isin @wc "0123456789abcdef"` is subtly different. In this case the return value is used to convert the character to it's integer hex value by using the value returned by **&isin**.

```
;
; calc-hexnum
; Convert the sting from the current position in the buffer
; to a hexadecimal number.
define-macro calc-hexnum
    forward-delete-char
    forward-delete-char
    set-variable #l0 0
```

```
    !while &not &sin @wc " \t\n"
        set-variable #l1 &isin @wc "0123456789abcdef"
        !if &not #l1
            ml-write "Bad Hex number found"
            !abort
        !endif
        set-variable #l0 &mul #l0 16
        set-variable #l0 &add #l0 &sub #l1 1
        forward-delete-char
    !done
    insert-string #l0
!emacro
```

The **&rsin** function is very similar to sin except the value return is the position of the last occurrence of the string in the given string instead of the first. This is particularly useful when extracting the path or file name from a complete file name. For example, given a UNIX style file name such as `"/usr/local/bin/me"` the path can be obtained using `set-variable %path &lef %pathfile &rsin "/" %pathfile` and the file name by using `set-variable %file &rig %pathfile &rsin "/" %pathfile`

## SEE ALSO

Operating Modes, Variable Functions, **&sequal**(4), **&rep**(4).

Ref: &sin(4) File: m4fil031.4 Date: 2022/06/12

## NAME

&sprintf - Formatted string construction

## SYNOPSIS

**&sprintf** *format args*

## DESCRIPTION

The **&sprintf** function (or **&spr** in it's abbreviated form) provides a mechanism to generated a formatted string, similar to the 'C' programming language **sprintf**(2) function.

The **&sprintf** function is generally used where a number of different sources of information have to be converted and joined together to form a new string. It is possible to do this using **&cat**(4), but it does become complicated if the number of strings to be spliced together is greater than about 4, **sprintf** alleviates these problems and results in faster execution. Where only two, or three strings are to be concatenated **&cat** provides better execution times.

The **&sprintf** function produces a string construct for the *format* and a caller determined number of arguments *args* (variable arguments). The *format* string may contain special '**%**' formatting commands to insert strings and numbers into the base *format* string. The format for the '**%**' commands is "**%nc**" where:

**n** An optional numerical argument, the interpretation of the numeric value is determined by the following command (**c**). As in 'C' it may be preceded by a minus sign, and may be followed by a full stop and a second number.

**c** The command determines the interpretation of the next argument *arg* which are specified as follows:

**d** (Decimal integer)
Expects a single numeric argument *arg* which is inserted into the *format* string as decimal text string. If *n* is specified then the inserted text string is fixed to *n* character in length, if *n* is prefixed with a 0 then the number is padded with zeros otherwise spaces are used.

**e** (floating point number - scientific) **f** (floating point number)

**g** (floating point number - shorter of **e** or **f**)
Expects a single numeric argument *arg* which is inserted into the *format* string as a floating point number string. If *n* is specified it must be of the form [min-width][.<precision>] where either the [min-width] or the [.<precision>] can be omitted.

**n** (Repeat String)
Expects two arguments *arg*, the first is a numeric argument giving the number of times to insert the given string (the second argument). The optional argument *n* is ignored if specified.

**o** (Octal integer)
Expects a single numeric argument *arg* which is inserted into the *format* string as an octal text string. If *n* is specified then the inserted text string is fixed to *n* character in length, if *n* is prefixed with a 0 then the number is padded with zeros otherwise spaces are used.

**s** (String)
: Expects a single argument *arg* which is a string to be inserted into the *format* string. If *n* is given it is the minimum field width and the string is padded with spaces on the left, or on the right when *n* is prefixed with a '−'. A '**.**' followed by a number gives the maximum number of characters to take from the string. To insert a string more than once use the **n** command.

**x** (Hexadecimal integer)
: Expects a single numeric argument *arg* which is inserted into the format string as hexadecimal text string. If *n* is given then the inserted text string will be fixed to *n* character in length, if *n* is prefixed with a 0 then the number is padded with zeros otherwise spaces are used.

**X** (Hexadecimal integer)
: Same as **x** only using uppercase letters.

**%**
: Inserts a single '%', *n* has no effect.

The **&sprintf** function may be nested (i.e. a string argument to **&sprintf** may be the result of another **&sprintf** invocation). Although this type of construct is not generally required !!

## EXAMPLE

The following examples show how the command may be used:

```
set-variable %result &sprintf "Foo [%s%s]" "a" "b"
```

generates "`Foo [ab]`"

```
set-variable %result &sprintf "Foo [%n%s]" 10 "a" "b"
```

generates "`Foo [aaaaaaaaaab]`".

```
set-variable %result &sprintf "[%d] [%3d] [%x] [%3x]" 10 11 12 13
```

generates "`[10] [ 11] [c] [  d]`"

## NOTES

It is the callers responsibility to ensure that the correct number of arguments is supplied to match the requested formatting string. The results are undefined if an incorrect number of arguments are supplied.

An error will be thrown if the **%** command is not one of the supported ones listed above.

## COMPATIBILITY

Prior to August 2026 the **s** command treated *n* as a repeat count rather than a field width, so "`%13s`" inserted the string thirteen times. Macros written against the older behaviour must be changed to use the **n** command, which is unaffected:

```
; before - inserted the box character 13 times
set-variable %line &sprintf "%13s" %box-char
; now
set-variable %line &sprintf "%n" 13 %box-char
```

A "`%`" followed by digits and an "`s`" is the construct to look for.

## SEE ALSO

Variable Functions, **&cat**(4).

## NAME

&stat - Retrieve a file statistic

## SYNOPSIS

**&stat** <*stat*> <*filename*>

## DESCRIPTION

**&stat** returns the specified <*stat*> on the given <*filename*>. Valid <*stat*> values are:

**a** Returns the absolute file name, corrects relative paths and symbolic links, i.e. on UNIX if the filename is a symbolic link it returns the file name the link points to (recursive), otherwise returns the file name.

**d** Returns the file's modification time stamp as an integer, larger values indicate a later time.

**i** Returns information on the file in a MicroEmacs list formatted string. The fields within the list are defined as follows:

1 File location, which will be one of the following:

```
F   File is an ftp URL file (see note).
H   File is a http URL link (see note).
L   File is a local file (may be prefixed with 'file:').
T   File is a tfs file (tfs://...).
```

2 File type, which will be one of the following:

```
D   File is a directory.
H   File is a http URL link (no further information is known).
L   File is a symbolic link (UNIX).
N   File is an untouchable system file.
R   File is a regular file.
X   File does not exist.
```

Note the information returned here is different to the 't' flag below as the type of an ftp file is evaluated, i.e. `D`, `R` or `X` will be returned.

3 Symbolic file type, which will be the empty string unless the file type above is 'L' in which case it will be one of the following:

```
D   File is a directory.
N   File is an untouchable system file.
R   File is a regular file.
X   File does not exist.
```

4 File permissions or attributes which are returned as an integer value specific to the Operating System.

5 and 6 File size given as an upper and lower 32 bit integers for large file support, these values will be the empty string if the file size could not be determined.

7 File modification time given as "`YYYYMMDDhhmmss`", this field has not yet been implemented.

**m** Returns the file's modification time stamp as a string, the format of the string is the same as **$time**(5).

**r** Returns a non-zero value if the user has permission to read the given file.

**s** Returns the size of the file in bytes.

**t** Returns the type of the file, where values returned are

```
X   File does not exist.
R   File is a regular file.
D   File is a directory.
H   File is a http URL link (see note).
F   File is an ftp URL file (see note).
N   File is an untouchable system file.
```

**w** Returns a non-zero value if the user has permission to write to the given file. Note that `0` is simply returned for ftp URLs whereas the user may have write permission depending on the server and login credentials supplied.

**x** Returns a non-zero value if the user has permission to execute the given file.

## NOTES

The URL type is determined from the start of the file name, e.g. http://..., and its existence is never verified if a http, nor for an ftp URL except for an **i** stat.
The following URL types are supported depending on the build of MicroEmacs:

```
file:     Local file.
ftp://    Standard ftp URL (default port 21).
ftpe://   Explicit secure ftp (default port 21).
ftps://   Implicit secure ftp (default port 990).
http://   Standard http URL (default port 80).
https://  Secure http URL (default port 443).
tfs://    Tacked-on file system URL.
```

## EXAMPLE

The following example is a macro which, given a file name, uses **&stat** to check that file file is regular:

```
define-macro test-file
    !force set-variable #l0 @1
    !if &not $status
        set-variable #l0 @ml04 "File name"
    !endif
    !if &not &equ &stat "t" #l4 1
        ml-write &spr "[%s is not a regular file]" #l0
        !abort
    !endif
!emacro

test-file "foobar"
```

The macro can be passed a file name and aborts if the file is not regular, there by returning the state.
The follow example checks that a file is not empty, this is used by **mail-check** to test for any incoming mail.

```
    !if &gre &stat "s" %incoming-mail-box
        ml-write "[You have new mail]"
    !endif
```

## SEE ALSO

Variable Functions, **find-file**(2), **&bstat**(4).

Ref: &stat(4) File: m4fil027.4 Date: 2023/02/21

## NAME

&uni, &uci, &ufi - Miscellaneous functions

## SYNOPSIS

**&uni** *int*
**&uci** *char*
**&ufi** *string*

## DESCRIPTION

These three functions provide information about Unicode charater support within the current MicroEmacs character-set.

**&uni** *int*

Returns information regarding the Unicode character *int*.

**&uci** *char*

Returns information regarding the MicroEmacs character-set character *char*.

**&ufi** *string*

Returns information regarding the UTF-8 encoded Unicode character *string*.

All three functions return a MicroEmacs formatted list of information of the follow form:

"*—flags—Unicode—char—utf8-len—utf8—*"

*flags* is a bitwise flag where bit 1 indicates which the given input was a valid Unicode character, see notes below and bit 2 indicates whether the input is supported in the current MicroEmacs display character-set.

*Unicode* is the hexadecimal value representing the input or zero if the input cannot be mapped to a valid Unicode character (bit 1 of *flags* is clear). The value is always 6 digits long with a `0x` hexadecimal number prefix.

*char* is the MicroEmacs display character-set equivalent of the input or the character 0x07 if the input is not supported (bit 2 of *flags* is clear).

In the case of **&ufi**, *utf8-len* is set to the length of *string* parsed, this may be less than the total length of *string*, otherwise *utf8-len* is the string length of *utf8*. Note that if the input character is 0x00 (the Nul character) the *utf8* string is "" (the empty string and 0 in length) but the *utf8-len* is

*utf8* is the UTF-8 equivalent of the input or the character string 0xEF 0xBF 0xBD (U+FFFD - replacement char) if the input to **&uni** or **&uci** cannot be mapped to a valid Unicode character.

## EXAMPLE

The following macro code can be used to convert the UTF-8 encode character string at the current location to the current display character set equivalent:

```
set-variable #l0 &ufi &mid @wl $window-col 4
!if &not &band 1 &lget #l0 1
  ml-write "Error: String at current location is not a valid UTF-8 char"
!elif &not &band 2 &lget #l0 1
```

```
  ml-write "Error: UTF-8 character not supported in current dsplay charset"
!else
  &lget #l0 4 forward-delete-char
  insert-string &lget #l0 3
!endif
```

## NOTES

In 2003 the Unicode character range was officially restricted to 0x0000-0x10ffff (see RFC 3629) which means that a 6 digit hexadecimal number can be used to represent all possible Unicode characters and the longest UTF-8 string is 4 bytes (0xF4 0x8F 0xBF 0xBD is the highest assigned char which is for private use). Currently the highest assigned non-private character is U+E01EF so 5 digits will cover all publicly used characters.

The mapping to and from the current MicroEmacs display character-set is configured by flag **c** of command **set-char-mask**(2).

In MicroEmacs character 0x07 is used to represent an unmappable character, this is better than using a character like '?' which is in common use. If extended character rendering is being used (see bit **0x10000** of **$system**(5)) the character is drawn as a crossed diamond, an attempt to represent the Unicode replacement character.

## SEE ALSO

Variable Functions, **set-char-mask**(2), **change-buffer-charset**(3), **&lget**(4), **&equal**(4).

Ref: &uni(4) File: m4fil038.4 Date: 2024/05/09

## NAME

@fs - Frame store variable
@fss - Frame store scheme variable

## SYNOPSIS

**@fs** *row column*
**@fss** *row column*

## DESCRIPTION

The frame store variable **@fs** permits macros to obtain the character currently being drawn on the screen at the specified location. If the given value of *row* or *column* is out range, i.e. less than zero or greater than or equal to the screen size (see **$frame-width**(5)) then the value returned is the empty string (i.e. `""`).

The frame store scheme variable **@fss** operates in the same way as **@fs** but returns the hilighting scheme number as opposed to the character.

These variables cannot be set and are only updated during a screen update, this means that macros that change the cursor position need to redraw the screen before using this variable (see **screen-update**(2)).

## EXAMPLE

The following example gets the word under the current mouse position, this may not be the current cursor position:

```
define-macro word-under-mouse
    set-variable #l0 $mouse-y
    set-variable #l1 $mouse-x
    !if &not &inw @fs #l0 #l1
        ml-write "[mouse not over a word]"
        !return
    !endif
    set-variable #l2 @fs #l0 #l1
    set-variable #l1 &sub #l1 1
    !if &inw @fs #l0 #l1
        set-variable #l2 &cat @fs #l0 #l1 #l2
        !jump -3
    !endif
    set-variable #l1 $mouse-x
    set-variable #l1 &add #l1 1
    !if &inw @fs #l0 #l1
        set-variable #l2 &cat #l2 @fs #l0 #l1
        !jump -3
    !endif
    ml-write &spr "[mouse is over the word \"%s\"]" #l2
!emacro
```

Knowledge of the display scheme as returned by the **@fss** variable is a little more obscure than the previous example. This variable is currently used to determine if there is a spelling error under the cursor position. When **auto-spell**(3) is enabled then spelling errors are hilighted in a different colour, return of the hilighting scheme allows the caller to determine that there is a spelling error at the current position, this could also be applied to collapsed regions etc. The following example is taken from `mouseosd.emf` to determine if there is a spelling error at the mouse position.

```
set-variable #l0 &band $mouse-pos 15
!if &equ #l0   0
    ; Text window - must goto the original position or cut etc wont work - must dup the position
```

```
0 show-region
set-variable #l2 $result
set-variable #l0 20
osd 27 5  "D"
!if &seq $buffer-input "auto-spell-input"
    !if &equ @fss $mouse-y $mouse-x .scheme.spell-error
        auto-spell-test
        !if &seq &lef $result 1 "E"
            set-position "\x83"
            set-variable .auto-spell-osd.word &rig $result 1
            osd 27 5  "M" "Auto Spell" .osd.spell-auto auto-spell-osd
            set-variable #l0 &add #l0 1
        !endif
    !endif
!endif
goto-position "\x82"
....
```

## SEE ALSO

**$frame-width**(5), **screen-update**(2), MacroArguments, MacroNumericArguments, **define-macro**(2).

Ref: @fs(4) File: m4fil034.4 Date: 2026/02/23

## NAME

@y - Yank buffer variable

## SYNOPSIS

**@y** - Yank buffer variable

## DESCRIPTION

The *Yank Buffer Variable* **@y** retrieves the current **yank**(2) string from the kill buffer and returns it to the caller.

The **@y** variable may be set to change the kill ring buffer.

The *Yank Buffer Variable* may be specified with a numeric qualifier to access one of the historical yank buffers **@y0** to **@y15** to directly extract text from the kill-ring.

## EXAMPLE

The current contents of the yank buffer can be obtained using **@y**, so to set variable `#l1` to the current or last word if the cursor is not in a word, simply use:

```
forward-char
backward-word
set-mark
forward-word
copy-region
set-variable #l1 @y
```

## SEE ALSO

**yank**(2), MacroArguments, MacroNumericArguments, **define-macro**(2).

Ref: @y(4) File: m4fil020.4 Date: 2022/06/12

## NAME

@clk, @cl - Last key or command name
@cck, @cc - Current key or command name
@cgk, @cg - Get a key or command name from the user
@cqk, @cq - Get a quoted key or command name from the user

## SYNOPSIS

**@clk**
**@cl**
**@cck**
**@cc**
**@cgk**
**@cg**
**@cqk**
**@cq**

## DESCRIPTION

The Command Variables allow macros to obtain MicroEmacs input commands and keystrokes from the user. The general format of the command is:

**@c***i*[**k**]

Where,

*i* Determines the source of the input as follows:

- **l** The last input entered.
- **c** The current input entered.
- **q** Provides a low level character input mechanism, obtaining a single raw character input from the user. The input fetch does not interact with the message line and the user is NOT prompted for input (use **ml-write**(2) to create your own message). **@cq** is very low level, it is generally preferable to use **@cg** which provides a more intelligent binding.
- **g** Like **@cq**, **@cg**[**k**] gets a single character input, however if the input is bound to a function then the function name is returned instead of the character e.g. if `^F` or `<left-arrow>` is depressed then **forward-char** is returned. This has distinct advantages over **@cq** as the binding becomes device independent and executes on all platforms. In addition, it honors the users bindings, however bizarre.

**k** When, omitted command input is returned to the caller (i.e. the name of the command, such as "`forward-char`"). When present, the raw keystroke is returned to the caller, i.e. "`^F` (control-F).

The **@cl**, **@clk**, **@cc** and **@cck** variables can also be set, this feature can be used by macros to change the command history. While setting the current command is limited in use, setting the last command can be immensely useful, consider the following macro code:

```
kill-line
forward-line
set-variable @cl kill-line
kill-line
```

Without the setting of the **@cl** variable, the current kill buffer will contain only the last line. But the setting of **@cl** to kill-line fools MicroEmacs into thinking the last command was a kill command so the last kill line as appended to the current yank buffer, i.e. the kill buffer will have both lines in it.

This feature can be used for any command whose effect depends on the previous command. Such commands include **forward-line**(2), **kill-region**(2), **reyank**(2) and **undo**(2). This feature should not be abused as unexpected things may happen.

### Summary

**@cl** Get or set the last command.

**@clk** Get or set the last key stroke.

**@cc** Get or set the current command.

**@cck** Get or set the current keystroke.

**@cg** Get a command name from the user.

**@cgk** Get a keystroke from the user.

**@cq** Get a quoted command name from the user.

**@cqk** Get a quoted keystroke from the user.

### EXAMPLE

The following example shows how the **@cc** and **@cl** commands are used:

```
define-macro current-last-command
    insert-string &spr "Last key [%s] name [%s]\n" @clk @cl
    insert-string &spr "Current key [%s] name [%s]\n" @cck @cc
!emacro
```

Pressing the up key and then executing this macro using execute-named-command (esc x) will insert the lines:

```
Last key [up] name [backward-line]
Current key [esc x] name [execute-named-command]
```

**@cg** like **@cq** gets a single character input, however if the keyboard input is bound to a function then the function name is returned instead of the character e.g. if ^F or <left-arrow> is depressed then **forward-char** is returned. This has distinct advantages over **@cq** as the binding becomes device independent and executes on all platforms, additionally it honors the users bindings, however bizarre.

**@cq** provides a low level character input mechanism, obtaining a single raw character input from the user. This does not interact with the message line and the user is not prompted for input (use **ml-write**(2) to create your own message). **@cq** is very low level, it is generally preferable to use **@cg** which provides a more intelligent binding.

## EXAMPLE

The following example is taken from `draw.emf` which uses **@cg** to obtain cursor movements from the user. Note how the input from **@cg** (stored in variable **%dw-comm**) is compared with the binding name rather than any keyboard characters.

```
!repeat
    0 screen-update
    !force set-variable #l0 @cg
    !if &seq #l0 "abort-command"
        !if &iseq @mc1 "Really quit [y/n]? " "nNyY" "y"
            find-buffer :dw-buf
            0 delete-buffer "*draw*"
            -1 buffer-mode "view"
            !abort
        !endif
    !elif &seq #l0 "newline"
        .
        .
    !elif &seq #l0 "forward-line"
        1 draw-vert
    !elif &seq #l0 "backward-line"
        -1 draw-vert
    !elif &seq #l0 "forward-char"
        1 draw-horz
    !elif &seq #l0 "backward-char"
        -1 draw-horz
    !elif &seq #l0 "osd"
        .osd.draw-help osd
    !elif &set #l1 &sin #l0 "mdeu-="
        !if &les #l1 5
            set-variable :dw-mode &sub #l1 1
            set-variable :dw-modes #l0
            draw-setmode-line
        !elif &sin #l0 "-="
            set-variable :dw-char #l0
            draw-setmode-line
        !endif
    !else
        ml-write "[Invalid command]"
    !endif
!until 0
```

## SEE ALSO

**@wc**(4), **&kbind**(4), **define-macro**(2).

## NAME

@wc, @wl - Extract characters from the current buffer

## SYNOPSIS

**@wl**
**@wc**

## DESCRIPTION

Buffer variables allow text to be taken from the current buffer and placed into a variable. Two types of extraction are provided **@wl** provides a line extraction method, **@wc** provides a character extraction method.
For example, if the current buffer contains the following text:

```
Richmond
Lafayette
<*>Bloomington                (where <*> is the current point)
Indianapolis
Gary
=* me (BE..) == rigel2 == (c:/data/rigel2.txt) ====================
```

The **@wl** variable allows text from the current buffer to be accessed, a command such as:

```
set-variable %line @wl
```

would start at the current point in the current buffer and grab all the text up to the end of that line and pass that back. Then it would advance the point to the beginning of the next line. Thus, after the set-variable command executes, the string "`Bloomington`" is placed in the variable **%line** and the buffer rigel2 now looks like this:

```
Richmond
Lafayette
Bloomington
<*>Indianapolis               (where <*> is the current point)
Gary
=* me (BE..) == rigel2 == (c:/data/rigel2.txt) ====================
```

The buffer command **@wc** gets the current character in the buffer, it does not change the buffer position. It is important to stress that the cursor position is not modified, in general a macro will interrogate the character under the cursor and then affect the buffer (i.e. by moving the cursor, deleting the character etc.) dependent upon the value of the character returned.

Assignment of the variables causes the buffer contents to be modified, deleting the pervious contents and setting the new value. This can be useful in certain situations but should be performed with care.

## EXAMPLE

The **@wc** variable provides the most useful mechanism to modify the current buffer. The following example is a macro called **super-delete** which is bound to `<CTRL-del>`. The macro deletes characters under the cursor in blocks. If a white space character is under the cursor then all characters up until the next non-white space character are deleted. If a non-white space character is under the cursor then all non-white space characters up until the next white space character are deleted, then the white space is deleted. White space in this context is a `SPACE`, `tab` or `CR` character.

```
;
;---    Macro to delete the white space, or if an a word all of the
```

```
;       word until the next word is reached.
;
define-macro super-delete
    !while &not &sin @wc " \t\n"
        forward-delete-char
    !done
    !repeat
        forward-delete-char
    !until &or &seq @wc "" &not &sin @wc " \t\n"
    !return
!emacro

global-bind-key super-delete "C-delete"
```

## SEE ALSO

**define-macro**(2).

Ref: CurrentBufferVariables(4) File: m4fil016.4 Date: 2026/02/21

## NAME

@h0, @h1, @h2, ... @h4 - History Group Variables.

## SYNOPSIS

**@h0** *index* - General history string
**@h1** *index* - Buffer history string
**@h2** *index* - Command history string
**@h3** *index* - File history string
**@h4** *index* - Search history string

## DESCRIPTION

The history group variables **@h***n* *index* return the strings in the history buffer ad a given *index*. There are 5 history buffers, *n*, as follows:

`0` - General history string
`1` - Buffer history string
`2` - Command history string
`3` - File history string
`4` - Search history string

Each history buffer contains 20 slots which contain the history information. The history slot is selected with the *index* argument which is valid in the range `0..19`, index `0` is the most recent history slot in the history group.

The history slots are automatically modified on the entry of a new command, search string etc.

**@h***n* *index* returns the whole of the history string for a given slot.

## DIAGNOSTICS

An error is generated if an attempt is made to access these variables and history buffer identifier *n* or the *index* is out of range.

## EXAMPLE

The last search history may be extracted as follows:

```
set-variable #l1 @h4 0
!force search-backward #l1
!if &not $status
    ; Not found
!else
    ; Found
!endif
```

## NOTES

**@h***n* was first introduced in the January 2004 release.

## SEE ALSO

**read-session**(3), **save-session**(3).

## NAME

@?, @#, @0, @1, @2, @3, ... @p - Macro arguments

## SYNOPSIS

**@?** - Boolean flagging if a numeric argument was supplied
**@#** - The value of the numeric argument
**@0** - The name of the macro
**@1** - The first argument of macro
**@2** - The second argument of macro
**@3** ... @*n*
**@p** - The name of the calling (or parent) macro.

## DESCRIPTION

Macros may be passed arguments, allowing a macro to be used by other macros. The **@?** and **@#** are used to determine the numeric argument given to the command. The @*n* variable (where *n* is an integer) used in the context of a macro allows the macro body to determine it's arguments.
From a macro all commands are called in the following form

```
[num] <macro-name> "arg1" "arg2" ....
```

When executed macros do not have to be given an argument, in this case **@?** will be *0* and **@#** will be *1* (the default argument). If an argument is given then **@?** will be *1* and **@#** will be set to the numeric argument given.
The current macro command name <*macro-name*> can be obtain by using the **@0** variable, e.g.

```
define-macro Test-it
    ml-write @0
!emacro
```

When executed, writes the message "`Test-it`" which is the name of the macro.
Arguments may be passed into macro commands in the same way as standard commands are given arguments. The macro being called can access these by the **@1** to **@n** variables, where *n* is a positive integer. Any variables given as arguments are evaluated so if the variable name is required then enclose it in quotes, e.g.

```
set-variable %test-var "Hello World"

define-macro Test-it
    ml-write &cat &cat &cat &cat @0 " " @1 " = " &ind @1
    set-variable  @1 @2
!emacro

Test-it "%test-var" "Goodbye World"
```

On execution the macro writes the message

"`Test-it %test-var = Hello World`"

and will set variable `%test-var` to "`Goodbye World`". Note that the **@1** has been used by reference on the **set-variable** rather than by value as it is used in an assignment.

The **@p** variable can be used to obtain the name of the macro which is executing the current macro, i.e. the value of the parent's **@0** variable. If the macro was executed directly by the user then there is no parent macro and the value of **@p** is an empty string (””).

The variables **@#** and **@?** are automatically set on entry to a macro, the values may be over-written by the calling macro if required, but their assignment values are restricted to being integer values, assignment of a character string causes an error.

## DIAGNOSTICS

If an attempt is made to access an argument which has not been given then a error occurs. This error can be trapped using the **!force**(4) directive, enabling the macro to take appropriate action, see example.

## EXAMPLE

Consider the implementation of **replace-all-string**(3) macro defined in search.emf:

```
define-macro replace-all-string
    !force set-variable #l0 @3
    !if &not $status
        set-variable #l1 @ml05 "Replace all"
        set-variable #l2 @ml05 &spr "Replace [%s] with" #l1
        set-variable #l0 @ml00 "In files"
    !else
        set-variable #l1 @1
        set-variable #l2 @2
    !endif
    .
    .
    .
!emacro
```

In this example if the 3rd argument is not given then the macro gets all arguments from the user. The **@p** variable having a value of ”” when a macro is called directly by the user can be useful when determining the amount of information to feed-back to the user. For example, executing the clean macro is an easy way to remove surplus white characters, so it is often used by other macros as well as by the user. When called directly **clean** refreshes the display and prints a message of completion, but when called by other macros this would cause an unwanted screen-update and message, so clean only does this when executed by the user. This is done as follows:

```
define-macro clean
    ;
    ; Prepare to clean up file.
    .
    .
    .
    !if &seq @p ""
        screen-update
        ml-write "[Cleaned up buffer]"
    !endif
!emacro
```

## NOTES

The parsing of arguments can be inefficient because of the way the arguments have to be parsed; to get the 4th argument the 1st, 2nd and 3rd arguments must be evaluated. This is because each argument is not guaranteed to be only one element, it could be an expression which needs to be evaluated. Consider the following invocation of our Test-it macro

```
Test-it &cat "%test" "-var" "Goodbye World"
```

The 2nd argument is not *”%test”* as this is part of the first argument, the 2nd argument is in fact the 4th element and the invocation will have the same effect except slower.

## SEE ALSO

MacroNumericArguments, **define-macro**(2), **replace-all-string**(3), **!force**(4).

Ref: MacroArguments(4) File: m4fil014.4 Date: 2026/02/23

## NAME

@#, @? - Macro numeric arguments

## SYNOPSIS

**@#** - The numerical argument to a macro
**@?** - The truth of the numerical argument to a macro

## DESCRIPTION

All built-in commands and macros are invoked with a numerical argument. The argument is obtained from either the command line when the user invokes a command line such as:

**esc 5 esc x forward-char**

where the argument is entered after prefix 1 (**esc**). In this case, causing the cursor to be moved forward 5 characters. Within a macro file the same operation is defined as:

**5 forward-char**

In both cases the numerical argument 5 is passed to the command requesting that the resultant operation is performed 5 times in succession before returning. The command itself is invoked once, it is the responsibility of the command to iterate if requested.

The command determines how the numerical argument is interpreted, in the case of spell-word the argument identifies the type of word that is being spelled and NOT the number of words to spell.

The invocation of named macros operate in the same way, the macro may use the variables **@?** and **@#** to determine the status of the numerical argument passed to it. The variables are interpreted as follows:

**@?** A logical value defined as `TRUE` (1) if a numerical argument has been specified, otherwise `FALSE` (0).

**@#** A signed integer value of the supplied numeric argument. If no argument is supplied (i.e. **@?**==`FALSE`) then **@#** is set to 1.

The **@?** and **@#** are only valid for the current macro invocation. Other macros or commands that are invoked have their own values of **@?** and **@#**.

## EXAMPLE

Consider the following example, which sorts lines into alphabetical order using the **sort-lines**(2) function. A new command **sort-lines-ignore-case** is created using a macro to sort lines case insensitively regardless of the current buffer mode. The command **sort-lines** takes an optional argument which determines which column should be used to perform the sort.

```
;
; sort-lines-ignore-case
; Sort lines case insensitively regardless of the current 'exact' mode
; setting.
define-macro sort-lines-ignore-case
    set-variable #l0 &bmod exact
    -1 buffer-mode "exact"
    !if @?
```

```
        @# sort-lines
    !else
        sort-lines
    !endif
    &cond #l0 1 -1 buffer-mode "exact"
!emacro
```

**@?** is used to test the presence of the argument, if it is false **sort-lines** is invoked without an argument. When true the numeric argument is propagated e.g. **@# sort-lines**.
This particular macro highlights an important consideration when passing the numerical argument to other functions, had the macro been implemented as:

```
; INCORRECT IMPLEMENTATION
define-macro sort-lines-ignore-case
    set-variable #l0 &bmod exact
    -1 buffer-mode "exact"
    @# sort-lines
    &cond #l0 1 -1 buffer-mode "exact"
!emacro
```

then when **sort-lines-ignore-case** is invoked with no arguments **@#** is defined as 1, this is would be incorrectly propagated to **sort-lines** causing it to sort on column 1 rather than column 0 as expected.

## SEE ALSO

MacroArguments, **define-macro**(2).

Ref: MacroNumericArguments(4) File: m4fil019.4 Date: 2022/06/12

## NAME

@mn, @mna, @ml, @mc, @mx, @mxa - Message line input

## SYNOPSIS

**@mn**
**@mna**
**@ml**[*f*][*h*] "*prompt*" ["*default*"] ["*initial*"] ["*com-list*"] ["*buffer-name*"] ["*history-list*"]
**@mc**[*f*] *prompt* [*valid-list*] [*help-string*]
**@mx** "*command-line*"
**@mxa** "*command-line*"

## DESCRIPTION

The **Message Line Variables** provide a method to prompt the user for an input returning the data to the caller. The **@mn** variable cause MicroEmacs to input data from the user in the default way for that command's argument, i.e. the normal prompt with the normal history and completion etc. Similarly **@mna** causes MicroEmacs to input the current argument and any subsequent arguments in the default way.

The **@ml** variable can be used to get a string (or Line) of text from the user using the message-line in a very flexible way. The first optional flag **f** is a single character bitwise flag given in hexadecimal, where each bit has the following meaning

`0x01` The *default* value will be specified (except when flag **h** is `b`, see below) and this will be returned by default.

`0x02` The *initial* value will be specified and this will be initial value given on the input line.

`0x04` Auto-complete using the initial value, usually used with bit `0x02`.

`0x08` Hide the input string, the characters in the current input string are all displayed as `'*'`s.

If no value is specified then default value is 0 and **h** can not be specified. The *default* value is returned when the user enters an empty string. If the *initial* string is specified the input buffer will be initialized to the given string instead of and empty one.

The flag **h** specifies what type of data is to be entered, this specifies the history to be used and the semantics allowed, **h** can have the following values

`0` For a general string input using the general history.
`1` For an absolute file name, with completion and history.
`2` For a MicroEmacs buffer name, with completion and history.
`3` For a MicroEmacs command name, with completion and history.
`4` For a file name, with completion and history.
`5` For a search string, with history.
`6` For a MicroEmacs variable name, with completion and history.
`7` For a MicroEmacs mode name, with completion and history.
`8` For a general string using no history.
`9` For a user supplied completion list (`com-list`).
`a` For a user supplied completion list (`buffer-name`).

`b` For a general string and a user supplied history list (`history-list`).

A default value of 0 is used if no value is specified. At first glance type 1 and 4 appear to be the same. They differ only when a non absolute file name is entered, such as "foobar". Type 1 will turn this into an absolute path, i.e. if the current directory is "`/tmp`" then it will return "`/tmp/foobar`". Type 4 however will return just "foobar", this is particularly useful with the **&find**(4) function to then find the file "foobar".

When a value of `9` is used the argument *com-list* must be given which specifies a list of completion values in the form of a MicroEmacs list (see help on **&lget**(4) for further information on lists). The user may enter another value which is not in the list, which will be returned.

Alternatively a completion list may be given in the form of a buffer using a value of `a`. The argument *buffer-name* must be given to specify the buffer name from which to extract the completion list; each line of the buffer is taken as a completion value. This option is particularly useful for large completion lists as there is no size restrictions.

When a value of `b` is used the argument *history-list* must be given which can specify up to 20 previous values to be used as a history. The list must be given in the form of a MicroEmacs list (see help on **&lget**(4) for further information on lists) and if used with bit `0x01` of the flag **f**, see above, the *default* value must be given as the first item in the given *history-list* rather than via the separate parameter.

The **@mc** variable can be used to get a single character from the user using the message-line. The optional flag **f** is a bitwise flag where each bit has the following meaning

`0x01` The *valid-list* specifies all valid letters.

`0x02` Quote the typed character, this allows keys such as '`C-g`' which is bound to the abort command to be entered.

`0x04` A *help-string* is given which will be displayed if the user presses '`?`'.

The default value for **f** is 0. When **@mc** is used, the user is prompted, with the given prompt, for a single character. If a *valid-list* is specified then only a specified valid character or an error can be returned. For example, a yes/no prompt can be implemented by the following

```
!if &iseq @mc1 "Are you bored (yn) ? " "yYnN" "y"
    save-buffers-exit-emacs
!endif
```

By using the **&isequal**(4) operator a return of "Y" or "y" will match with "y". However, there are two problems with this, firstly if the user aborts by pressing '`C-g`' a macro error will be generated because the **&iseq** operator failed to get 2 arguments. The second problem is that the meaning of "`(yn)`" may not be obvious to all users so a help string should be supplied. To address these 2 issues the following should be used in preference:

```
set-variable #l0 @mc5 "Are you bored (?yn) ? " "yYnN" "(Y)es, (N)no, (C-g)Abort ? "
!if &iseq #l0 "y"
    save-buffers-exit-emacs
!endif
```

When the **@mx** variable is used MicroEmacs sets the system variable **$result**(5) to the input prompt, it will then execute the given `command-line`. If this command aborts then so does the calling command, if it succeeds then the input value is taken from the **$result** variable. Similarly **@mxa** causes MicroEmacs to get the current and any subsequent arguments in this way.

These variables are useful when trying to use existing commands in a different way, such as trying to provide a GUI to an existing command. See the **delete-buffer** example below.

## EXAMPLE

The following example can be used to prompt the user to save any buffer changes, the use of **@mna** ensures the user will be prompted as usual regardless of the number of buffers changed:

```
save-some-buffers @mna
```

The following example sets `%language` to a language supplied by the user from a given list, giving the current setting as a default

```
set-variable %languages "|American|British|French|Spanish|"
set-variable %language "American"

set-variable %language @ml19 "Language" %language %languages
```

The following example is taken from `diff-changes` in tools.emf, it uses **@mc** to prompt the user to save the buffer before continuing:

```
define-macro diff-changes
    !if &seq $buffer-fname ""
        ml-write "[Current buffer has no file name]"
        !abort
    !endif
    !if &bmod "edit"
        set-variable #l0 @mc5 "Save buffer first (?yn) ? " "nNyY" "(Y)es, (N)o, (C-g)Abort ? "
        !if &iseq #l0 "y"
            save-buffer
        !endif
    !endif
        .
        .
```

Note that the input is case insensitive. The following version would not work as the user may expect when the buffer has not been edited:

```
        .
        .
    !if &and &bmod "edit" &iseq @mc1 "Save buffer first (?yn) ? " "nNyY" "y"
        save-buffer
        .
        .
```

Unlike **C** and other similar languages MicroEmacs macro language always evaluates both **&and** arguments. This means that the user will be prompted to save the buffer regardless of whether the buffer has been edited.

The **@mx** variables are useful when using existing commands in a new environment. For example, consider providing a GUI for the **delete-buffer**(2) command, when executed the calling GUI may not be aware that changes could be lost or a process may still be active. These variables can be used as a call back mechanism to handle this problem:

```
define-macro osd-delete-buffer-callback
    !if &sin "Discard changes" $result
        2 osd-xdialog "Delete Buffer" "  Dicard changes?  " 2 10 6 "&Yes" "&No"
        set-variable $result &cond &equ $result 1 "y" "n"
    !elif &sin "Kill active process" $result
        2 osd-xdialog "Delete Buffer" "  Kill active process?  " 2 10 6 "&Yes" "&No"
        set-variable $result &cond &equ $result 1 "y" "n"
    !else
        1000 ml-write &spr "[Unknown prompt %s]" $result
        !abort
    !endif
!emacro
```

```
define-macro osd-delete-buffer
    .
    . set #l0 to buffer name to be deleted
    .
    delete-buffer #l0 @mxa osd-delete-buffer-callback
!emacro
```

## SEE ALSO

**define-macro**(2).

Ref: MessageLineVariables(4) File: m4fil015.4 Date: 2026/02/23

## NAME

@s0, @s1, @s2, ... @s9 - Last search group values

## SYNOPSIS

**@s0** - Last search's whole match string
**@s1** - Last search's first group value
**@s2** - Last search's second group value
...
**@s9** - Last search's nineth group value

## DESCRIPTION

The search group variables **@s***n* return the string matches of the last regular expression search i.e. **search-forward**(2) (in **magic**(2m) mode) or **search-buffer**(2).

**@s0** returns the whole of the matched string, **@s***n*, *n* = 1..9, returns the bracket matches corresponding to the group demarkation points indicated by \( and \) in the search regular expression.

## DIAGNOSTICS

An error is generated if an attempt is made to access these variables and the last search failed or the last search did not have the specified group.

The value returned for an unused group, e.g. @s2 for the regex string "\(a\)\|\(b\)" if "a" was matched, is an empty string ("").

## EXAMPLE

The following macro code gives a simple example of their potential use:

```
forward-search "Token *{\\(Start\\|End\\)}"
!if $status
    ml-write "[found \"%s\"]" @s0
    !if &seq @s1 "Start"
        .
        .
```

## NOTES

Remember that the regular expression escape character '\' has to be duplicated within a macro file as '\' is also the macro file escape sequence.

## SEE ALSO

**magic**(2m), **search-buffer**(2), **search-forward**(2).

Ref: SearchGroups(4) File: m4fil033.4 Date: 2022/06/12

## NAME

Variables - Macro variables

## SYNOPSIS

**#***tn*
**$***variableName*
**%** *variableName*
**.***variableName*
**.***commandName***.***variableName*
**:***variableName*
**:***bufferName***:***variableName*

## DESCRIPTION

Variables are part of MicroEmacs macro language and may be used wherever an argument is required. The variable space comprises:

**#** - Register Variable
**$** - System Variable
**%** - Global Variable
**.** - Command Variable
**:** - Buffer Variable

All variables hold string information, the interpretation of the string (numeric, string or boolean) is determined when the variable is used within the context of the command. There are five types of variable, **Register Variables** (prefixed with a hash '**#**'), **System Variables** (prefixed with a dollar '**$**'), **Global Variables** (prefixed with a percentage '**%**'), **Buffer Variables** (prefixed with a colon '**:**') and **Command Variables** (prefixed with a period '**.**').

### Register Variables

Register Variables provide a set of 10 prefixed global (**#g0** .. **#g9**), parent (**#p0** .. **#p9**) and local (**#l0** .. **#l9**) register variables. The interpreted decode time of the register variables is significantly smaller than other variable types as no name space search is performed.

Register variables are assigned using **set-variable**(2), their value may be queried with **describe-variable**(2), unlike Global Buffer or Command variables they cannot be deleted.

Register variables are implemented like a stack, where the global registers are the top of the stack and every executing macro gets its own set of resister variables (**#l?**). The macro also has access to the global registers (**#g?**) and its calling, or parent macro (**#p?**). If the macro has no parent macro then the global registers are also the parent registers. Outside macros, i.e. using **set-variable** manually, the global parent and local registers are the same.

Register variables are typically used for retaining short term state, computation steps etc. As with the User Variables, the global register variables are global and care must be taken with nested macro invocations to ensure that the register usage does not conflict.

### System Variables

MicroEmacs defines many System variables which are used to configure many aspects of the editors environment. The functionality of each system variable has been documented, they can

be set and described but cannot be unset. If the user attempts to set or describe a non-existent MicroEmacs system variable (e.g. **$PATH**) the system environment is used instead, allowing the user to query and alter the system environment.

### Global, Command and Buffer Variables

The Global variables are denoted by an initial **%** character followed by the name of the variable *variableName*. The *variableName* may be any ASCII character string up to 127 characters in length, all characters of the name are significant. Shorter names are preferred as this speeds up execution. Global Variables exist in a global context which all macros have access to.
Command variables exist within the scope of a command, they are denoted by the period (**.**) character. They can be accessed by one of two forms, either **.***variableName* or **.***commandName***.***variableName*. The first form, without the command name, assumes the scope to be the current command, as such may only be used to access internal variables. The second form qualifies the scope by specifying the command, this form is much more versatile and may be used to access any command variable from any other command, e.g.

```
define-macro foo
    set-variable .foo "Hello world"
    1000 ml-write &cat "foo1: " .foo
    1000 ml-write &cat "foo2: " .foo.foo
!emacro
define-macro bar
    foo
    1000 ml-write &cat "bar1: " .foo
    1000 ml-write &cat "bar2: " .foo.foo
!emacro

bar
```

When **bar** is executed the following messages may be observed:

```
foo1: Hello World
foo2: Hello World
bar1: ERROR
bar2: Hello World
```

When a macro file or buffer is executed, they are executed within their own scope so local scope command variables (form 1) may be created and used in that scope. Any such variables created are automatically deleted at the end of execution. For example, the default color scheme generator macro file, `schemed.emf`, creates command variables for the created colors to aid readability:

```
add-color &set .green     3 0   200 0
a0dd-color &set .lgreen   11 0   255 0

...

add-color-scheme .scheme.cardback   .lgreen   .green   .lgreen ...
```

The variables only exist as a file or buffer is being executed, they are not accessible by another command once the command or buffer execution has finished.
Buffer variables are similar to Command variable in function and behaviour except that their scope is of a buffer and are denoted by the colon (**:**) character. Access can be in one of two forms, either **:***variableName* where the scope is assumed to be the current buffer or **:***bufferName***:***variableName*, where the scope is explicitly given allowing access to any buffer variable, e.g.

```
find-buffer "foo"
set-variable :foo "Hello world"
find-buffer "bar"
set-variable :bar "Hello world"
1000 ml-write &cat ":foo     " :foo
1000 ml-write &cat ":foo:foo " :foo:foo
1000 ml-write &cat ":bar     " :bar
1000 ml-write &cat ":bar:bar " :bar:bar
```

When the above is executed the following messages may be observed:

```
:foo     ERROR
:foo:foo Hello World
:bar     Hello World
:bar:bar Hello World
```

Global, Buffer and Command variables are automatically defined when they are used. A variable is assigned with **set-variable**(2) and may be subsequently deleted with **unset-variable**(2). The current assignment of a variable may be queried from the command line using **describe-variable**(2). e.g.

```
define-macro foo
!emacro
set-variable %foo "Some string"
set-variable :bar "Some string"
set-variable .foo.bar "Some string"

...

ml-write &spr "%s %s %s" %foo :bar .foo.bar

...

unset-variable :bar
unset-variable %foo
unset-variable .foo.bar
```

An undefined variable returns the string `ERROR`, this known state is used to advantage with the hilighting initialization, e.g.

```
!if &sequal .hilight.c "ERROR"
    set-variable .hilight.c &pinc .hilight.next 1
!endif
;
; Hi-light C Mode
;
0 hilight .hilight.c  2 50             $global-scheme
```

In this case the variable **.hilight.c** is explicitly tested for definition, if it is undefined then it is assigned a new value.

Conventionally, names are separated with a minus sign character (**-**) e.g. `foo-bar`. It is strongly advised that the name space is kept reasonably clean, since there are no restrictions on the number of macros that may be defined, problems will arise if different macros use the same variables in different contexts. Where possible, Command or Buffer Variables are preferable to Global Variables since they have no side effects on other macros or buffers. It is advised that all variable names associated with a particular macro set are prefixed with short identifier to make the variable name space unique. e.g. the **Metris** macro prefixes all variables with **:met-**; the **draw** macro uses **:dw-**, the **patience** macro **:pat-** etc.

Macro writers should endeavor to use the minimal number of variables, obviously the more variables that exist in the system, the greater the lookup time to find a variable. Use Register Variables in preference to Command, Global or Buffer variables for intimidate computation steps, temporary state etc.

Note that Buffer Variables are automatically deleted when the buffer is deleted.

## EXAMPLE

The following example is the macro to convert tabs to spaces, it is shown in two forms, with User Variables and with Register Variables, the register variable implementation is obviously preferable since no new variables have been defined.

### User Variable Implementation

```
;
; tabs-to-spaces.
; Convert all of the tabs to spaces.
define-macro tabs-to-spaces
    set-variable %curline $window-line        ; Remember line
    beginning-of-buffer
    !force search-forward "\t"
    !while $status
        3 drop-history
        set-variable %curcol $window-acol
        backward-delete-char
        &sub %curcol $window-acol insert-space
        !force search-forward "\t"
    !done
    3 drop-history
    goto-line %curline
    update-screen
    ml-write "Converted tabs!"
!emacro
```

### Register Variable Implementation

```
;
; tabs-to-spaces.
; Convert all of the tabs to spaces.
define-macro tabs-to-spaces
    ; Remember line
    set-variable #l0 $window-line
    beginning-of-buffer
    !force search-forward "\t"
    !while $status
        set-variable #l1 $window-acol
        backward-delete-char
        &sub #l1 $window-acol insert-space
        !force search-forward "\t"
    !done
    goto-line #l0
    screen-update
    ml-write "[Converted tabs]"
!emacro
```

### SEE ALSO

**@wc**(4), **define-macro**(2), **describe-variable**(2), **set-variable**(2), **unset-variable**(2).

Ref: Variables(4) File: m4fil018.4 Date: 2026/02/23

## NAME

$auto-time - Automatic buffer save time

## SYNOPSIS

**$auto-time** *seconds*; Default is `300` seconds

0 <= *seconds* <= t

## DESCRIPTION

Sets the number of seconds to wait until an edited buffer is auto-saved to temporary file to t seconds. A setting of 0 disables the auto-saving command. Auto-saving can be enabled and disabled on a per buffer basis using buffer mode **autosv**(2m).

The auto-save file naming convention is the same as the backup name only using hash ('`#`') instead of tilde (' ') and is automatically removed on saving a buffer.

On unlimited length file name systems (UNIX), the following file naming conventions are used for file `xxxxx`:

```
xxxxx -> xxxxx#
```

On systems with an `xxxxxxxx.yyy` file name (DOS etc), the following file naming conventions are used:

```
xxxxxxxx      -> xxxxxxxx.###
xxxxxxxx.y    -> xxxxxxxx.y##
xxxxxxxx.yy   -> xxxxxxxx.yy#
xxxxxxxx.yyy  -> xxxxxxxx.yy#
```

## NOTES

The user is warned to be extra careful if files ending in ' ' or '`#`'s are used, it is advisable to disable backup creation (see **global-mode**(2)) and auto-saving (`$auto-time = 0`). The author denies all responsibility (yet again) for any loss of data! Please be careful.

Auto-save files of URL files (i.e. "`ftp://...`" and "`http://...`") are written to the system's temporary directory. This avoids potentially slow auto-saves. This can however lead to recovery problems as the buffer name must be used to avoid auto-saving conflict with other buffers with the same base file name but different paths.

## SEE ALSO

**autosv**(2m), **backup**(2m), **buffer-mode**(2) **find-file**(2), **ftp**(3).

Ref: $auto-time(5) File: m5var000.5 Date: 2022/06/12

## NAME

$box-chars - Characters used to draw lines

## SYNOPSIS

**$box-chars** "*string*"; Default is "|++++++++++-"

## DESCRIPTION

**$box-chars** is a fixed length string that defines the set of characters used to render lines to the screen. **Osd**(2), **directory-tree**(2), **list-registry**(2) and many macros use these characters as a platform independent method of drawing lines. The characters have fixed indices defined as follows:

Index 0 Line joining north to south (vertical line).

Index 1 Line joining south to east.

Index 2 Line joining south to west.

Index 3 Line joining north to east.

Index 4 Line joining north to west.

Index 5 Line joining east to south to west.

Index 6 Line joining north to east to south.

Index 7 Line joining north to east to south to west.

Index 8 Line joining north to south to west.

Index 9 Line joining north to east to west.

Index 10
Line joining east to west.

## EXAMPLE

The **$box-chars** is typically platform dependent, it's setting is determined by the characters available in character set of the hosting platform. MS-DOS and Microsoft Windows environments might use a string such as:

```
"\xB3\xDA\xBF\xC0\xD9\xC2\xC3\xC5\xB4\xC1\xC4"
```

X-Windows environments might use a string such as:

```
"\x19\x0D\x0C\x0E\x0B\x18\x15\x0F\x16\x17\x12"
```

Both utilize platform specific characters.

## SEE ALSO

**Osd**(2), **directory-tree**(2), **list-registry**(2) **$window-chars**(5).

Ref: $box-chars(5) File: m5var022.5 Date: 2025/09/07

## NAME

$buffer-backup - Buffer backup file name

## SYNOPSIS

**$buffer-backup** *FileName*

## DESCRIPTION

**$buffer-backup** is automatically set to the file name the current buffer's file would be backed up to if required. If the current buffer has no file name the variable will be set to "".

The value depends on whether DOS compliant file names are being used (see **$system**(5)), whether multiple backups are being kept (see **$kept-versions**(5)) and the setting of the environment variables **$MEBACKUPPATH** and **$MEBACKUPSUB**. The variable does not take into consideration the current setting of the buffer's **backup**(2m) mode which determine whether a backup will be made.

The environment variable **$MEBACKUPPATH** can be used to change the location of the backup files, it can also be used to prepend the backup filename with a string. **$MEBACKUPPATH** can specify an absolute path (e.g. "`c:/temp/mebackup/`") or a relative path (e.g. "`mebackup/`" which will move all backup files into a sub-directory automatically in the files directory).

The trailing '/' is important as the file name is simple appended, i.e. is creating a backup for "c:/foo/bar.txt" and $MEBACKUPPATH is set the "backup" the backup file name will be "c:/foo/backupbar.txt".

The environment variable **$MEBACKUPSUB** can be used to substitute strings within the backup filename for another. The format of the value is a list of **sed**(1) string substitutions, i.e.

```
$MEBACKUPSUB="s/from1/to1/ s/from2/to2/ s/fr..."
```

The 3 divide characters do not have to be '/'s, they can be any character as long as they are the same, e.g. "sXfrom1Xto1X". When define MicroEmacs performs a simple search for string "`from1`" (i.e. no regex support) and replaces any match with the string "`to1`" etc.

## EXAMPLE

The following example compares the differences between the current version and the bucked up version using the **diff**(3) macro. The **diff-changes** macro is defined in `tools.emf`.

```
define-macro diff-changes
    !if &seq $buffer-fname ""
        ml-write "[Current buffer has no file name]"
        !abort
    !endif
    !if &bmod "edit"
        !if &iseq @mc1 "Save buffer first [y/n]? " "nNyY" "y"
            save-buffer
        !endif
    !endif
    ; get the real file name - this only has effect on unix, copes with symbolic links
    set-variable #l0 &stat "a" $buffer-fname
    ; get the backup name
    set-variable #l1 $buffer-backup
    diff #l1 #l0
!emacro
```

## NOTES

The variable **$buffer-backup** can not be set, any attempt to set it will result in an error.

On Windows and DOS platforms if the $MEBACKUPPATH and $MEBACKUPSUB variables are used all remaining ':' characters are changed to '/'s as these are illegal in the middle of a filename.

## SEE ALSO

**backup**(2m), **$system**(5), **$kept-versions**(5).

Ref: $buffer-backup(5) File: m5var082.5 Date: 2022/06/12

## NAME

$buffer-bname - Name of the current buffer
$buffer-fname - Name of the current buffer's file name

## SYNOPSIS

**$buffer-bname** *BufferName*
**$buffer-fname** *FileName*

## DESCRIPTION

**$buffer-bname** the string name of the current buffer. Buffer names are unrestricted in length, but must be unique. By default the buffer name is derived from the buffer's file name without the path. But this can lead to conflicts, caused by identical file names but different paths. In these situations a counter is appended to the end of the buffer name and is incremented until a unique buffer name is created. For example:

```
File Name               Buffer Name
_____________________________________

/etc/file.c             file.c
/tmp/file.c             file.c<1>
/usr/file.c             file.c<2>
```

**$buffer-fname** contains the name of the current buffer's file name complete with path.

## SEE ALSO

**change-buffer-name**(2).

## NAME

$buffer-fhook - Buffer macro hook command name (buffer creation)
$buffer-dhook - Buffer macro hook command name (buffer deletion)
$buffer-bhook - Buffer macro hook command name (buffer current)
$buffer-ehook - Buffer macro hook command name (buffer swapped)

## SYNOPSIS

**$buffer-fhook** *FunctionName*
**$buffer-dhook** *FunctionName*
**$buffer-bhook** *FunctionName*
**$buffer-ehook** *FunctionName*

## DESCRIPTION

Defines the buffer create, delete, begin and end hook command which are executed:

**$buffer-fhook**
: When the buffer is created.

**$buffer-dhook**
: When the buffer is deleted.

**$buffer-bhook**
: When the buffer becomes the current buffer.

**$buffer-ehook**
: When the buffer is swapped out from being the current buffer.

The variable **$buffer-fhook** cannot be directly set and is largely redundant as the file hook is executed only when the buffer's **major-mode**(2) is first assigned. Its main use is within macros which wish to ascertain what type of buffer it is executing on, i.e. if a command was to be executed only on C files then the follow ensures that this is the case:

```
!iif &not &seq $buffer-fhook "fhook-c"  !abort
```

Where the command *fhook-c* is the C major-mode file hook, note that the **$buffer-major-mode**(5) can often be used for the same purposes.

**dhooks** are executed when a buffer is deleted, but before the contents of the buffer are lost. Note that dhooks will not be called if the buffer never becomes active, or if MicroEmacs quits due to the receipt of a panic signal.

**bhooks** and **ehooks** are usually used to set and restore global variables which require different setting in the current buffer.

These variables are initially set by the system when the buffer first becomes active, when the buffer's **major-mode**(2) is determined. After this point the **dhook**, **bhook** & **ehook** variables can be set to any function the user wishes, however the **$buffer-fhook** variable can only be changed by calls to **major-mode** as the value of the variable must stay in sync with **$buffer-major-mode**(5).

## SEE ALSO

**major-mode**(2), **buffer-init**(3), **$buffer-major-mode**(5), Language Templates

## NAME

$buffer-fmod - Buffer file modes (or attributes)
$global-fmod - Global file modes (or attributes)

## SYNOPSIS

**$buffer-fmod** *FileMode*
**$global-fmod** *FileMode*

## DESCRIPTION

**$buffer-fmod** is bit based variable setting the buffers file system modes or attributes. If the buffer was loaded from an existing file then the value of **$buffer-fmod** is taken directly from the file. But if the buffer was created then the buffer inherits the default file modes, **$global-fmod**, which is determined from the users umask on UNIX or a default on others.

Note that the retrieved value of **$buffer-fmod** can be different to the set value, see the **NOTES** section below.

The definition of the file mode bits are platform specific and are considered independently, as follows:

## UNIX

The file modes of Unix are the standard read, write and execute permissions for user, group and global. See **chmod**(1) for a full description of their use and effect.

The variable is displayed in octal.

### Microsoft Windows and DOS

On Microsoft platforms each file attribute (see **attrib**(1)) is assigned a bit, on windows 95 and NT the new file attributes such as compressed are also represented. The bits are assigned as follows

```
Bit     Attrib Flag     Attribute
0x001       R           Read Only
0x002       H           Hidden
0x004       S           System
0x010                   Directory
0x020       A           Archive
0x080                   Normal
0x100                   Temporary
0x800                   Compressed
```

## EXAMPLE

The following example changes the $buffer-fmod so that the file will be executable (UNIX only), useful when writing a shell script.

```
set-variable $buffer-fmod 0775
```

## NOTES

When retrieved, the buffer's file flags are also merged into the value of **$buffer-fmod**. These flags are not platform specific and used to store the current properties of the buffer's file. They cannot be set and are defined as follows:

**0x10000**

When set the buffer is a directory listing, created by the **find-file**(2) command when the file name given is a directory. Earlier versions of MicroEmacs had a **dir** buffer mode, this mode has been removed as this property cannot be simply toggled like other modes. Checking the value of this bit should be used instead.

**0x20000**

When set the file was found to contain binary data that cannot be safely represented in a buffer, as a result saving the buffer will result in data loss and corruption. Note that this is different to **binary**(2m) or **rbin**(2m) buffer modes which can safely represent any binary data, i.e. a file loaded with binary mode enabled will never have this bit set.

**0x40000**

When set the file was found to contain inconsistent line terminations, i.e. typically some lines terminated with '\n' and others with '\r\n'. Saving the file will unify/correct the line terminations, but this could be a corruption.

## SEE ALSO

**find-file**(2), **cr**(2m), **lf**(2m), **binary**(2m), **ctrlz**(2m), **auto**(2m).

Ref: $buffer-fmod(5) File: m5var070.5 Date: 2026/02/17

## NAME

$buffer-hilight - Define current buffer hilighting scheme.

## SYNOPSIS

**$buffer-hilight** *hilightNum*; Default is 0

0 <= *hilightNum* <= 255

## DESCRIPTION

**$buffer-hilight** Sets the current buffer's hi-lighting scheme (see **hilight**(2) for a full description of hi-lighting). The default setting is 0 which specifies no hi-lighting, when set to a non-zero, the hi-light scheme of that number MUST already be defined.

Terminals that cannot display color directly may still be able to take benefit from hi-lighting. A terminal that has fonts can use them in the same way using the **add-color-scheme**(2) command. The hi-light scheme is also used in printing (see **print-buffer**(2)). If, however, your terminal cannot display color in any way, it is recommended that hi-lighting is disabled (except when printing) as it does take CPU time.

## SEE ALSO

**hilight**(2), **print-buffer**(2), **$buffer-scheme**(5), **$buffer-indent**(5).

Ref: $buffer-hilight(5) File: m5var004.5 Date: 2022/06/12

## NAME

$buffer-indent - Current buffer indentation scheme.

## SYNOPSIS

**$buffer-indent** *indentNum*; Default is `0`

0 <= *indentNum* <= 255

## DESCRIPTION

**$buffer-indent** sets the current buffers indentation scheme. *indentNum* is the identity of the indentation scheme, as defined by **indent**(2), which is typically the same value as the buffers hilighting scheme number (see **$buffer-hilight**(5)).

The default setting is 0 which specifies no indentation scheme is present. When non-zero, the value identifies the indentation scheme.

A buffer assigned an indentation method, MicroEmacs performs automatic line re-styling, by moving the left indentation, according to the defined indentation method. The `tab` key is typically disabled. This behavior can be altered using bit `0x1000` of the **$system**(5) variable, which can be changed using **user-setup**(3).

The use of tab characters to create the required indentation is determined by the setting of the buffers **tab**(2m) mode. If the mode is disabled tab characters are used wherever possible, otherwise spaces are always used.

## NOTES

The commands **restyle-region**(3) and **restyle-buffer**(3) use the indentation method when defined.

The buffer indentation scheme is typically assigned in the *fhook* macro, see Language Templates.

## EXAMPLE

The following example sets up an indentation scheme for a buffer within the *fhook* macro.

```
!if &sequal .hilight.foo "ERROR"
    set-variable .hilight.foo &pinc .hilight.next 1
!endif

....

; Define the indentation scheme
0 indent  .hilight.foo 2 10
indent .hilight.foo n "then" 4
indent .hilight.foo s "else" -4
indent .hilight.foo o "endif" -4

....

; File hook - called when new file is loaded.
define-macro fhook-foo
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "foo"
    !endif
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
```

```
    ; Assign the buffer indentation
    set-variable $buffer-indent .hilight.foo
    ; Set the abbreviation file
    buffer-abbrev-file "foo"
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
```

This provides an indentation of the form:

```
if condition
then
    XXXX
else
    if condition
    then
        XXXX
    endif
endif
```

## SEE ALSO

**indent**(2), **tab**(2m), **$system**(5), **user-setup**(3), **restyle-buffer**(3), **restyle-region**(3), **$buffer-hilight**(5).

## NAME

$buffer-input - Divert buffer input through macro.

## SYNOPSIS

**$buffer-input** *commandName*

## DESCRIPTION

**$buffer-input** allows the buffer input mechanism to be diverted through a command macro defined by *commandName*. If this variable is set to a valid command, which may be a user defined macro, this command will be called instead. The command can access the actual key-code typed by the user via the command variable **@cc**(4), e.g. the following macro prints out the name of the command that the user presses until the **abort-command**(2) is executed.

```
define-macro test-input
    ml-write &spr "Current command: %s" @cc
    !if &seq @cc "abort-command"
        set-variable $buffer-input ""
    !endif
!emacro

set-variable $buffer-input test-input
```

## WARNING

Caution is advised when using this, if there is no way of reseting the variable then **MicroEmacs** must be killed.

## SEE ALSO

**abort-command**(2), **@cc**(4).

Ref: $buffer-input(5) File: m5var005.5 Date: 2025/08/28

## NAME

$buffer-ipipe - Divert buffer incremental pipe input through macro.
$buffer-ipipe-flags - Incremental pipe creation flags.

## SYNOPSIS

**$buffer-ipipe** *commandName*

**$buffer-ipipe-flags** *flags* ; read-only

## DESCRIPTION

**$buffer-ipipe** allows the buffer incremental pipe input mechanism to be diverted through a command macro defined by *commandName*. On a buffer running an **ipipe-shell-command**(2) the command, set by this variable, will be called whenever new text has been inserted by the executing process. Two *alpha-marks* will be set in the buffer, '`i`' denotes the start of the newly inserted text and '`I`' denotes the end.

When the ipiped command exits the *commandName* is called with a numeric argument of 0 and the **$result**(5) variable is set to the exit code of the process, this will be the last call of the command. The following values of **$result** are also used by MicroEmacs:

**-1000** The ipiped process did not exit normally, cause unknown.

**-1009** The ipiped process was killed.

**-1011** The ipiped process core dumped.

**-1015** The ipiped process was terminated.

Not all platforms support all of these exit codes.

**$buffer-ipipe-flags** is a read-only variable which records how the incremental pipe running, or last run, in the current buffer was created by **ipipe-shell-command**(2). It allows macros to determine how the process was launched, and how its output was processed, without having to retain this information themselves. *flags* is a bit based flag where:

**0x01** The buffer is, or was, running an ipiped process. This bit remains set after the process has exited, mode **pipe**(2m) may be used to determine whether the process is still running. If this bit is clear the buffer has never run an ipiped process and all other bits are 0.

**0x02** The process was launched with a pseudo terminal, i.e. a UNIX PTY or a Windows ConPTY, as opposed to a plain anonymous pipe. A pseudo terminal may be requested but not granted, this bit reflects what was created and not what was asked for, see **ipipe-shell-command** bits **0x08** and **0x2000**.

**0x04** The pipe was created as a raw pipe, see **ipipe-shell-command** bit **0x40**. As no text annotation is added in this mode, the buffer does not contain the leading "`cd` *path*" and command lines, so the command-line which was executed cannot be recovered from the buffer.

**0x08** **wrap**(2m) mode was enabled when the pipe was created, i.e. neither bit **0x200** nor bit **0x40**, raw mode which also forces **wrap** mode off, was given to **ipipe-shell-command**. The buffer's **wrap** mode may subsequently be changed, by the user or a macro, this bit always records the state at creation.

**0x10** The pipe output is being decoded from UTF-8, see **ipipe-shell-command** bit **0x800**. Note that this bit is clear when the output is not decoded for any reason, such as the terminal not operating in UTF-8 mode, and not simply when decoding was explicitly disabled.

**0x20** ANSI color processing was enabled, see **ipipe-shell-command** bit **0x1000**.

The remaining bits are reserved for future use and are currently always 0.

## EXAMPLE

The following example, taken from the **ipipe-rerun** macro, uses **$buffer-ipipe-flags** to construct the numeric argument required to re-launch the command in the current buffer with the same settings:

```
; A raw pipe leaves no command-line in the buffer to re-run
!iif &band 0x04 $buffer-ipipe-flags \
    -8 ml-write "[ipipe-rerun Error: command-line lost]"
; Convert the flags back into an ipipe-shell-command argument
set-variable #l0 &con &band 0x02 $buffer-ipipe-flags 0x08 0x2000
set-variable #l0 &add #l0 &con &band 0x08 $buffer-ipipe-flags 0 0x200
set-variable #l0 &add #l0 &con &band 0x20 $buffer-ipipe-flags 0x1000 0
```

Note that bits **0x02** and **0x08** are inverted in the construction as **ipipe-shell-command** bits **0x2000** and **0x200** disable the features which **$buffer-ipipe-flags** reports as enabled.

## SEE ALSO

**ipipe-shell-command**(2), **goto-alpha-mark**(2), **$result**(5), **$buffer-input**(5), **pipe**(2m), **wrap**(2m).

## NAME

$uffer-major-mode - Define buffer major-modem

## SYNOPSIS

**$uffer-major-mode** *MajorModeId*

## DESCRIPTION

The $uffer-major-mode variable defines the current buffer's active major mode. The variable cannot be directly set and is largely redundant as it must be configured by calling **major-mode**(2). Its main use is within macros which wish to ascertain what type of buffer it is executing on, i.e. if a command was to be executed only on C files then the follow ensures that this is the case:

```
!iif &not &seq $buffer-major-mode "c"  -8 ml-write "Error: Not a C file!"
```

The variable and buffer's current major mode can be configured by calling **major-mode** or **buffer-setup**(3). When a buffer first becomes active, the system uses the buffer's content and file name to determine the initial major-mode to apply, this in turn sets the initial value of this variable. See help on **major-mode**(2) for more information on how this auto-assignment process works and is configured.

## NOTES

The value of this variable must stay in sync with the buffer's major-mode and the variable **$buffer-fhook**(5).

## SEE ALSO

**major-mode**(2), **buffer-setup**(3), **$buffer-fhook**(5), **buffer-init**(3), Language Templates

Ref: $buffer-major-mode(5) File: m5var097.5 Date: 2026/01/18

## NAME

$buffer-mask - Current buffer word class mask.

## SYNOPSIS

**$buffer-mask string**; Default is `luh`

## DESCRIPTION

**$buffer-mask** sets the current buffer word class mask. MicroEmacs has an internal word lookup table which defines whether a given letter is considered to be part of a word. This functionality is used in many areas such as **forward-word**(2), **forward-kill-word**(2) hilighting etc. The mask is composed with any combination of the following flags, the order in which the flags are specified is not important:

**l** All lower case letters.

**u** All upper case letters.

**h** All hexadecimal characters (used to include numerical digits).

**s** Spell extended characters, typically set to accent (`'`), hyphen (`-`) and period (`.`).

**1** User set **1**, usually set to just underscore (`_`) for many system and programming files such as 'C'.

**2** User set **2**, usually set to '`-`', '`$`', '`&`', '`#`', '`!`', '`%`', '`:`' and '`@`' for MicroEmacs files.

**3** User set **3**, not usually defined.

**4** User set **4**, not usually defined.

The character sets may be modified using the **set-char-mask**(2) command.

## SEE ALSO

**set-char-mask**(2), **forward-word**(2).

Ref: $buffer-mask(5) File: m5var007.5 Date: 2025/08/28

## NAME

$buffer-mode-line - Buffer mode line string

## SYNOPSIS

**$buffer-mode-line** "*string*"

## DESCRIPTION

Sets the buffer mode line, unique to this buffer, see **$mode-line**(5) use, description and syntax. If this variable is NOT set for a buffer and **$mode-line** is changed, then the buffer's mode line will also change to the new value. If this variable is set, then then buffer's mode line will be unaffected by any setting of **$mode-line**.

## SEE ALSO

**$mode-line**(5).

Ref: $buffer-mode-line(5) File: m5var008.5 Date: 2025/03/22

## NAME

$buffer-names - Filtered buffer name list

## SYNOPSIS

**$buffer-names** *BufferName*

## DESCRIPTION

**$buffer-names** must first be set to the required filter string, if the variable is evaluated before it is initialized the value will be set to "*ABORT*" and the command will fail. The filter takes the form of a regex.

Once initialized, evaluating **$buffer-names** returns the name of the next buffer which matches the filter until no more buffers are found, in which case an empty string is returned.

## EXAMPLE

The following example prints out the name of all buffers to the massage line one at a time. Note that **&set**(4) is used on the **!while**(4) statement to avoid evaluating **$buffer-names** twice per loop.

```
set-variable $buffer-names ".*"
!while &not &seq &set #l0 $buffer-names ""
    100 ml-write &cat "buffer: " #l0
!done
```

The following example is the same except it lists only the buffers which are not directory listings

```
set-variable $buffer-names ".*[^/]"
!while &not &seq &set #l0 $buffer-names ""
    100 ml-write &cat "buffer: " #l0
!done
```

## NOTES

The list of buffers is evaluated when the variable is initialized, buffers created after the initialization will not be included in the list.

Deleting buffers which are in the list, before they are evaluated, will have undefined effects.

## SEE ALSO

**list-buffers**(2), **$buffer-bname**(5), **$file-names**(5), **$command-names**(5), **$major-mode-ids**(5), **$mode-names**(5), Regular Expressions.

Ref: $buffer-names(5) File: m5var061.5 Date: 2026/01/30

## NAME

$buffer-scheme - Buffer color scheme.

## SYNOPSIS

**$buffer-scheme** *schemeNum*; Default is 0

## DESCRIPTION

**$buffer-scheme** sets the current buffer's color scheme to *schemeNum*, where *schemeNum* is a color scheme defined with **add-color-scheme**(2), which identifies the foreground and background color schemes of the buffer. The color scheme is initialized to the global color scheme settings (see **$global-scheme**(5)) when the buffer is created.

## SEE ALSO

**$buffer-hilight**(5), **$cursor-color**(5), **$trunc-scheme**(5), **$global-scheme**(5), **$ml-scheme**(5), **$mode-line-scheme**(5), **$scroll-bar-scheme**(5), **$system**(5).

Ref: $buffer-scheme(5) File: m5var001.5 Date: 2022/06/12

## NAME

$buffer-tab-width - Buffer tab character interval

## SYNOPSIS

**$buffer-tab-width** `integer`; Default is **$tab-width**(5).

-0 < *integer* <= *n*

## DESCRIPTION

**$buffer-tab-width** defines the width of a tab character in terms of displayable spaces, a tab character is typically 8 spaces. A tab character appearing in a buffer is rendered with up to **$buffer-tab-width** spaces to take it to the next column boundary of modulo **$buffer-tab-width**.

Modification of the variable value causes the displayed tab width to change to the new value. The macro **tabs-to-spaces**(3) may be used to convert all tabs to spaces.

When a buffer is created (file loading or creation) then the default value is taken from the **$tab-width**(5) variable.

## NOTES

The indentation of a buffer is controlled by the **$buffer-indent-width**(5) and not the **$buffer-tab-width** when automatic **indent**(2) is enabled.

The November 2004 version of MicroEmacs removed the existing variables **$tabwidth** and **$tabsize** which were ambiguous. The existing variable scope was global which caused problems when moving to buffers with different tab width properties. The variables have been replaced with **$tab-width**(5) and **$buffer-tab-width**. The indentation which was previously controlled by the tab width has been replaced with **$indent-width**(5) and **$buffer-indent-width**(5).

## SEE ALSO

**$buffer-indent-width**(5), **$indent-width**(5), **$tab-width**(5), **buffer-mode**(2), **indent**(2), **tab**(2m), **tabs-to-spaces**(3).

Ref: $buffer-tab-width(5) File: m5var043.5 Date: 2022/06/12

## NAME

$buffer-xlate - Buffer on-save translation configuration

## SYNOPSIS

**$buffer-xlate**

## DESCRIPTION

**$buffer-xlate** defines the translation needed to be performed before the buffer is written to file, this variable is only ever used when the **xlate**(2m) buffer mode is enabled. The variable is a bitwise flag where each bit has the following meaning:

**0x003** The lower 2 bits define the base translation, where a value of `0` indicates a lossless text file, `1` a UTF-8 encoded Uncode file, `2` a UTF-16 and `3` a UTF-32 encoded Unicode file.

**0x004** When this bit is set and converting to UTF-16 or UTF-32 then the file is written in big-endian byte order, if clear the file is written in little-endian byte order.

**0x008** When set a Unicode 'BOM' character (`U+FEFF`)is written out first.

## NOTES

The value of this variable is set by the **translate-buffer**(2) command such that on saving the same format will be written out. As a result this value can be passed in to **translate-buffer** if the file is re-read.

The value of **$buffer-xlate** for a Unicode file can be safely changed to different formats, i.e. UTF-16 LE to UTF-32 BE etc. However, changing between lossless text and Unicode formats can have unexpected results due to additional Unicode character sequences and line termination styles.

## SEE ALSO

**translate-buffer**(2), **xlate**(2m), **save-buffer**(2), **reread-file**(3), **find-hfile**(3), **find-ufile**(3).

Ref: $buffer-xlate(5) File: m5var017.5 Date: 2025/03/25

## NAME

$command-names - Filtered command name list

## SYNOPSIS

**$command-names** *CommandName*

## DESCRIPTION

**$command-names** must first be initialized to the required filter string, if the variable is evaluated before it is initialized the value will be set to "*ABORT*" and the command will fail. The filter takes the form of a regex.

Once initialized, evaluating **$command-names** returns the name of the next command which matches the filter until no more commands are found, in which case an empty string is returned.

## EXAMPLE

The following example prints out the name of all commands to the massage line one at a time. Note that **&set**(4) is used on the **!while**(4) statement to avoid evaluating **$command-names** twice per loop.

```
set-variable $command-names ".*"
!while &not &seq &set #l0 $command-names ""
    100 ml-write &cat "command: " #l0
!done
```

The following example is an alternative implementation of **command-apropos**(2).

```
define-macro alt-commad-apropos
    set-variable #l1 @ml "Apropos string"
    set-variable $command-names &cat &cat ".*" #l1 ".*"
    !force 0 delete-buffer "*commands*"
    1 popup-window "*commands*"
    !while &not &seq &set #l0 $command-names ""
        insert-string &spr "    %s\n" #l0
    !done
    beginning-of-buffer
    -1 buffer-mode "edit"
    1 buffer-mode "view"
!emacro
```

## NOTES

**$command-names** does not differentiate between built in commands and macros.

The list of commands is evaluated when the variable is initialized, macros created after the initialization will not be included in the list.

## SEE ALSO

**list-commands**(2), **command-apropos**(2), **$buffer-names**(5), **$file-names**(5), **$mode-names**(5), **$variable-names**(5), Regular Expressions.

## NAME

$cursor-blink - Cursor blink rate $cursor-color - Cursor foreground color

## SYNOPSIS

*$cursor-blink integer*; Default is `0`

*$cursor-color colorNum*; Default is `0`

0 <= *colorNum* <= *n*

## DESCRIPTION

**$cursor-blink** sets the cursor's flash rate, i.e. the period in which the cursor is drawn, hidden and then redrawn. The default setting of 0 disables cursor blinking. When set to a none zero value the variable is split into two componants, the first 16 bits, or lower short, sets the cursor visible time in milliseconds, and the higher short sets the hidden time. If the hidden time is set to 0 then the cursor will be hidden for the same length of time it is visible.

The cursor blink rate can be setup in the platform section of **user-setup**(3).

**$cursor-color** sets the cursor's fore-ground color, and can greatly improve cursor visibility. *colorNum* is a integer palette number created using **add-color**(2), the default is 0. The first 8 bits, lowest byte, of **$cursor-color** is used to define the normal cursor colour, however the next 8 bits, bits `9` to `16`, are used to define the cursor colour to use when the current buffer has **over**(2m) mode enabled; if the value is 0 (the default) then the normal cursor colour is used, otherwise it is the colour palette index.

## PLATFORM

UNIX termcap interface does not support **$cursor-color**.

## EXAMPLE

The following example sets the cursor visible time to 600 ms (0x258) and a hidden time to 200 ms (0xc8):

```
set-variable $cusror-blink 0x00c80258
```

## SEE ALSO

**user-setup**(3), **add-color**(2), **$global-scheme**(5), **$ml-scheme**(5), **$mode-line-scheme**(5), **$system**(5).

Ref: $cursor-blink(5) File: m5var010.5 Date: 2025/09/05

## NAME

$cursor-x - Cursor X (horizontal) position
$cursor-y - Cursor Y (vertical) position

## SYNOPSIS

**$cursor-x** *integer*

0 <= *integer* <= $frame-width - 1

**$cursor-y** *integer*

0 <= *integer* <= $frame-depth - 1

## DESCRIPTION

**$cursor-x** and **$cursor-y** are automatically set to the position of the cursor at the last screen update (i.e. the variables are not updated between screen updates). The top left character of the screen is coordinate 0,0 bottom right is $frame-width, $frame-depth.

## NOTES

These variables can not be set. Any attempt to set them will result in an error.

## SEE ALSO

**$mouse-x**(5), **$frame-depth**(5), **$frame-width**(5).

Ref: $cursor-x(5) File: m5var078.5 Date: 2022/06/12

## NAME

$debug - Macro debugging flag

## SYNOPSIS

**$debug** *bitmask*; Default is `0`

## DESCRIPTION

**$debug** is used to control macro debugging, it is a bit based flag where:

`0x01` Displays an error message when an un-handled exception is encountered, this is useful to locate problems in the macro code i.e. missing arguments etc. This setting may be used in normal running and macro execution is halted when an error is encountered, it is non-intrusive up until the point of failure. Enabling slightly slows execution but is not typically noticeable on modern processors.

`0x02` Setting this bit enables the macro debugger. With macro debugging enabled, before a line of macro code is executed the line and its location is first print on the message line with a prompt allowing the user to view variables, abort execution, dive into sub macros etc. The behaviour of the macro debugger can be altered by bits `0x04` and `0x08`.

`0x04` When set all macro lines encountered are printed. When this bit is not set only the lines of code that are executed are printed, i.e. if a false **!if** was encountered the lines within the **!if** would not be printed. Problems arise with **!elif** and **!else** lines and this bit is not set as the **!elif** and **!else** lines are never printed.

`0x08` When set the debug prompt is disabled so the macro line is printed on the message line and then immediately executed. This allows the user to see the execution flow of the macro code while still executing the code at a reasonable pace.

`0x10` Setting this bit has no immediate effect. However as soon as the key bound to **abort-command**(2) (typically "`C-g`") is pressed the macro debugger is enabled, i.e. bit `0x02` is set. This can be invaluable when tracing problems, for example the following macro code will loop infinitely:

```
!repeat
    beginning-of-line
    backward-char
    !force forward-line
!until &not $status
```

This is a fairly obvious bug, but if buried in a thousand lines of macro code it could be very difficult to spot and to find it during execution would be very tedious if not impossible. But by setting **$debug** to `0x10` the macro can be executed as normal and as soon as the macro is stuck the user can simply press "`C-g`" to start macro debugging at the current execution point.

Note that the "`C-g`" which triggers the macro debugger is discarded so it does not cause the current command to abort.

`0x20` Setting this bit is similar to `0x10` except the macro debugger is enabled by the ringing of the bell. When some commands fail they always print an error message and ring the

bell as the error needs handling (e.g. missing argument in a **!if** statement). However if the macro has been called with a **!force** the execution will not be aborted so tracing this issue could be difficult without this setting.

## SEE ALSO

**execute-file**(2).

Writing MicroEmacs Macros
Macro Glossary
Macro Debugging - A Crash Course
MicroEmacs Macro Language at a Glance

## NAME

$delay-time - Mouse time event delay time
$repeat-time - Mouse time event repeat time

## SYNOPSIS

**$delay-time** *milliseconds*; Default is `500`
**$repeat-time** *milliseconds*; Default is `25`
10 <= *milliseconds* <= t

## DESCRIPTION

**$delay-time** sets the time waited between the user picking a mouse button and the generation of a `mouse-time-?` key event.

When user presses the left button (say) a `mouse-pick-1` key event is generated, If this key is bound then the command it is bound to is executed. If the user then holds down the button for **$delay-time** or more milliseconds then MicroEmacs checks the binding of the special `mouse-time-1` key, if this pseudo key is bound then the command it is bound to will be executed.

If the user continues to hold down the button for a further **$repeat-time** milliseconds another **mouse-time-1** key event will be generated. A **mouse-time-1** key event will be generated after every **$repeat-time** milliseconds until the user releases the button, at which point a `mouse-drop-1` key event is generated.

## EXAMPLE

The following example implements the vertical scroll-bar up and down scrolling arrows for a buffer window:

```
define-macro mouse-pick-command
    set-cursor-to-mouse
    !if &equ &band $mouse-pos 15 5
        ml-write "Mouse on up-arrow"
        1 scroll-up
        1 global-bind-key scroll-up "mouse-time-1"
    !elif &equ &band $mouse-pos 15 9
        ml-write "Mouse on down-arrow"
        1 scroll-down
        1 global-bind-key scroll-down "mouse-time-1"
    !endif
!emacro

define-macro mouse-drop-command
    !force global-unbind-key "mouse-time-1"
!emacro

global-bind-key mouse-pick-command "mouse-pick-1"
global-bind-key mouse-drop-command "mouse-drop-1"
```

## SEE ALSO

**$idle-time**(5), **set-cursor-to-mouse**(2), **$mouse-pos**(5).

Ref: $delay-time(5) File: m5var071.5 Date: 2026/02/17

## NAME

$file-ignore - File extensions to ignore

## SYNOPSIS

**$file-ignore** "*string*"; Default is ""

## DESCRIPTION

**$file-ignore** specifies a space separated list of file endings which the file completion is to ignore. This is used by any function which prompts the user for a file name, such as **find-file**(2). A file ending in this case is NOT the extension but the last *n* characters where *n* is the number of characters in the specified ignore file.

The variable can be used to ignore directories by making the last character of the postfix a '/' directory character, see examples below. The variable can also affect the directories listed by the **directory-tree**(2) command.

## EXAMPLE

To ignore all files which have the extension "o", using:

```
set-variable $file-ignore "o"
```

would not only ignore "foo.o", but also "foo.oo", "foo.po" and "foo" as well as any file that ends in an "o". What is really required is

```
set-variable $file-ignore ".o"
```

It is useful to ignore the "./" and "../" directories so that a directory containing one file will auto-complete to that one file. This is achieved by using:

```
set-variable $file-ignore "./"
```

To ignore MicroEmacs backup files (" "), C object files (".o"), "./" and "../" directories try using:

```
set-variable $file-ignore "~ .o ./"
```

## NOTES

The file completion only completes further than the first non-unique point in the current list of possibles if and only if it can ignore all but one file, so if the current directory contains:

```
./ ../ foo foo.c foo.c~ foo.o
```

using the above ignore list, completing with "" has no effect as "foo" and "foo.c" cannot be ignored; completing with "foo." will however complete to "foo.c".

## SEE ALSO

**find-file**(2), **directory-tree**(2).

Ref: $file-ignore(5) File: m5var058.5 Date: 2023/06/29

## NAME

$file-names - Filtered file name list

## SYNOPSIS

**$file-names** *FileName*

## DESCRIPTION

**$file-names** must first be initialized to the required filter string, if the variable is evaluated before it is initialized the value will be set to "*ABORT*" and the command will fail.

The filter takes the form of a regex. The filter string should also contain the path to the required directory, the path many not contain wild-cards. If no path is specified the the path of the current buffer's file name is taken, if the current buffer has no file name then the current working directory is used.

On initialization, **$result**(5) is set to the absolute path of the directory being evaluated.

Once initialized, evaluating **$file-names** returns the name of the next file or directory which matches the filter until no more files or directories are found, in which case an empty string is returned.

## EXAMPLE

The following example creates a list of all files in the current directory to a fixed buffer "**files**". Note that **&set**(4) is used on the **!while**(4) statement to avoid evaluating **$file-names** twice per loop.

```
set-variable $file-names ".*"
!force 0 delete-buffer "*files*"
1 popup-window "*files*"
insert-string &spr "Directory listing of %s\n\n" $result
!while &not &seq &set #l0 $file-names ""
    insert-string &spr "    %s\n" #l0
!done
beginning-of-buffer
-1 buffer-mode "edit"
1 buffer-mode "view"
```

## NOTES

Unlike MS-DOS and Windows systems, to match every file a filter of just "*" is required. A filter of "*.*" only matches file names with a '.' in them.

The list of files is evaluated when the variable is initialized, files created after the initialization will not be included in the list.

## SEE ALSO

**$result**(5), **find-file**(2), **$buffer-fname**(5), **$buffer-names**(5), **$command-names**(5), **$mode-names**(5), Regular Expressions.

Ref: $file-names(5) File: m5var064.5 Date: 2026/01/22

## NAME

$file-size-prompt - Large file size warning prompt

## SYNOPSIS

**$file-size-prompt** "*integer*"; Default is "`1024`"

## DESCRIPTION

Sets the size, in megabytes, to trigger a warning prompt before loading a file. Any attempt to load a file this size or larger will result in a "`File <name> is #MB, continue ?`", entering '**N**' or aborting will stop the file loading processes resulting in an empty buffer.

The default value is 1024MB, i.e. 1 gigabyte.

## SEE ALSO

**find-file**(2).

Ref: $file-size-prompt(5) File: m5var016.5 Date: 2024/05/21

## NAME

$file-template - Regular expression file search string

## SYNOPSIS

**$file-template** "*string*"; Default is ""

## DESCRIPTION

**$file-template** defines a regular expression search string used to identify a file in the **grep**(3) and **compile**(3) buffers. The format of the string is the same as magic mode search strings (see **search-forward**(2)).

## EXAMPLE

A UNIX file name may be considered to contain any ASCII character except a space or a '**:**' (used as a divider in many programs). Thus **$file-template** should be:

```
set-variable $file-template "[!-9;-z]+"
```

This will correctly identify "`foo.c`" in the following example.

```
foo.c: 45:      printf("hello world\n") ;
```

## SEE ALSO

**$line-template**(5), **compile**(3), **get-next-line**(2), **grep**(3), **search-forward**(2).

Ref: $file-template(5) File: m5var020.5 Date: 2022/06/12

## NAME

$fill-bullet - Paragraph filling bullet character set
$fill-bullet-len - Paragraph filling bullet search depth

## SYNOPSIS

**$fill-bullet** "*string*"; Default is "`*)].-`"
**$fill-bullet-len** *length*; Default is `5`
0 <= *length* <= $buffer-fill-col

## DESCRIPTION

**$fill-bullet** contains the set of characters which are classified as bullet markers for **fill-paragraph**(2). If these characters are encountered in the first **$fill-bullet-len** characters of the paragraph AND the character is followed by a `SPACE` or a `tab` character then the user is given the option to indent to the right of the bullet.

**$fill-bullet-len** determines the maximum depth into the paragraph (in characters) the filling routines should search for a bullet character. The default value is 15. Note that the paragraph starts at the first non-white space character. e.g. to detect "`xviii)` " as a bullet then the bullet length must be set to at least 6 to detect the bullet character "`)`".

## EXAMPLE

Examples of filled bullet paragraphs are shown as follows, based on the default **$fill-bullet** character set.

```
a) This is an  example of a  fill-paragraph.  The  closing
   bracket is classified as a bullet character and filling
   optionally takes place to the right of the bullet.

a] Another paragraph

*  A bullet paragraph

1. A numbered paragraph.

item - A dashed bullet.
```

## SEE ALSO

**$buffer-fill-col**(5), **$buffer-fill-mode**(5), **$fill-col**(5), **$fill-ignore**(5), **$fill-mode**(5), **fill-paragraph**(2), **justify**(2m).

Ref: $fill-bullet(5) File: m5var053.5 Date: 2022/06/12

## NAME

$fill-col - Global paragraph mode; right fill column $buffer-fill-col - Buffer paragraph mode; right fill column

## SYNOPSIS

**$fill-col** *columnNumber*
**$buffer-fill-col** *columnNumber*

-1 <= *columnNumber* <= 32767 the default is `78`.

## DESCRIPTION

The **fill-col** variables specify the position of the right text margin, or fill column, when used with **fill-paragraph**(2).

**$fill-col** defines the global default fill column number, all new buffers that are subsequently created assume a fill column value of **$fill-col**. *columnNumber* defaults to 78 when undefined. This value is used in conjunction with **justify**(2m) and **wrap**(2m) modes.

**$buffer-fill-col** specifies the fill column value to use in a buffer, a value of **$fill-col** is used as a default when the buffer is created. Modification of the **$buffer-fill-col** causes the buffer fill column position to be modified, this does not affect any other buffer. The value is typically set by the **fhook-xxx** macro when initializing a buffer using the Language Templates. **$buffer-fill-col** may also be modified from **buffer-setup**(3).

## SEE ALSO

**$buffer-fill-mode**(5), **$fill-mode**(5), **buffer-mode**(2), **buffer-setup**(3), **fill-paragraph**(2), **justify**(2m), **wrap**(2m).

Ref: $fill-col(5) File: m5var021.5 Date: 2022/06/12

## NAME

$fill-eos - Paragraph filling; end of sentence fill characters
$fill-eos-len - Paragraph filling; end of sentence padding length

## SYNOPSIS

**$fill-eos** "*string*"; Default is "`. ! ?`"

**$fill-eos-len** *integer*; Default is `1`
0 <= *integer* <= *n*

## DESCRIPTION

**$fill-eos** defines the end of sentence character set. Sentences ending in these characters are padded with additional *end-of-sentence* spaces, as defined by **$fill-eos-len**.

**$fill-eos-len** sets the number of spaces inserted after a full stop during paragraph filling. The default is 1 space.

## SEE ALSO

**fill-paragraph**(2).

Ref: $fill-eos(5) File: m5var012.5 Date: 2022/06/12

## NAME

$fill-ignore - Ignore paragraph filling character(s)

## SYNOPSIS

**$fill-ignore** "*string*"; Default is ">_@"

## DESCRIPTION

**$fill-ignore** describes a set of characters used by **fill-paragraph**(2) which disable paragraph filling when they appear at the start of a paragraph. An obvious example is an inserted mail message which is usually quoted with ">" characters. Any attempt to fill the paragraph causes **fill-paragraph** to skip to the end of it.

## EXAMPLE

This is an example of an ignored paragraph when encountered by **fill-paragraph** with the default ignore character set.

```
> This is an example of a paragraph that
> is ignored.
```

## SEE ALSO

**$buffer-fill-col**(5), **$buffer-fill-mode**(5), **$fill-bullet**(5), **$fill-col**(5), **$fill-mode**(5), **fill-paragraph**(2), **justify**(2m).

Ref: $fill-ignore(5) File: m5var054.5 Date: 2022/06/12

## NAME

$fill-mode - Global paragraph mode; justification method $buffer-fill-mode - Buffer paragraph mode; justification method

## SYNOPSIS

**$fill-mode** *justification*
**$buffer-fill-mode** *justification*
*justification* `b` — `c` — `l` — `n` — `o` — `r` — `B` — `C` — `L` — `N` — `R` the default is `N`

## DESCRIPTION

The **fill-col** variables specify the text justification modes for text filling of **fill-paragraph**(2).

**$fill-mode** defines the global default justification mode i.e. *left*/*right*/*both*/... The default value is *none automatic* (**N**). The modes available are:

**b** Both Enables left and right margin justification.

**c** Center
Enables center justification.

**l** Left Enables left justification.

**n** None No filling is performed, adjacent lines are not merged into a single line. This subtly different from *left* justification which fills lines to the **$fill-col**(5).

**o** One Line
Enables the filling of the paragraph to a single line. Typically used to prepare a file for transfer to a word processing package.

**r** Right Enables right justification.

**B** Both (automatic)
Automatically determines the mode, defaulting to left and right (both) justification.

**C** Center (automatic)
Automatically determines the mode, defaulting to center justification.

**L** Left (automatic)
Automatically determines the mode, defaulting to left justification.

**N** None (automatic)
Automatically determines the mode, defaults to *both* and not *none*.

**R** Right (automatic)
Automatically determines the mode, defaulting to right justification.

**$buffer-fill-mode** specifies the fill mode value to use in a buffer, a value of **$fill-mode** is used as a default when the buffer is created. Modification of the **$buffer-fill-mode** causes the buffer fill mode to be modified, this does not affect any other buffer. The value is set by the **fhook-xxx** macro when initializing a buffer using the Language Templates. **$buffer-fill-mode** may also be modified from **buffer-setup**(3).

The buffer lines are automatically justified only when the justification mode **justify**(2m) is enabled. Justification is performed between the left and right margins, defined as 0 and **$buffer-fill-col**(5) respectively.

### Automatic Filling

Automatic filling is performed when the mode **$buffer-fill-mode** is specified in upper case. The format of the line (and adjacent lines) is interrogated and an *informed* guess is made as to the expected formating which is then adopted. The criteria for automatic formatting is defined as follows:

*center* If the left and right margins contain the same amount of white space +/-1 character then the paragraph is centered.

*right* If the text commences past half of the **$buffer-fill-col**(5) (i.e. first half of the line comprises white space) AND the line extends to, or past, the `$buffer-fill-col` then the paragraph is assumed to be right justified.

*none* If the text commences in column 0 and occupies less than half of the line then the paragraph is assumed to be not justified. (i.e. left justified, but consecutive lines of the paragraph are not filled)

*default* If none of the above criteria are met then the default mode is adopted, as determined by the lower-case value of the **$buffer-fill-mode** value.

### SEE ALSO

**$buffer-fill-col**(5), **$fill-col**(5), **buffer-mode**(2), **buffer-setup**(3), **fill-paragraph**(2), **justify**(2m).

Ref: $fill-mode(5) File: m5var028.5 Date: 2024/11/18

## NAME

$find-words - Filtered word list

## SYNOPSIS

**$find-words** *word*

## DESCRIPTION

**$find-words** must first be initialized to the required filter string, if the variable is evaluated before it is initialized the value will be set to "*ABORT*" and the command will fail.

The filter string can contain wild-card characters compatible with most file systems, namely:

**?** Match any character.

**[abc]** Match character only if it is *a*, *b* or *c*.

**[a-d]** Match character only if it is *a*, *b*, *c* or *d*.

**[^abc]** Match character only if it is not *a*, *b* or *c*.

***** Match any number of characters.

Note that these are not the same characters used by **exact**(2m) mode.

Once initialized, evaluating **$find-words** returns the next word found in the main spell dictionaries which matches the filter until no more words are found, in which case an empty string is returned.

## EXAMPLE

The following example finds all the words with "*foo*" in it (e.g. "*footnote*"), printing them to the massage line one at a time. Note that **&set**(4) is used on the **!while**(4) statement to avoid evaluating **$find-words** twice per loop.

```
set-variable $find-words "*foo*"
!while &not &seq &set #l0 $find-words ""
    100 ml-write &cat "Word: " #l0
!done
```

## NOTES

The order of the words is undefined.

Due to the way words are derived, it is possible to have two or more copies of a word in the dictionary. If this is a matching word **$find-words** will return the word two or more times.

## SEE ALSO

**spell**(2).

Ref: $find-words(5) File: m5var067.5 Date: 2022/06/12

## NAME

$frame-depth - Number of lines on the current frame canvas
$frame-width - Number of columns on the current frame canvas

## SYNOPSIS

**$frame-depth** *integer*

3 <= *integer* <= 400

**$frame-width** *integer*

8 <= *integer* <= 400

## DESCRIPTION

These variables allow the viewable size of the current frame canvas to be determined.

**$frame-depth** identifies depth of the current frame given as the number of character lines. This is the whole frame width, not just what is currently visible. The value returned is in the range 3 - *n*, *n* is system dependent but no greater than 400.

**$frame-width** identifies the width of the current frame as the number of character columns. The value returned is in the range 8 - *n*, *n* is system dependent but no greater than 400.

## NOTES

The name of these variables changed from **$screen-depth** and **$screen-width** due to the support for multiple frames introduced in April 2002.

## SEE ALSO

**change-frame-depth**(2), **change-frame-width**(2).

Ref: $frame-depth(5) File: m5var037.5 Date: 2022/06/12

## NAME

$frame-id - Current frame identity

## SYNOPSIS

**$frame-id**

## DESCRIPTION

**$frame-id** retrieves the identity of the current frame, this is a integer value (1..n) that is automatically allocated when the frame was created using **create-frame**(2).

The **$frame-id** is guaranteed to be unique and is typically used in macros to locate a previously created frame.

## SEE ALSO

**create-frame**(2), **delete-frame**(2), **next-frame**(2), **$window-id**(5).

Ref: $frame-id(5) File: m5var086.5 Date: 2022/06/12

## NAME

$frame-title - Frame title prefix
set-title - Set frame title prefix

## SYNOPSIS

**$frame-title** "*string*"
**set-title** "*title*"

## DESCRIPTION

On windowing systems, such as UNIX XTerm and Windows, the frame title is composed of the **$frame-title** followed by the current buffer. Setting the variable can be particularly useful when running MicroEmacs multiple times as an easy way to distinguish between each instance.

The **read-session**(3) command sets this variable to "`MicroEmacs: <session-name>`" so that if multiple instance of MicroEmacs are run with different sessions each can be easily identified.

**set-title** is a simple macro used to set the **$frame-title** variable.

## NOTES

Unlike other frame variables, such as **$frame-id**(5), there is only one **$frame-title** value, changing its value will effect the titles of all external frames.

## SEE ALSO

**create-frame**(2), **read-session**(3), **$frame-id**(5).

Ref: $frame-title(5) File: m5var015.5 Date: 2024/08/06

## NAME

$global-scheme - Default global buffer color scheme.

## SYNOPSIS

`$global-scheme schemeNum`; Default is `0`

## DESCRIPTION

**$global-scheme** defines the default buffer color scheme to *schemeNum*, a color scheme defined by **add-color-scheme**(2).

## SEE ALSO

**add-color**(2), **add-color-scheme**(2), **$buffer-hilight**(5), **$buffer-scheme**(5), **$cursor-color**(5), **$trunc-scheme**(5), **$ml-scheme**(5), **$osd-scheme**(5), **$mode-line-scheme**(5), **$scroll-bar-scheme**(5), **$system**(5).

Ref: $global-scheme(5) File: m5var024.5 Date: 2022/06/12

## NAME

$home - Users ‘home’ directory location

## SYNOPSIS

**$home** *directory*

## DESCRIPTION

**$home** contains the pathname of the users home directory. The file naming convention utilizes tilde (’ ’) to identify the users home directory (`$HOME`). When entering a file name:

```
~/xxx     -> $home/xxx
~yyy/xxx -> $home/../yyy/xxx
```

On most systems this is automatically set to the environment variable ”HOME” if it is defined. On systems (i.e. Microsoft Windows or MS-DOS) where there may be no home defined then the system user directory is used by default.

The value may be explicitly redefined by the user in the start-up file to be a different location. If **$home** is re-defined then ensure that a trailing directory separator is included at the end of the pathname.

## SEE ALSO

**$user-path**(5), **$user-name**(5), **$progdata**(5), **$progname**(5), **$MENAME**(5), **$MEINSTALL-PATH**(5), Variable Functions.

Ref: $home(5) File: m5var026.5 Date: 2022/06/12

## NAME

$idle-time - System idle event delay time

## SYNOPSIS

**$idle-time** *milliseconds*; Default is `1000`

10 <= *milliseconds* <= t

## DESCRIPTION

**$idle-time** sets the time waited between the last user event and the generation of a `idle-pick` key event. When user input stops for **$idle-time** milliseconds MicroEmacs checks the binding of the special `idle-pick` key, if this pseudo key is bound then the command it is bound to will be executed. MicroEmacs will then cycle, generating a `idle-pick` every **$idle-time** milliseconds until user activity starts. At this point a `idle-drop` key event is generated, if this pseudo key is bound then the command it is bound to will be executed.

This system is useful for things which can be done in the background.

## EXAMPLE

The following example is taken from `ssaver.emf` and implements a simple screen saver:

```
set-variable %screen-saver 0
define-macro screen-saver
    !if &not &pinc %screen-saver 1
        !if &seq @cck "idle-pick"
            ; default is to switch on in 5 minutes time
            &cond @? @# 300000 create-callback screen-saver
        !else
            !if &seq @cck "callback"
                @# create-callback screen-saver
            !elif @?
                ; user has suppled argument, install or remove
                !if &gre @# 0
                    &mul @# 60000 global-bind-key screen-saver "idle-pick"
                !else
                    !force global-unbind-key "idle-pick"
                !endif
                set-variable %screen-saver &sub %screen-saver 1
                !return
            !endif
            set-variable @# $frame-depth
            !while &dec @# 1
                2 screen-poke @# 0 $global-scheme &spr "%n" $frame-width " "
            !done
            0 screen-poke 0 0 $global-scheme &spr "%n" $frame-width " "
            set-variable #l0 &bmo "cursor"
            -1 buffer-mode "cursor"
            ; must set this to stop recursion when waiting for a key!
            set-variable %screen-saver 0
            set-variable @# @cg
            set-variable %screen-saver 1
            &cond #l0 1 -1 buffer-mode "cursor"
            screen-update
            0 ml-write
        !endif
    !endif
    set-variable %screen-saver &sub %screen-saver 1
!emacro
```

## NOTES

Care must be taken to ensure that a recursive loop is not created, consider the following example:

```
define-macro bored
    !if &iseq @mc1 "Are you bored (y/n)? " "nNyY" "y"
        ml-write "Play a silly game!"
    !endif
!emacro
global-bind-key bored idle-pick
```

If this was executed MicroEmacs would very quickly crash! As soon as a second past **bored** would execute, which will prompt the user and wait for input. If a second passes without input **bored** will be executed again and again and again until stack memory runs out! To avoid this `idle-pick` should be unbound before waiting for user input, i.e.:

```
define-macro bored
    global-unbind-key idle-pick
    !if &iseq @mc1 "Are you bored (y/n)? " "nNyY" "y"
        ml-write "Play a silly game!"
    !endif
    global-bind-key bored idle-pick
!emacro
global-bind-key bored idle-pick
```

## SEE ALSO

**$delay-time**(5).

## NAME

$indent-width - Global indentation width in characters $buffer-indent-width - Buffer indentation width in characters

## SYNOPSIS

**$indent-width**
**$buffer-indent-width**

## DESCRIPTION

**$indent-width** defines the default indentation width assumed by the editor, this defines the number of characters the **indent**(2) command should use for indenting text. Once the value is set then all new buffers that are subsequently created assume an indent value of **$indent-width**. The default value when omitted is 4 characters.

**$buffer-indent-width** specifies the indent to use in a buffer, the value of **$indent-width** is used as a default when the buffer is created. Modification of the **$buffer-indent-width** value causes the buffer indentation width to be modified. The value is typically set by the **fhook-xxx** macro when initializing a buffer using the Language Templates.

## NOTES

**$buffer-indent-width** and **$indent-width** were introduced into MicroEmacs in November 2004.

## SEE ALSO

**indent**(2).
Language Templates.

Ref: $indent-width(5) File: m5var091.5 Date: 2022/06/12

## NAME

$kept-versions - Number of backups to be kept

## SYNOPSIS

**$kept-versions** *integer*; Default is 0

0 <= *integer* <= n

## DESCRIPTION

**$kept-versions** allows the user to specify the number of backup versions that are required for each file. For file "XXXX", each backup version is renamed to "XXXX`.?`", where `?` is the backup number. If **$kept-versions** is set to 0 this feature is disabled and the default single backup file is created.

The most recent backup will always be `.0` and the last version will be `.n` where `n` is **$kept-versions** - 1. when the file is next saved the `.0` backup file is moved to `.1`, `.1` to `.2` etc, backup `.n` is removed. Evidently if **$kept-versions** it set to a large number this can effect performance.

## RESTRICTIONS

**$kept-versions** may only be used when DOS file name restrictions are not enabled. This means that some systems (such as DOS) cannot use this functionality, see **$system**(5) for more information. Backup files are only created when buffer mode **backup**(2m) is enabled.

## NOTES

This feature is not supported when writing ftp files, a single backup file is created when backup files are enabled.

## SEE ALSO

**$system**(5), **autosv**(2m), **backup**(2m), **ftp**(3), **save-buffer**(2).

Ref: $kept-versions(5) File: m5var074.5 Date: 2022/06/12

## NAME

$line-flags - Current buffer line flags

## SYNOPSIS

**$line-flags** *bitmask*; No default

## DESCRIPTION

The **$line-flags** variable is used to retrieve or define the behavioral settings of the current line in the buffer. This variable is a bit flag, the length of the bit flag is variable. In smaller builds of MicroEmacs where features have been disabled then the field is only 8 bits wide, in a normal build then the full 16 bits are present.

**0x0001** The line has changed as a result of an edit and has not been re-displayed. The flag is cleared when the change has been displayed. This bit is read only.

**0x0002** The line contains an alpha mark defined by **set-alpha-mark**(2). This bit is read only.

**0x0004** The line contains a narrow region, set by **narrow-buffer**(2). This bit is read only.

**0x0008** The line contains some other type of anchor. This bit is read only.

**0x0010** The line should be saved without a new-line or nil termination character.

**0x0020** The line is a mark-up line used for annotation and should not be saved on a write operation. This bit permits lines in a buffer to be inserted for display or annotation purposes which are automatically discarded when the buffer is written to file. Mark-up lines are sometimes used by the buffer folding and collapsing to annotate folds.

**0x0040** The line is protected. A protected line may be deleted or edited however it is not possible to split or join a protected line with any other line. This is used by the folding or collapse mechanisms to protect narrrow regions from editing operations.

**0x0080-0x0800**
User defined. Users may use the upper most bits from left to right. JASSPA may use more bits in future releases which shall be consumed from the lower bits, right to left.

**0xf000** The shifted **$line-scheme**(5) setting. A value of zero indicates that there is no line scheme in effect on the line.

Where a bit flag is defined as read only then an attempt by the user to set a read-only bit then the bit setting is ignored, any remaining writable bits are set as requested.

## EXAMPLE

Refer to the file `fold.emf` for examples of the use of **$line-flags**.

## NOTES

**$line-flags** was first introduced in MicroEmacs '04.

## SEE ALSO

**narrow-buffer**(2), **set-alpha-mark**(2), **set-variable**(2), **$buffer-hilight**(5), **$buffer-scheme**(5) **$line-scheme**(5).

Ref: $line-flags(5) File: m5var085.5 Date: 2022/06/12

## NAME

$line-scheme - Set the current line color scheme

## SYNOPSIS

**$line-scheme** *schemeNum*; Default is `-1`

## DESCRIPTION

**$line-scheme** sets the color scheme to be used for the current line of the current window. The given *schemeNum* can be any scheme number previously defined by the function **add-color-scheme**(2).

A line's $line-scheme setting is removed by setting the variable to -1.

A $line-scheme setting takes precedence over the buffer's color scheme ( **$buffer-scheme**(5)) and the buffer's hilighting scheme ( **$buffer-hilight**(5)).

## EXAMPLE

**c-hash-eval**(3) greys out lines of text by doing:

```
set-variable $line-scheme %lblack
```

The lines are rest by doing

```
set-variable $line-scheme -1
```

The **gdb**(3) interface hilights the current line of source by doing:

```
set-variable $line-scheme %yellow-lblack
```

## NOTES

Due to line storage restrictions, only 15 different color schemes can be used in a buffer at any one time. When the 16th color scheme is used it replaces the first color scheme, all lines using the first color scheme will be colored using the new color scheme.

## SEE ALSO

**add-color-scheme**(2), **c-hash-eval**(3), **$buffer-hilight**(5), **$buffer-scheme**(5), **$line-flags**(5), **$mode-line-scheme**(5), **$scroll-bar-scheme**(5), **$system**(5).

Ref: $line-scheme(5) File: m5var056.5 Date: 2022/06/12

## NAME

$line-template - Command line regular expression search string

## SYNOPSIS

**$line-template** "*string*"; Default is ""

## DESCRIPTION

**$line-template** defines a regular expression search string used to identify a line number in the **grep**(3) and **compile**(3) buffers. The format of the string is the same as magic mode search strings (see **search-forward**(2)).

## EXAMPLE

The line number may be considered to contain any numeric number, thus **$line-template** is defined as:

```
set-variable $line-template "[0-9]+"
```

This correctly identifies "45" in the following ***grep*** output example:

```
foo.c: 45:      printf("hello world\n") ;
```

## SEE ALSO

**$file-template**(5), **compile**(3), **get-next-line**(2), **grep**(3), **search-forward**(2).

## NAME

$major-mode-ids - Filtered major-mode ID list

## SYNOPSIS

**$major-mode-ids** *MajorModeId*

## DESCRIPTION

**$major-mode-ids** must first be initialized to the required filter string, if the variable is evaluated before it is initialized the value will be set to "*ABORT*" and the command will fail. The filter takes the form of a regex.

Once initialized, evaluating **$major-mode-ids** returns the name of the next major-mode which matches the filter until no more major-modes are found, in which case an empty string is returned.

## EXAMPLE

The following example prints out the full name of all major-modes to the massage line one at a time. Note that **&set**(4) is used on the **!while**(4) statement to avoid evaluating **$major-mode-ids** twice per loop.

```
set-variable $major-mode-ids ".*"
!while &not &seq "" &set #l0 $major-mode-ids
  0 major-mode #l0
  set-variable #l1 $result
  100 ml-write &spr "mode %s: %s" #l0 &lget #l1 5
!done
```

## SEE ALSO

**major-mode**(2), **$buffer-major-mode**(5), **$mode-names**(5), **$command-names**(5), Regular Expressions.

## NAME

$MEINSTALLPATH - MicroEmacs system macros search path
$MEPATH - MicroEmacs search path
$MEUSERPATH - MicroEmacs user directory

## SYNOPSIS

*[Microsoft Windows/MS-DOS]*
**$MEINSTALLPATH** *<path1>***;***<path2>***;....;***<pathn>*
**$MEPATH=** *<path1>***;***<path2>***;....;***<pathn>*
**$MEUSERPATH** *<path>*
*[UNIX]*
**$MEINSTALLPATH** *<path1>***:***<path2>***:....:***<pathn>*
**$MEPATH=** *<path1>***:***<path2>***:....:***<pathn>*
**$MEUSERPATH** *<path>*

## DESCRIPTION

These environment variables control the run-time search paths for MicroEmacs and affect the operation of **$search-path**(5) and **$user-path**(5), the macro search path and user macro directory respectively.

**$MEINSTALLPATH** is an environment variable that overrides any internally defined search path for MicroEmacs system macros and defines the base directory(s) of the MicroEmacs system macros. The presence of the environment variable affects **$search-path**(5).

The environment variable **$MEINSTALLPATH** is ignored if **$MEPATH** is defined. The search path is always included after any userpath defined by **$user-path**(5).

**$MEUSERPATH** is an environment variable that defines the users private directory where user configuration files such as $user-name**.emf**, $user-name**.erf** and personal spelling dictionaries are stored. This is an absolute directory path. The environment variable sets the MicroEmacs variable **$user-path**(5).

**$MEPATH** is an environment variable that may be used to define the absolute search path for MicroEmacs macro files and affects the setting of **$search-path**. The path should include the system macros location(s) and should include the user directory location **$user-path**(5) as the first path when the environment variable **$MEUSERPATH**(5) is not defined. If **$MEUSERPATH** is defined then this path name appears before the **$MEPATH** variable to define the user directory location. **$MEPATH** is not searching, so ALL all directories to be searched must be included in the path specification.

**$MEUSERPATH**(5) is the preferred method used to defined the user directory.

### Search Paths

MicroEmacs needs to know the installation directory location(s) of its macro files when it executes, this includes both user defined and system macros. MicroEmacs performs a searching operation of well-known directory locations where it may expect to find these macros at start up, the locations that it has found are saved in macro variables **$user-path**(5) and **$search-path**(5) and are used to subsequently locate files thereafter. The environment variables defined here allow that default search process to be modified by the user or system administrator.

Directories are specified as *searching* if a search operation is performed. The following sub-

directories of a base directory are searched for the existence of subdirectories:

$user-name - private user directory.
*company* - company files.
*macros* - JASSPA standard macros.
*spelling* - spelling dictionaries.
. - base directory.

MicroEmacs performs a searching operation on some directories to locate information. Generally, if the directory exists then it is included in the search path, otherwise it is excluded. Any subdirectories that are found are added in the order presented above. The $search-path is ONLY evaluated at start-up therefore directories created after MicroEmacs is running are not detected until the editor is closed down and re-started.

### UNIX

The defined search paths for UNIX are defined as follows:

Default install directories:

*/opt/jasspa* - searching (SunOS, HP-UX, AIX and IRIX only)
*/usr/share/jasspa* - searching
*/usr/local/share/jasspa* - searching (BSD systems only)
*/usr/local/jasspa* - searching

The default user location is defined as follows.

*$(passwd-home)/.config/jasspa* - searching

MicroEmacs searches for user defined configuration files in this directory and will write new user configuration files to this directory. The behaviour of the search may be over-ridden by setting some environment variables.

`$MEUSERPATH` - not searching

Specified as a single pathname and redefines the *$user-path* where user macros are placed, this overrides the *$(passwd-home)/.config/jasspa* directory location.

`$MEINSTALLPATH` - searching

Specified as multiple pathname components and overrides the default install directory location (i.e. `/usr/local/jasspa`, etc. Typically used when the system macro directory has been install in another location that is not included in the default search path. This does not affect the user directory location.

`$MEPATH` - not searching

Overrides the *$(passwd-home)/.config/jasspa* and default search paths (including `$MEINSTALLPATH`), MicroEmacs will use this directory as the search path. The user directory is assumed to be the FIRST directory path in the string unless the `$MEUSERPATH` environment variable is defined which specifies the user path.

### Microsoft Windows

he defined search paths for WINDOWS are defined as follows:

Default search locations:

*Executable directory location* - searching

The default user location is defined as follows. MicroEmacs searches for user defined configuration files in this directory and will write new user configuration files to this directory.

*$APPDATA\jasspa* - not searching

If the system does not define a user application data location (**Win95** and **Win32s**) then a directory $user-name or the executable location is used.

The behaviour may be over-ridden by setting the environment variables as follows:

`$MEUSERPATH` - not searching
: Specified as a single pathname and redefines the users home directory, this overrides the *$APPDATA\jasspa* directory location.

`$MEINSTALLPATH` - searching
: Specified as multiple pathname components and overrides the default install directory location. Typically used when the macro directory has been install in another location that is not in the same directory as the executable. This does not affect the user directory location.

`$MEPATH` - not searching
: Overrides the *$APPDATA\jasspa* and default search paths. MicroEmacs will use the FIRST directory path as the $user-path unless `$MEUSERPATH` is defined.

### MS-DOS

MS-DOS does not include a concept of a user name or user directory.

Default search locations:

*Executable directory location* - searching

If a sub-directory exists at the executable location with name $user-name then this is used as $user-path, otherwise the execution directory location is used.

The behaviour may be over-ridden using the environment variables.

`$MEUSERPATH` - not searching
: Specified as a single pathname and redefines the $user-path overriding the *executable* directory location.

`$MEINSTALLPATH` - searching
: Specified as multiple pathname components and overrides the default executable directory location. Typically used when the macro directory has been installed at a different location that is not in the same directory as the executable. This does not affect the user directory location.

`$MEPATH` - not searching
: Overrides the *Executable...* and default search paths (including `$MEINSTALLPATH`). MicroEmacs will use FIRST directory path as the $user-path unless `$MEUSERPATH` is defined.

### SEE ALSO

**me**(1), **$home**(5), **$progname**(5) **$search-path**(5), **$user-name**(5), **emf**(8), **erf**(8).

## NAME

$MENAME - MicroEmacs user name
$LOGNAME - System user name (UNIX)

## SYNOPSIS

**$MENAME** *string*;
**$LOGNAME** *string*

## DESCRIPTION

**$MENAME** is an environment variable which may be used to initialize the MicroEmacs environment for a given user. At start-up, if **$MENAME** is defined then the user's configuration and history file "`name`.*erf*" is located and read, where `name` is the variable value. The environment variable is used to initialise the **$user-name**(5) variable.

Where the user name has been specified on the command line using the **-u** option to **me**(1) then the environment variable is ignored and **-u** option takes precedence.

The user configuration and session file has many uses, see **user-setup**(3) and **read-session**(3) for more information.

### Microsoft Windows Environments

Within Microsoft Windows environments, if *login* is enabled then the users login name is automatically used as the first choice login name. No environment variables need to be set. If login is not enabled then one of the aforementioned methods should be used.

### Microsoft Windoes

In Microsoft Windows environments, **$LOGNAME** is sometimes defined.

## NOTES

The three variables must be defined before start-up for them to have any effect.

## SEE ALSO

**me**(1), **user-setup**(3), **read-session**(3), **$user-name**(5), **$user-path**(5), **$MEPATH**(5).

Ref: $MENAME(5) File: m5var076.5 Date: 2022/06/12

## NAME

$ml-scheme - Message line color scheme

## SYNOPSIS

**$ml-scheme** *schemeNum*; Default is 0

## DESCRIPTION

**$ml-scheme** defines the color scheme to be used on the message line, the color scheme *schemeNum* identifies the foreground and background color and is defined by an invocation to **add-color-scheme**(2).

The background color is always defined by **$global-scheme**(5).

## SEE ALSO

**$global-scheme**(5), **$osd-scheme**(5), **$mode-line-scheme**(5), **$scroll-bar-scheme**(5), **$system**(5), **add-color-scheme**(2).

Ref: $ml-scheme(5) File: m5var030.5 Date: 2022/06/12

## NAME

$mode-chars - Mode characters

## SYNOPSIS

**$mode-chars**

## DESCRIPTION

**$mode-chars** is a read-only variable that returns a string that contains all of the buffer mode characters. The variable is typically used in the macros to determine the integer value of a buffer mode, the string is defined as follows:

```
AaBbcYzdDeEfHIJkMnNOPQrSTtU12345678VW
```

The index into the string matches **$show-modes**(5).

## SEE ALSO

**$show-modes**(5).

## NAME

$mode-line - Mode line format

## SYNOPSIS

**$mode-line** "*string*"; Default is "`%s%r%u me (%e) - %l %b (%f) `"

## DESCRIPTION

**$mode-line** defines the format of the mode line printed for every window, where the character following a percent ('`%`') has the following effect:

`%` Prints a percentage escape character.
`-` Prints a literal minus character ('`-`') - see NOTES.
`b` Prints the current buffer's name.
`c` Prints the current buffer's column number.
`d` Prints the current day (01 to 31).
`f` Prints the current buffer's file name.
`H` Prints the current hour of the day (00 to 23).
`i` Prints the current day (1 to 31).
`K` Prints the current keyboard macro status.
`k` Prints the current hour of the day (0 to 23).
`l` Prints the current buffer's line number.
`M` Prints the current minute of the hour (01 to 59).
`m` Prints the current month (01 to 12).
`n` Prints the current buffer's total number of lines.
`N` Prints the current minute of the hour (1 to 59).
`O` Prints the current buffer's shown modes - see NOTES.
`o` Prints the current month (1 to 12).
`r` Prints the current root user status (UNIX or Windows only).
`s` Prints the horizontal window split character.
`u` Prints the current buffer's (un)changed or view mode flag.
`Y` Prints the current year (2 digits).
`y` Prints the current year (4 digits).
`*` All other characters are printed literally.

## NOTES

- Refer to **$window-chars**(5) for the characters utilized in the mode line. Typically a the '`-`' character is changed to a '`=`' if it is the current window. If a '`-`' is always required, use "`%-`".
- The buffer modes shown by **%O** is controlled by system variable **$show-modes**(5).
- A buffer can have its own mode-line, and be uneffected be the global mode line, see **$buffer-mode-line**(5).

## SEE ALSO

**$buffer-mode-line**(5), **$mode-line-scheme**(5), **$show-modes**(5), **$window-chars**(5).

Ref: $mode-line(5) File: m5var031.5 Date: 2025/03/26

## NAME

$mode-line-scheme - Mode line color scheme

## SYNOPSIS

**$mode-line-scheme** *schemeNum*; Default is `1`

## DESCRIPTION

Sets the window mode-line color scheme, defining the foreground and background colors. The *schemeNum* is defined by a previous invocation to **add-color-scheme**(2).

## SEE ALSO

**add-color-scheme**(2), **$global-scheme**(5), **$ml-scheme**(5), **$scroll-bar-scheme**(5), **$system**(5).

Ref: $mode-line-scheme(5) File: m5var032.5 Date: 2022/06/12

## NAME

$mode-names - Filtered mode name list

## SYNOPSIS

**$mode-names** *ModeName*

## DESCRIPTION

**$mode-names** must first be initialized to the required filter string, if the variable is evaluated before it is initialized the value will be set to "*ABORT*" and the command will fail. The filter takes the form of a regex.

Once initialized, evaluating **$mode-names** returns the name of the next mode which matches the filter until no more modes are found, in which case an empty string is returned.

## EXAMPLE

The following example prints out the name of all modes to the massage line one at a time. Note that **&set**(4) is used on the **!while**(4) statement to avoid evaluating **$mode-names** twice per loop.

```
set-variable $mode-names "*"
!while &not &seq &set #l0 $mode-names ""
    100 ml-write &cat "mode: " #l0
!done
```

## SEE ALSO

**buffer-mode**(2), **&bmode**(4), **$buffer-names**(5), **$command-names**(5), Regular Expressions.

Ref: $mode-names(5) File: m5var065.5 Date: 2022/06/12

## NAME

$mouse - Mouse configuration variable

## SYNOPSIS

**$mouse** *bitmask*; Default is system dependent

## DESCRIPTION

The **$mouse** is used to define and configure the MicroEmacs mouse support, it is a bit based flag where:

**0x00f** Defines the number of button the mouse has, only values 1, 2 & 3 are useful. By default MicroEmacs uses the system information to determine the number of buttons on the mouse, this is not fool proof so the user can set these bits to the appropriate number if the initial value is incorrect.

**0x010** If set the mouse is enabled, if clear the mouse will not function. On systems which do not support mice (such as UNIX Termcap) this bit will be clear and can not be altered.

**0x020** If set the buttons are reversed, i.e. the left button becomes the right and vice versa. By default this bit is clear.

**0xf0000**

Defines the current mouse icon to used, valid values are as follows:

**0x00000** - Set mouse to default icon.
**0x10000** - Set mouse to arrow icon.
**0x20000** - Set mouse to text I-beam icon.
**0x30000** - Set mouse to crosshair icon.
**0x40000** - Set mouse to the grab icon.
**0x50000** - Set mouse to the wait icon.
**0x60000** - Set mouse to the stop icon.

This feature is not supported on some systems and on others some icons are not obvious due to platform limitations.

## EXAMPLE

The following example checks that the mouse is currently available, if not, it aborts.

```
!if &not &band $mouse 0x10
    ml-write "[Mouse support is not currently available]"
    !abort
!endif
```

## NOTES

The mouse can be easily configured using **user-setup**(3).

## SEE ALSO

**user-setup**(3), **$system**(5), **$platform**(5).

Ref: $mouse(5) File: m5var073.5 Date: 2022/06/12

## NAME

$mouse-pos - Mouse position information

## SYNOPSIS

**$mouse-pos** *integer*

## DESCRIPTION

**$mouse-pos** is generated by invocation of the command **set-cursor-to-mouse**(2). The variable is set to a value that indicates the position of the mouse within a window. The values to the mouse intersection are interpreted as follows:

**0 - Text area**
: Intersection with the window text area.

**1 - Message Line**
: Intersection with the message line.

**2 - Mode Line**
: Intersection with the mode line.

**3 - Horizontal Separator**
: Intersection with the horizontal window separator. This value is only set if a scroll bar is not present.

**4 - Up Arrow**
: Intersection with the scroll bar up-arrow character.

**5 - Upper Shaft**
: Intersection with the scroll bar upper shaft (above the scroll box).

**6 - Scroll Box**
: Intersection with the scroll bar scroll box.

**7 - Lower Shaft**
: Intersection with the scroll bar lower shaft (below the scroll box).

**8 - Down Arrow**
: Intersection with the scroll bar down-arrow character.

**9 - Corner**
: Intersection with the window corner, that is the character at the intersection of the scroll bar (or separator) and the mode line.

**10 - Menu Line**
: Intersection with the menu line.

**255 - Error**
: The position of the mouse could not be determined. This value should not arise, if it does then it is an indication that the window structure is probably corrupted. A **delete-other-windows**(2) is suggested or rapid exit from the editor after a **save-some-buffers**(2) command to save any edits (latter option is preferred).

**Bit 4 - 2nd Column**
: Bit 4 (16) is set if 2 character column scroll bar or vertical window separator is in effect and the cursor exists in the second column This value is bitwise OR'ed with the aforementioned intersection values.

## EXAMPLE

The following macro can be used to print out the current position of the mouse, try binding the macro to the "mouse-move" key:

```
define-macro print-mouse-position
    !force set-cursor-to-mouse
    set-variable #l0 &band $mouse-pos 15
    !if &equ #l0 0
        ml-write "Mouse in text window"
    !elif &equ #l0 1
        ml-write "Mouse on message line"
    !elif &equ #l0 2
        ml-write "Mouse on Mode line"
    !elif &and &gre #l0 2 &les #l0 10
        ml-write "Mouse on scroll bar"
    !elif &equ #l0 10
        ml-write "Mouse on corner"
    !elif &equ #l0 11
        ml-write "Mouse on menu line"
    !endif
!emacro

global-bind-key print-mouse-position mouse-move
```

**$mouse-pos** is utilized by the mouse picking code, found in macro file `mouse.emf`.

## SEE ALSO

**$mouse-x**(5), **$mouse-y**(5), **set-cursor-to-mouse**(2), **set-scroll-with-mouse**(2).

Ref: $mouse-pos(5) File: m5var038.5 Date: 2022/06/12

## NAME

$mouse-x - Mouse X (horizontal) position
$mouse-y - Mouse Y (vertical) position

## SYNOPSIS

**$mouse-x** *integer*

0 <= *integer* <= $frame-width - 1

**$mouse-y** *integer*

0 <= *integer* <= $frame-depth - 1

## DESCRIPTION

**$mouse-x** and **$mouse-y** are automatically set to the position of the mouse at the last mouse event, where an event is a button press or release. Initialized to 0,0. The top left character of the screen is coordinate 0,0 bottom right is $frame-width, $frame-depth.

## NOTES

These variables can not be set. Any attempt to set them will result in an error.

## SEE ALSO

**set-cursor-to-mouse**(2), **$mouse-pos**(5), **$cursor-x**(5), **$frame-depth**(5), **$frame-width**(5).

Ref: $mouse-x(5) File: m5var033.5 Date: 2022/06/12

## NAME

$osd-scheme - OSD color scheme

## SYNOPSIS

**$osd-scheme** *schemeNum*; Default is `1`

## DESCRIPTION

**$ml-scheme** defines the color scheme by default on an **osd**(2) dialog, the color scheme *schemeNum* identifies the foreground and background color and is defined by an invocation to **add-color-scheme**(2). Every osd dialog can over-ride this value by using the '*S*' flag.

## SEE ALSO

**osd**(2), **add-color-scheme**(2), **$global-scheme**(5), **$ml-scheme**(5), **$mode-line-scheme**(5), **$scroll-bar-scheme**(5), **$system**(5).

Ref: $osd-scheme(5) File: m5var009.5 Date: 2022/06/12

## NAME

$pause-time - Length of time to pause for user

## SYNOPSIS

**$pause-time** *milliseconds*; Default is `2000`

0 <= *milliseconds* <= *t*

## DESCRIPTION

Defines the number of milliseconds to pause, allowing the user to see some information. The pause time is used predominantly by the fence match operation, determining the time that the opening fence is displayed, see **fence**(2m) mode.

The **$pause-time** is also used during the printing of some error messages on the message-line. These errors are typically important ones that should not be ignored so the system pauses to ensure the user has enough time to read them.

## NOTES

The variable name was changed from **$fmatchdelay** in December 2004.

## SEE ALSO

**fence**(2m).

Ref: $pause-time(5) File: m5var023.5 Date: 2022/06/12

## NAME

$platform - MicroEmacs host platform identifier
%platform - MicroEmacs host platform type identifier

## SYNOPSIS

**$platform** "*string*"; Default is platform specific
***%*platform** "*string*"; Default is platform specific

## DESCRIPTION

The **$platform** variable is a fixed ASCII string used to identify the current working platform, attempts to set this variable result in an error returned from **set-variable**(2).

Possible values are:

"**aix**" All IBM AIX O/S.

"**dos**" All IBM-PCs and compatibles running MS-DOS.

"**freebsd**"
All FreeBSD O/S.

"**hpux**" All Hewlett Packard's with HP-UX O/S.

"**irix**" All Silicon Graphics (SGI) IRIX platforms 4.x, 5.x, 6.x.

"**linux**" All LINUX O/S.

"**sunos**" All Sun's with SUNOS O/S.

"**unixwr1**"
PC based UNIX platform (Consensus and Unixware).

"**windows**"
Microsoft Windows based systems including Windows 3.x (with Win32s), Windows '95 and NT etc.

**$platform** is often used in **.emf** files to allow portability of macro files across platforms, allowing macro files to perform platform specific operations. **$system**(5) is also often used for this purpose as its value is easier to assess.

***%*platform** is created at start-up when `me.emf` is executed, its value is identical to **$platform** except when the platform is a console in which case a '`c`' is appended to the $platform value, e.g. for MicroEmacs running a termcap version on LINUX the value will be "`linuxc`". The variable is used when the console and window based versions need to be distinguish, e.g. some of the user-setup settings.

## EXAMPLE

The following example is taken from the **me.emf** file which uses the **$platform** variable to load the platform specific initialization files.

```
;
; load in the platform specific stuff
execute-file $platform
```

This could be more explicitly done by:

```
;
; load in the platform specific stuff
!if   &seq $platform "dos"            ; is it an IBM-PC running dos ?
    execute-file "dos"
!elif &seq $platform "irix"           ; is it an sgi ?
    execute-file "irix"
!elif &seq $platform "hpux"           ; is it an hp ?
    execute-file "hpux"
    .
    .
!endif
```

## NOTES

The **$platform** variable can not be set. Any attempt to set it will result in an error.

## SEE ALSO

**$system**(5), **set-variable**(2).

Ref: $platform(5) File: m5var034.5 Date: 2026/02/17

## NAME

$progdata - Users program data directory

## SYNOPSIS

**$progdata** *directory*

## DESCRIPTION

**$progdata** contains the pathname of the user's system program data directory for MicroEmacs, this is unlikely to be the same location as **$user-path**(5).

On Windows the path will be the jasspa sub-directory within the user's application data path, i.e. '`%AppData%/jasspa/`'. On UNIX systems the path will be '**$home/.config/jasspa/**' and on other systems it will be '`$home/jasspa/`'.

Note that the path may not currently exist, particularly if **$user-path**(5) points to a different location due to one of **$MEUSERPATH**(5), **$MEPATH**(5) or **$MEINSTALLPATH**(5) variables being set.

## SEE ALSO

**$user-path**(5), **$home**(5), **$progname**(5), **$MEUSERPATH**(5), **$MEPATH**(5), **$MEINSTALLPATH**(5), Variable Functions.

Ref: $progdata(5) File: m5var018.5 Date: 2025/05/02

## NAME

$progname - Program file name

## SYNOPSIS

**$progname** *string*

## DESCRIPTION

**$progname** contains the absolute file and path to the MicroEmacs binary file currently being executed. This value is typically used by macros to spawning another MicroEmacs sessions and to determine where MicroEmacs is being executed from.

## EXAMPLE

The following example is used to spawn of another MicroEmacs command to create a C-tags file:

```
shell-command &cat $progname " \"@ctags\" *.c *.h"
```

## SEE ALSO

**me**(1), **shell-command**(2), **$progdata**(5), **$home**(5), **$search-path**(5), **$user-name**(5), **$user-path**(5).

Ref: $progname(5) File: m5var066.5 Date: 2025/08/28

## NAME

$quiet - Quiet mode control variable

## SYNOPSIS

**$quiet**

## DESCRIPTION

**$quiet** is variable that controls the audible and visual warnings. When the variable is set to 0 then quiet mode is disabled, when set to 1 then quiet mode is enabled.

When quiet mode is enabled, visual warnings are given instead of the default audible warning. This mode can only be globally changed, an error will occur if an attempt is made to change the mode for a buffer.

The default state is on, so users of MicroEmacs can relax in the knowledge that they will not annoy other people when things go wrong.

When disabled the system bell is rung when required, otherwise the usual visual warning is the ”[BELL]” string, printed on the bottom right hand side.

## SEE ALSO

**!bell**(4).

Ref: $quiet(5) File: m5var093.5 Date: 2022/06/12

## NAME

$random - Generate a random number

## SYNOPSIS

`$random` *integer*

0 <= *integer* <= 16777215

## DESCRIPTION

The **$random** variable returns a random number in the range 0 - 16777215 (0x0ffffff) on reference to the variable.

The random number generator does not need seeding first as this is handled by MicroEmacs, any attempt to set the variable will result in an error.

The range of the random number generator is fixed across all systems. The value is typically capped to the required range using the **&mod**(4) arithmetic operator, but note that this creates a slight biasing if 16777216 is not a multiple of the cap (there is a bias towards numbers less than the remainder, but this is generally ignorable).

## NOTES

MicroEmacs uses algorithm xoshiro128++ to generate 32 bit random numbers and then rotated right 8 bits to discard the least significant byte and avoid negative numbers.

This should not be considered a cryptographically secure generator.

## EXAMPLE

In the following example, the returned value is used with the **&mod** operator to limit the value to a desired range:

```
set-variable %random0to9 &mod $random 10
```

## SEE ALSO

**&mod**(4).

Ref: $random(5) File: m5var035.5 Date: 2023/10/17

## NAME

$rcs-file - RCS (and SCCS) file name
$rcs-ci-com - RCS (and SCCS) check in command
$rcs-cif-com - RCS (and SCCS) check in first command
$rcs-co-com - RCS (and SCCS) check out command
$rcs-cou-com - RCS (and SCCS) check out unlock command
$rcs-ue-com - RCS (and SCCS) unedit file command

## SYNOPSIS

**$rcs-file** "*string*"; Default is ""
**$rcs-ci-com** "*string*"; Default is ""
**$rcs-cif-com** "*string*"; Default is ""
**$rcs-co-com** "*string*"; Default is ""
**$rcs-cou-com** "*string*"; Default is ""
**$rcs-ue-com** "*string*"; Default is ""

## DESCRIPTION

RCS (Revision Control System) and SCCS (Source Code Control System) are programmers source code history data-bases. RCS introduces a system in which only one programmer can edit a source file at any one time, enforcing some form of stability in the global environment. The fact that this interface was developed for the RCS system is irrelevant, and should be usable under any other control systems such as SCCS.

When using RCS, finding a file (see **find-file**(2)) checks for the existence of the actual file. If this is not found then it checks for the existence of an RCS **$rcs-file** variable, and if present then it constructs the RCS file name and checks for its existence. If this file does not exist then it really is a new file and an new buffer is created. If the file does exist then the file is checked out using the **$rcs-co-com** which executes to create a file with the original file name, ready for loading.

**$rcs-file** is the name of the file when it is fully check in, as opposed to when it is ready to be viewed or edited. In RCS, this is usually in the `RCS` directory with an appended "`,v`", i.e. for the file `foo.c` in the `/test` directory, when fully checked in, the file will not be found at "`/test/foo.c`", but at "`/test/RCS/foo.c,v`". When testing for an RCS file, the file name is split into two parts, the path name and the file name, the path is always inserted at the start, and the file name can inserted in the rcs string by using the special "`%f`" token, thus if **$rcs-file** is set to "`RCS/%f,v`", the RCS file name is constructed from "`/test/`" + "`RCS/`" + "`foo.c`" + "`,v`".

If the RCS file is found then the **$rcs-co-com** (RCS **C**heck **O**ut **COM**mand) which is a simple system command line with the exception for `%f` which is replaced by the file name, is executed. This is expected to create the file (with the correct file name) ready for viewing.

Once a file is loaded, then the **rcs-file**(2) command has one of two effects:

> If the file is in view mode then the **$rcs-cou-com** (RCS **C**heck **O**ut **U**nlock **COM**mand) is executed (system command line using the "%f" as the file name). If the RCS file does not exist then is simply toggles the view mode, allowing editing.
>
> If the file is not in view mode MicroEmacs attempts to check the file back into RCS using either **$rcs-ci-com** (if the RCS file already exists) or the the **$rcs-cif-com** (RCS

**C**heck **I**n **F**irst **COM**mand). The "`%f`" is again used for the file name, the "`%m`" can also be used to get a comment from the user at check in time which will be inserted (without quotes) into the **$rcs-ci-com** command line. For example, one possible **$rcs-ci-com** setting is "`ci -m\"%m\" %f`" which uses the **ci**(1) program with the **-m** option to give a check in message.

If **rcs-file** is given a negative argument instead of checking in or out the current buffer's file it executes the command specified by **$rcs-ue-com** to unedit or abort any changes made to the file. After the command has been executed the file is reloaded.

## NOTES

The RCS variables are by default undefined and must be explicitly enabled in the start-up files.

## EXAMPLE

The following are typical variable definitions for the RCS interface:

```
set-variable $rcs-file     "RCS/%f,v"
set-variable $rcs-co-com   "co %f"
set-variable $rcs-cou-com  "co -l %f"
set-variable $rcs-ci-com   "ci -u -m\"%m\" %f"
```

Note that the **$rcs-cif-com** variable is usually left unassigned and **$rcs-ci-com** is used by default.
The following are typical variable definitions for the SCCS interface:

```
set-variable $rcs-file     "SCCS/s.%f"
set-variable $rcs-co-com   "sccs get %f"
set-variable $rcs-cou-com  "sccs edit %f"
set-variable $rcs-ci-com   "sccs delget -y\"%m\" %f"
set-variable $rcs-ci-com   "sccs create %f"
set-variable $rcs-ue-com   "sccs unedit %f"
```

The following variable definitions can be used for MicroSoft's Visual Source Safe:

```
set-variable $rcs-file    "%f"
set-variable $rcs-cou-com "ss.exe checkout %f"
set-variable $rcs-co-com  "ss.exe checkout %f"
set-variable $rcs-ci-com  "ss.exe checkin %f \"-c%m\""
```

The above definitions can check a file out for edit and commit changes back.

## SEE ALSO

**find-file**(2), **rcs-file**(2).

Ref: $rcs-file(5) File: m5var036.5 Date: 2022/06/12

## NAME

$recent-keys - Recent key history.

## SYNOPSIS

**$recent-keys** *string*

## DESCRIPTION

**$recent-keys** is a system variable that displays the last 100 keys entered into the system in reverse order. This variable is typically used to solve keyboard mapping problems when keys are not bound etc. allowing a visual inspection of the input into the editor.

The variable may only be set to the empty string (*""*), emptying the list.

## SEE ALSO

**buffer-bind-key**(2), **global-bind-key**(2), **list-variables**(2), **translate-key**(2).

Ref: $recent-keys(5) File: m5var077.5 Date: 2022/06/12

## NAME

$result - Various command return values

## SYNOPSIS

**$result** *returnValue*

## DESCRIPTION

**$result** is used to return the results of several commands:

**buffer-abbrev-file**(2)
: **$result** is set to the name of the current abbreviation file name.

**buffer-info**(2)
: **$result** is set to the same output string as printed to the message-line by this command.

**change-font**(2)
: **$result** is used to return the current or user select font.

**copy-file**(3)
: **$result** is used to return the number of files and directories that were successfully copied or moved and the number that failed.

**count-words**(2)
: **$result** is set to the same output string as printed to the message-line by this command.

**delete-file**(3)
: **$result** is used to return the number of files and directories that were successfully deleted and the number that failed.

**describe-key**(2)
: **$result** is set to the command bound to the given key.

**file-op**(2)
: **$result** is used to return an error code when the command fails.

**find**(3) **$result** is set to whether a match was found or a match count if option **-c** is used.

**find-registry**(2)
: **$result** is used to return the name of a registry child node given the parent and index from the user.

**generate-hash**(2)
: **$result** is set to the generated hash string.

**generate-password**(3)
: **$result** is set to the generated password.

**global-abbrev-file**(2)
: **$result** is set to the name of the current abbreviation file name.

**goto-alpha-mark**(2)
: **$result** is set to the list of all alpha marks in the current buffer when an argument of `0` is given.

**grep**(3) **$result** is set to whether a match was found or a match count if option **-c** is used.

**indent**(2)
: **$result** is set to the indent scheme used at current location when an argument of 2 is given.

**ipipe-shell-command**(2)

**pipe-shell-command**(2)

**shell-command**(2)
: **$result** is set to the exit status of the **system** call. The combination of **shell-command** calls and return value checking can be used in a variety of ways, for example, to test the existence of a file:

```
set-variable %filename @ml "Enter file name"
shell-command &cat "test -f " %filename
!if &equ $result 0
    ml-write "file exists"
!else
    ml-write "file does not exists"
!endif
```

In the case of **ipipe-shell-command**, **$result** is only set to the exit status of the **system** process during the final call to the *buffer-ipipe-command*, for more information see help for variable **$buffer-ipipe**(5).

**major-mode**(2)
: **$result** is used to return information about a given major-mode.

**mark-registry**(2)
: **$result** is used to return the full name of the given registry node.

**osd**(2)
: **$result** is used to give and return information to osd item commands, information depends on the type of **osd** item.

**osd-dialog**(3)

**osd-xdialog**(3)
: **$result** is used to return the button pressed by the user.

**replace-string**(2)

**query-replace-string**(2)

**$result** is used to return the number of substitutions, the final replace mode ('a' for all, 'l' for last or 'q' for query) and the replace with string in a list.

**search-buffer**(2)
: **$result** is set to the number of matches found when numeric argument is 0.

**set-char-mask**(2)
: **$result** is set to the characters in the given set.

**show-region**(2)
: **$result** is set to the current status of the region when an argument of *0* is given to **show-region**.

**spell**(2)

> **$result** is used to return information on the current word, the information depends on the argument given to **spell**.

@**mx**(4)

> **$result** is set to the prompt to be used during the input.

**$file-names**(5)

> **$result** is set to the absolute path of the **$file-names** query directory when the variable is set.

For more information see the help pages on referenced commands and variables.

## NOTES

The current value of **$result** is lost on the next command call which uses it. As a call to **create-callback**(2) can cause the execution of a macro to interrupt another which is waiting for user input, the value of **$result** should be copied before getting user input.

## SEE ALSO

**buffer-abbrev-file**(2), **buffer-info**(2), **change-font**(2), **copy-file**(3), **count-words**(2), **create-callback**(2), **delete-file**(3), **describe-key**(2), **file-op**(2), **find-registry**(2), **generate-hash**(2), **global-abbrev-file**(2), **goto-alpha-mark**(2), **indent**(2), **ipipe-shell-command**(2), **major-mode**(2), **mark-registry**(2), **osd**(2), **pipe-shell-command**(2), **query-replace-string**(2), **replace-string**(2), **search-buffer**(2), **set-char-mask**(2), **shell-command**(2), **show-region**(2), **spell**(2), **generate-password**(3), @**mx**(4), **$file-names**(5), **$status**(5).

Ref: $result(5) File: m5var041.5 Date: 2026/08/12

## NAME

$scroll - Screen scroll control

## SYNOPSIS

**$scroll** *scrollNum*; Default is `1`

0 <= *scrollNum* <= n

## DESCRIPTION

**$scroll** controls the horizontal and vertical scrolling method used to display long lines and buffers. The variable is split into two componants, the first nibble (`0x0f`) sets the horizontal scroll, and the second nibble (`0xf0`) sets the vertical. For the purpose of documentation these parts are kept separate, but when setting the variable a single combined value must be given.

The horizontal settings are defined as follows:

`0x00` Scroll method 0 will only scroll the current line, this is the fastest method in execution time.

`0x01` Scroll method 1 (the default) will scroll the whole page horizontally when the **scroll-left**(2) and **scroll-right**(2) commands are used. However, when the current line must be scrolled to display the cursor due to a **forward-char**(2) type cursor movement, only the current line is scrolled and the rest are reset.

`0x02` Scroll method 2 always scrolls the whole page horizontally, keeping the cursor in the current column range. If the cursor moves out of this range then all the page is scrolled to the new position. This is particularly useful when editing long lined tables.

`0x03` Scroll method 3 fixes the scroll column using the **scroll-left** and **scroll-right** functions. If the current cursor position is not visible in the column range then only the current line is scrolled to the new position.

The vertical settings are defined as follows:

`0x00` Scroll method 0 (the default) will scroll the current line to the middle of the current window whenver it is moved off screen, this is the fastest method in execution time.

`0x10` Scroll method 1 will scroll the current line to the the top of the window whenver the current line is moved off the screen using **backward-line**(2) and to the bottom of the window when **forward-line**(2) is used. This creates the effect of a smooth scroll.

## EXAMPLE

The following example sets the scrolling method to be the default horizontally (`0x01`) and smooth method (`0x10`) vertically :

```
set-variable $scroll 0x11
```

## SEE ALSO

**scroll-left**(2), **forward-line**(2), **$window-x-scroll**(5), **$window-y-scroll**(5).

Ref: $scroll(5) File: m5var027.5 Date: 2026/02/23

## NAME

$scroll-bar - Scroll bar configuration

## SYNOPSIS

**$scroll-bar** "*bitmask*"; Default is platform specific

## DESCRIPTION

**$scroll-bar** defines the configuration of the scroll bar and/or the horizontal window separator for both main text windows and **osd**(2) dialogs. The variable is interpreted as a bit mask and defines which components of the scroll bar (or separator) should be rendered in a window. The characters used to render the scroll bar or separator are defined by **$window-chars**(5). The bit mask is defined as follows:

**0x001** - Vertical Scroll Bar Width
: Bit 0 controls the width of the vertical scroll bar (or separator). A value of 0 corresponds to a single column width, a value of 1 is a double column width.

**0x002** - Upper end cap
: Bit 1 set indicates that the scroll bar has an upper end cap. This is the up arrow character at the top of a scroll bar.

**0x004** - Lower end cap
: Bit 2 set indicates that the scroll bar has a lower end cap. This is the down arrow character at the bottom of a scroll bar.

**0x008** - Corner
: Bit 3 set indicates that separate corner character is used at the intersection of the mode line and the separator.

**0x010** - Scroll Box Enable
: Bit 4 determines if the scroll bar has a scrolling box, when the bit is set each scroll bar will have a scroll box. When clear, scroll bars are rendered according to bits 0-3 & 7 only and the main area of the bar is left empty.

**0x020** - Reverse Video Box
: Bit 5 when set enables the scroll box to be rendered in reverse video, that is the background and foreground/hilight scroll colors are interchanged. This bit is typically set on X-Window platforms allowing the scroll box to comprise of `SPACE` characters allowing a solid box to be rendered in the foreground color.

  Bit 5 is only enacted if scroll boxes are enabled.

**0x040** - Horizontal Scroll Bar Width
: Bit 6 controls the width of the horizontal scroll bar, used only by **osd**(2). A value of 0 corresponds to a single column width, a value of 1 is a double column width.

**0x080** - Splitter
: Bit 7 set indicates that the scroll bar has a splitter. This is the split bar character at the top of a scroll bar.

**0x100** - Enable window Scroll Bars
: When Bit 8 is clear, scroll bars are not present on windows. If a horizontal split has been performed then the window separator is rendered plain. This is useful when performance is important, as scroll bars require constant up-date.

**0x200** - Horizontal Scroll Bar Width
Bit 9 enables scroll bars, when the bit is set each window is assigned a scroll bar in the right-hand column(s) of the window with a scroll box.

## SEE ALSO

**$mouse-pos**(5), **$scroll-bar-scheme**(5), **set-scroll-with-mouse**(2), **$window-chars**(5).

Ref: $scroll-bar(5) File: m5var055.5 Date: 2022/06/12

## NAME

$scroll-bar-scheme - Scroll bar color scheme

## SYNOPSIS

**$scroll-bar-scheme** *schemeNum*; Default is `1`

## DESCRIPTION

Sets the horizontal window scroll bar color scheme, assigning the foreground, background and selection colors which are used to render the vertical separator / scroll bars (see **add-color-scheme**(2). The separator is rendered in reverse video, i.e. the foreground color of the color scheme is used as the background color, and vice versa.

The separator is rendered in the standard colors when the associated buffer is not active, and in the current color when the buffer is active.

The scroll-bar is the window separator constructed by **split-window-horizontally**(2) or when the scroll bars are enabled via **$scroll-bar**(5).

## SEE ALSO

**$global-scheme**(5), **$ml-scheme**(5), **$mode-line-scheme**(5), **$scroll-bar**(5), **$system**(5), **$window-chars**(5), **split-window-horizontally**(2).

Ref: $scroll-bar-scheme(5) File: m5var050.5 Date: 2022/06/12

## NAME

$search-path - MicroEmacs search path

## SYNOPSIS

**$search-path** *string*

*[Microsoft Windows/MS-DOS]*
**$search-path** <*path1*>**;**<*path2*>**;**....**;**<*pathn*>

*[UNIX]*
**$search-path** <*path1*>**:**<*path2*>**:**....**:**<*pathn*>

## DESCRIPTION

**$search-path** is automatically determined by MicroEmacs when it initializes. The variable identifies the directory paths which are searched to locate editor specific files. The variable may contain multiple search paths each separated by the platform path separator (semi-colon ('**;**') on Microsoft Windows or MS-DOS environments and a colon ('**:**') on UNIX environments). Where multiple search paths are defined then they are searched in a left to right order.

**$search-path** is determined at run-time and is fixed once the editor starts. The value of the search path is determined from a number of different sources in a platform specific manner as follows:

- Built in pathnames - On UNIX platforms then preset search paths are built into the executable i.e. `/opt/jasspa`, `/usr/local/jasspa` etc.
- The users home directory, taken from the `/etc/passwd` under UNIX and from the shell environment under Microsoft Windows.
- The location of the executable, typically only used under Microsoft Windows and MS-DOS environments where the concept of standard installation locations is vague.
- The environment variable **$MEUSERPATH**(5) which identifies the preferred user directory location, overriding any home directory location specified by the system.
- The environment variable **$MEINSTALLPATH**(5) which identifies the installation directory of the standard macros overriding any built in search path(s).
- The environment variable **$MEPATH**(5) which absolutely overrides any search paths with a user defined path where **$MEUSERPATH**(5) is not defined.
- The command line option **-u** *user-name* to **me**(1) which overrides the user name found by the system.
- The environment variable **$MENAME**(5) which overrides the user name found by the system.

The search paths are generally ordered from highest priority to lowest priority and typically arranged in the order:

**$search-path=**<*user*>:<*company*>:<*me*>

where <*user*> represents the users path(s); <*company*> is the company file path(s) (e.g. company wide template files) and <*me*> path(s) which contain the standard MicroEmacs files.

This would correspond to a typical directory installation, of user **foo** on UNIX such as:

```
/usr/foo/.jassa              - User files.
/usr/local/jasspa/company    - Company wide files
/usr/local/jasspa/macros     - MicroEmacs installation directory
```

## USAGE

The current working directory is always checked first for the location of any file.

**$search-path** is used to locate all macro files, and other files located with operators such as **&find**(4).

Users should not modify this variable directly, the external aforementioned environment variables should be used to affect any desired changes.

## NOTES

The search path operation was revised in the MicroEmacs 2004 release and operates differently from earlier releases.

## SEE ALSO

**me**(1), **execute-file**(2), **&find**(4). **$home**(5), **$progname**(5), **$user-name**(5), **$user-path**(5), **$MEINSTALLPATH**(5), **$MENAME**(5), **$MEPATH**(5), **$MEUSERPATH**(5), Variable Functions.

Ref: $search-path(5) File: m5var057.5 Date: 2022/06/12

## NAME

$show-modes - Select buffer modes to display

## SYNOPSIS

**$show-modes** "*bit-string*"; Default is ""

## DESCRIPTION

**$show-modes** defines which buffer modes are displayed on the mode-line. The index into the string matches **$mode-chars**(5).

## SEE ALSO

**$user-setup**(3), **$mode-chars**(5), **$mode-line**(5).

Ref: $show-modes(5) File: m5var019.5 Date: 2022/06/12

## NAME

$show-region - Enable the hilighting of regions

## SYNOPSIS

**$show-region** *flag*; Default is `1`

## DESCRIPTION

**$show-region** enables or disables the current region hilighting, normally associated with mouse interaction in a buffer. Region hilighting occurs between the *mark* (see **set-mark**(2)) and *point* (current cursor) positions within the current buffer. An argument *n* of 0 disables region hilighting, an argument of 1 enables region hilighting between the two positions. If it is set to 3 then region hilighting will be enabled and a defined region (created using **copy-region**(2)or **yank**(2)) will continue to be hilighted until the region is changed.

A defined region can be redisplayed (if still valid) using the command **show-region**(2). The color of the region hilighting is defined by **add-color-scheme**(2) and is determined by **$buffer-scheme**(5), **$global-scheme**(5) or **$buffer-hilight**(5).

## SEE ALSO

**show-region**(2), **$buffer-hilight**(5), **$buffer-scheme**(5), **$global-scheme**(5), **$buffer-scheme**(5), **add-color-scheme**(2), **set-mark**(2).

Ref: $show-region(5) File: m5var059.5 Date: 2022/06/12

## NAME

$status - Macro command execution status

## SYNOPSIS

**$status** *boolean*

*boolean* `TRUE` (1) — `FALSE` (0)

## DESCRIPTION

**$status** contains the return status of the last command executed (`TRUE` or `FALSE`). **$status** is generally used with the !force directives in macros.

## NOTES

This variable can not be set, any attempt to set it will result in an error.

## EXAMPLE

The following example shows how the variable is used within a macro construct, it converts all tab characters to their SPACE equivalent.

```
;
; tabs-to-spaces.
; Convert all of the tabs to spaces.
define-macro tabs-to-spaces
    ; Remember line
    set-variable #l0 $window-line
    beginning-of-buffer
    !force search-forward "\t"
    !while $status
        set-variable #l1 $window-acol
        backward-delete-char
        &sub #l1 $window-acol insert-space
        !force search-forward "\t"
    !done
    goto-line #l0
    screen-update
    ml-write "[Converted tabs]"
!emacro
```

In this case **$status** monitors the search-forward command which is searching for a tab character. The command returns a status value of `TRUE` if a tab is found, otherwise `FALSE`.

The **!force** statement prevents the macro from terminating when a `FALSE` condition is detected, if omitted the macro would terminate with an error as soon as the `FALSE` status is encountered. The definition of **tabs-to-spaces**(3) can be found in format.emf.

## SEE ALSO

**execute-file**(2), **!force**(4), **$result**(5), **tabs-to-spaces**(3).

Ref: $status(5) File: m5var040.5 Date: 2022/06/12

## NAME

$system - System configuration variable

## SYNOPSIS

**$system** *bitmask*; Default is system dependent

## DESCRIPTION

The **$system** is used to define and configure the MicroEmacs environment, it is a bit based flag where:

**0x001** This bit is set if MicroEmacs is running in Console mode. On UNIX systems the default is to use X whenever possible, in which case this bit will be clear. If X is not used then a TERMCAP base interface is used instead and this bit will be set (see notes below on how to set which interface to use). On all other systems this bit will be clear.

**0x002** If this bit is set then the current system supports definable RGB colors allowing any color to be created and used in a color scheme. This bit cannot be set, typically Windows and UNIX X-Windows systems support this.

**0x004** If this bit is set then the current system supports ANSI colors (8 colors, black, red, green, yellow, blue, magenta, cyan & white), bits 0x002 and 0x004 are mutually exclusive. On UNIX systems if the TERMCAP interface is being used then this bit can be changed to (de)select the used of color. Many unix terminals do not support color so this should be set appropriately. On all other systems this bit cannot be changed and MS-DOS is currently the only other system to use ANSI colors.

**0x008** If this bit is set then the current system supports Extended ANSI colors, brighter versions of the 8 ANSI colors doubling the number of colors available to 16. On UNIX systems if the TERMCAP interface is being used then this bit can be changed to (de)select the used of bold with color to create this extended color set for foreground colors. But many UNIX terminals do not support this use of color with the bold font so this should be set appropriately. On all other systems this bit cannot be changed and MS-DOS is currently the only other system to support this.

**0x010** If this bit is set then the current system supports the use of fonts (bold, italic, light and underline). Whether these fonts can be successfully utilized depends upon the platform and the system font being used, for UNIX TERMCAP systems it will also depend on the terminal being used. This option is not supported on MS_DOS.

**0x020** When this bit is set the cursor is always shown in OSD (main menu) dialogs, this helps to improve navigation of dialogs ion systems with limited colors.

**0x080** This bit is set if the current system is a UNIX based system such as LINUX or HPUX. This bit cannot be altered, its use is within macros.

**0x100** This bit is set if the current system is a Microsoft based system such as DOS or Windows. This bit cannot be altered, its use is within macros.

**0x200** If this bit is set then the current system uses the concept of drives (i.e. `c:/` on DOS systems). This bit cannot be altered, its use is within macros.

**0x400** If this bit is set then a DOS style `8.3` file naming system should be used (i.e. "`BBBBBBBB.XXX`"), otherwise an unlimited file name length is used. This effects the backup and auto-save file names generated by MicroEmacs, the bit can be altered on systems that support unlimited file name length.

**0x800** If this bit is set then the current system supports and uses **ipipe-shell-command**(2) when required. For systems such as DOS which cannot support ipipes, this bit will be clear and cannot be altered. For systems which do support ipipes, this bit can be cleared to disable their use.

**0x1000** If this bit set, then the execution of the **tab**(2) command (bound to `tab`) always checks and adjusts the indentation of the current line when the current buffer has an indentation method. If the bit is clear then the `tab` may only checks the indentation when the cursor is in column zero depending on the setting of bit **0x200000**.

**0x2000** If this bit is set the main menu Alt hot-key bindings are enabled. These are dynamic bindings automatically generated from the main menu. Typically the first item in the main menu is "`File`" with a hot key of '**F**', with this bit set '`A-f`' will open this menu item. Note that global and local key bindings override these. Also see bit **0x4000**.

**0x4000** If this bit is set the Alt key acts as a prefix 1 modifier key. By default '`A-n`' is not bound, with this bit set the key is inferred to '`esc n`' which is bound to **forward-paragraph**. Note that global, local and menu hot-key bindings override these. Also see bit 0x2000.

**0x8000** If this bit is set the undo history is kept after a save allowing the **undo**(2) command to back-up changes beyond the last save. When clear the undo history is discarded after the buffer is saved.

**0x10000**

Enable box character rendering fix, supported on Windows and XTerm window based interfaces and UNIX terminals when UTF-8 encoding is in use (see bit **0x20000000**). Windows ANSI fonts and many XTerm ISO-8859-1 fonts do not have well formed box characters which are used by **osd**(2) and other commands to create a better looking interface. When this bit is enabled MicroEmacs traps the printing of characters with an ASCII value of less than 32 and renders them directly. Following is a table of supported characters, other characters in the range of `0x00` to `0x1f` not listed are rendered as a space:

| | |
|---|---|
| `0x01` | Special Character; Small encoded Unicode char ([u]) |
| `0x02` | Special Character; Large encoded Unicode char ([U]) |
| `0x03` | Window Character; Checkbox left side ([) |
| `0x04` | Window Character; Checkbox centre not selected ( ) |
| `0x05` | Window Character; Checkbox centre selected (X) |
| `0x06` | Window Character; Checkbox right side (]) |
| `0x07` | Special Character; Unsupported/unmappable char (<+>) |
| `0x08` | Special Character; Visible space (.) |
| `0x09` | Special Character; Visible tab (->) |
| `0x0a` | Special Character; Visible newline (<-—) |
| `0x0b` | Box Character; Bottom right (_—) |
| `0x0c` | Box Character; Top right (-—) |

| | |
|---|---|
| 0x0d | Box Character; Top left (—-) |
| 0x0e | Box Character; Bottom left (_—) |
| 0x0f | Box Character; Center cross (+) |
| 0x10 | Arrows; Right (—>) |
| 0x11 | Arrows; Left (<—) |
| 0x12 | Box Character; Horizontal line (-) |
| 0x13 | Window Character; Empty box ([ ]) |
| 0x14 | Window Character; Box with cross ([X]) |
| 0x15 | Box Character; Left Tee (—-) |
| 0x16 | Box Character; Right Tee (-—) |
| 0x17 | Box Character; Bottom Tee (_—_) |
| 0x18 | Box Character; Top Tee (T) |
| 0x19 | Box Character; Vertical Line (—) |
| 0x1a | Window Character; Bottom right with resize (_—) |
| 0x1b | Window Character; Vertical scroll box (=) |
| 0x1d | Window Character; Horizontal scroll box (———) |
| 0x1e | Arrows; Up (/\) |
| 0x1f | Arrows; Down (\/) |

**0x20000**

Enables the client server, default is disabled (UNIX and Win32 NT or Win95+ platforms only). When enabled a hidden "`*server*`" buffer is created which monitors commands written to the server, the socket "`/tmp/mesrv`**uid**" on UNIX systems and the command input file "**$TEMP**`/me`**$MENAME**`.cmd`" on Win32 systems. Commands can be written out using the command **ipipe-write**(2) while in the "*server*" buffer, the command is written to the same socket on UNIX systems and to the response file and response file "**$TEMP**`/me`**$MENAME**`.rsp`" on Win32 systems. This functionality is used by the **-m** and **-o** command-line options.

**0x40000**

Enables the capture of the Alt space key ("`A-space`"), default is enabled (Win32 platform only). In the Windows environment the Alt Space key is used to activate the main window's pull down menu at the top left. if this bit is set MicroEmacs captures this key and executes it as normal, thereby disabling this standard windows binding.

**0x80000**

Enables the drawing of visible white spaces, i.e. space, tab and new-line characters. When disabled (default) white spaces are drawn using spaces (' ') which means the user cannot distinguish between a tab and spaces or determine the last character of the line by merely looking at the display. When enabled MicroEmacs uses visible characters to draw the white spaces, the characters used are set with the variable **$window-chars**(5).

**0x100000**

Enables hiding MicroEmacs generated backup files. On Windows and Dos platforms the Hidden file attribute is used to hide the file, whereas on UNIX the backup file name is prepended with a '**.**'.

**0x200000**

If this bit set, then the execution of the **tab**(2) command (bound to `tab`) checks and adjusts the indentation of the current line when the cursor is in column zero and current buffer has an indentation method. The setting of this bit has no effect if bit **0x1000** is set. If this and bit **0x1000** are clear then the `tab` will not check the indentation.

**0x400000**

When this bit is set the external clipboard (Windows & XTerm platforms) will never be set to empty, if the current yank buffer is the empty string the cut buffer will be set to a space (i.e. " "). This feature has been added to avoid problems with other software (e.g. **exceed**(1) which can crash if given an empty cut buffer).

**0x800000**

When this bit is set all use of the external clipboard (Windows & XTerm platforms) is disabled, this means that MicroEmacs will not attempt to retrieve or set the content of the system clipboard.

**0x1000000**

When this bit is set the `-p` or `-P` option was used on the command line (see **me**(1)). This bit can be used by macros to determine how to function, see **filetype**(3f) for an example of its use.

**0x2000000**

When this bit is set **ipipe-kill**(2) uses `SIGTERM`, rather than `SIGKILL`, by default to terminal the process (UNIX platforms only).

**0x4000000**

When this bit is set on a UNIX X-Windows system MicroEmacs will retrieve and sync the kill-buffer with the X-Windows Primary selection buffer rather than the Clipboard selection. Note that this bit is ignore if the use of the external clipboard is disabled, see bit **0x800000**.

**0x8000000**

When this bit is set on a UNIX X-Windows system MicroEmacs will set both the X-Windows Primary and the Clipboard selection buffers when the kill-buffer is changed. When not set only one buffer is set depending on the setting of bit **0x4000000**. Note that this bit is ignore if the use of the external clipboard is disabled, see bit **0x800000**.

**0x10000000**

When this bit is set on a UNIX Console system the use of the alternative screen buffer is disabled, see command-line **-A** option.

**0x20000000**

When this bit is set MicroEmacs UTF-8 encodes all terminal output and UTF-8 decodes all terminal input, this bit is currently only used when MicroEmacs is running in a UNIX terminal, other interfaces such as XTerm and Windows handle Unicode I/O differently. When UTF-8 encoding is being used, bit **0x10000** is also supported in the terminal as Unicode drawing characters can be used.

## EXAMPLE

The follow example works out the current buffer's backup file name using **$system** to determine the naming system being used by MicroEmacs:

```
set-variable #l0 &stat "a" $buffer-fname
; Is an 8.3 dos style naming system being used?
```

```
!if &band $system 0x400
    !if &not &set #l1 &sin "." #l0
        set-variable #l1 &cat #l0 ".~~~"
    !elif &gre &set #l1 &sub &len #l0 #l1 2
        set-variable #l1 &cat &lef #l0 &sub &len #l0 1 "~"
    !else
        set-variable #l1 &spr "%s%n" #l0 &sub 3 #l1 "~"
    !endif
!elif $kept-versions
    set-variable #l1 &cat #l0 ".~0~"
!else
    set-variable #l1 &cat #l0 "~"
!endif
```

The following macro can be used to toggle the visible drawing of white spaces:

```
define-macro toggle-visible-white-spaces
    set-variable $system &bxor $system 0x80000
    screen-update
!emacro
```

### NOTES

Most of the **$system** functionality can be set using the **$user-setup**(3) dialog.

### UNIX X verses Termcap

By default, on X supporting systems MicroEmacs creates a new X window. This feature may be disabled in one of two ways:

- The environment variable `$TERM` is set to "`vt...`", in this case it is assumed that the machine is a server, and the host cannot support X.
- The `-n` option is used on the command line (see **me**(1)) to disable the windowing interface.

If X is disabled then the **termcap** interface is used instead, still allowing the use of colors through the ANSI standard, or the use of fonts (see bits **0x004** and **0x008**).

X provides the following features over and above a **termcap** based version of MicroEmacs:

- R,G,B style color creator giving access to up to 256 different colors for the ultimate hilighting schemes (see bit **0x002** and **add-color**(2)).
- Full mouse support, allowing user definable bindings to every mouse event (see **global-bind-key**(2)).
- Copy from and pasting to X's selection buffer (see **yank**(2)).

### SEE ALSO

**user-setup**(3), **$mouse**(5), **$platform**(5), **add-color**(2), **add-color-scheme**(2), **ipipe-shell-command**(2), **$global-scheme**(5).

Ref: $system(5) File: m5var047.5 Date: 2025/09/19

## NAME

$tab-width - Global tab character width

## SYNOPSIS

**$tab-width** *integer*; Default is 8

0 < *integer* <= *n*

## DESCRIPTION

**$tab-width** defines the default width of a tab character specified in spaces. The default value is used when new files are loaded and created and seeds the variable **$buffer-tab-width**(5).

Setting tabs to arbitrary widths is possible in MicroEmacs but you must be aware of a subtle difference that it makes to your file and hence to your editing. When you start MicroEmacs, the tab width is set to the default (usually every 8th column) for the tab character (CTRL-I). As long as you stay with the default, every time you insert the tab character, a CTRL-I get inserted. Hence, you logically have a single character which might appear to be several spaces on the screen (or the output) depending upon the column location of the tab character. This means that to remove the spacing you have to delete a *single* character – the tab character.

On the other hand, the moment you explicitly set the tab interval (even if it is to the default value), MicroEmacs handles the tab character by expanding the character into the required number of spaces to move you to the appropriate column. In this case, to remove the spacing you have to delete the appropriate number of spaces inserted by M-e to get you to the right column.

The operating mode of the tab expansion is controlled by the **tab**(2m) mode.

## NOTES

The November 2004 version of MicroEmacs removed the existing variables **$tabwidth** and **$tabsize** which were ambiguous. The existing variable scope was global which caused problems when moving to buffers with different tab width properties. The variables have been replaced with **$tab-width** and **$buffer-tab-width**(5). The indentation which was previously controlled by the tab width has been replaced with **$indent-width**(5) and **$buffer-indent-width**(5).

## SEE ALSO

**buffer-mode**(2), **tab**(2m), **$buffer-indent-width**(5), **$buffer-tab-width**(5), **$indent-width**(5).

Ref: $tab-width(5) File: m5var042.5 Date: 2026/06/19

## NAME

$temp-name - Temporary file name

## SYNOPSIS

**$temp-name** *FileName*

## DESCRIPTION

**$temp-names** is automatically set to a nonexistent file name in the systems temporary file directory. On UNIX systems the temporary directory is fixed to "`/tmp/`", on other systems the temporary directory is set by the **$TEMP** environment variable.

## EXAMPLE

The following example uuencodes a given file into a temporary file and then inserts this file into the current buffer.

```
set-variable #l0 @ml04 "Uuencode and insert file"
set-variable #l1 $temp-name
!force shell-command &spr "uuencode %s < %s > %s" #l0 #l0 #l1
insert-file #l1
!force shell-command &cat "rm " #l1
```

## NOTES

This variable can not be set, any attempt to set it will result in an error.

The returned file name is not guaranteed to be unique between calls, only that the file does not currently exist.

## SEE ALSO

**shell-command**(2), **file-op**(2).

Ref: $temp-name(5) File: m5var069.5 Date: 2022/06/12

## NAME

$time - The current system time

## SYNOPSIS

**$time** "*string*"
**$unix-time** *integer*"

## DESCRIPTION

**$time** is a constantly changing variable which is set to the current system time. The format of **$time** is "`YYYYCCCMMDDWhhmmssSSS`", where:

**YYYY** The current year (full 4 digits so should be millennium bug free).

**CCC** Day of the year (0-366).

**MM** The month of the year (1-12).

**DD** The day of the month (1-31).

**W** The day of the week (0-6 Sunday=0).

**hh** The hour (0-23).

**mm** The minute (0-59).

**ss** The second (0-59).

**SSS** The millisecond (0-999).

**$time** can be set to an integer value which is a time offset in seconds, for example if the following was executed;-

```
set-variable $time "3600"
ml-write &cat "$time is " $time
set-variable $time "0"
```

The written time would one hour ahead of the system time.

**$unix-time** is also a constantly changing variable, however this variable cannot be set. Its value is a count of the number of seconds since 1970-01-01 00:00:00 UTC, otherwise known as the UNIX epoch time, presented as a floating point number with a guaranteed 9 decimal places so its length is 20 characters. The focus of the fraction part is more on providing a sub-second timer at the highest resolution the system can realistically provide, however, the actual resolution is platform specific and should only be considered accurate to 10ms at best, see example section below.

## EXAMPLE

The following macro uses **$time** to calculate the time taken to execute a user command:

```
define-macro time
    !force set-variable #l2 @1
    !iif &not $status  set-variable #l2 @ml00 "Time command"
    set-variable #l0 $time
    !force execute-line #l2
```

```
    set-variable #l1 $time
    set-variable #l2 &add &mid #l0 16 2 &mul 60 &add &mid #l0 14 2 &mul 60 &mid #l0 12 2
    set-variable #l3 &add &mid #l1 16 2 &mul 60 &add &mid #l1 14 2 &mul 60 &mid #l1 12 2
    !if &les &set #l4 &sub &rig #l1 18 &rig #l0 18 0
        set-variable #l2 &add #l2 1
        set-variable #l4 &add 1000 #l4
    !endif
    ml-write &spr "Command took %d.%03d sec" &sub #l3 #l2 #l4
!emacro
```

The following implementation uses **$unix-time** and **&fsub**(4):

```
define-macro time
  !force set-variable #l2 @1
  !iif &not $status  set-variable #l2 @ml00 "Time command"
  set-variable #l0 $unix-time
  !force execute-line #l2
  set-variable #l1 $unix-time
  ml-write &spr "Command took %.6f sec" &fsub #l1 #l0
!emacro
```

This implementation of **$unix-time** suggests a nanosecond accuracy, however this is unrealistic due to the granularity/limitations of the system clock, a more realistic level would be around 1 microsecond. However, at this level the variability of the CPU task scheduler, the time taken to get the time and the algorithms used can all degrade the actual accuracy of the measure. For example, the above macro code use floating-point maths (`&fsub`), Double precision floats, as used by MicroEmacs, only provide around 15 to 16 significant figures of accuracy. Consider the following:

```
set-variable #g1 &fsub "1234567890.000010000" "1234567890.000000000"
set-variable #g2 &fsub "1234567890.000001000" "1234567890.000000000"
set-variable #g3 &fsub "1234567890.000000100" "1234567890.000000000"
```

The answer given for `#g1` typically has an error of around 0.1%, an error of 5% for `#g2` and an answer of 0 for `#g3`. This means if simple floating point maths is used the maximum accuracy is 1 microsecond +/- 5%. Attempting to measure performance down to this level is generally a waste of time, it would be far better to increase the length of the task (i.e. run 1000 times and divide the total time by 1000).

## SEE ALSO

**time**(3), **&fsub**(4).

Ref: $time(5) File: m5var060.5 Date: 2025/09/06

## NAME

$timestamp - Time stamp string

## SYNOPSIS

**$timestamp** "*string*"; Default is "<%Y%M%D.%h%m>"

## DESCRIPTION

**$timestamp** defines the file time-stamping string. MicroEmacs searches for, and modifies, the string to the current time and date whenever the file is saved (written to disk) and **time**(2m) mode is enabled.

Time stamp string is defined, by default, as "<%Y%M%D.%h%m>". The first occurrence of the string in the file is up-dated with the time and date information when the buffer is written. The **$timestamp** string may contain any text, and includes the following, magic characters escaped by a percentage ('%') character:

- `D` - Day.
- `M` - Month.
- `Y` - Year.
- `h` - Hour.
- `m` - Minute.
- `s` - Second.

The format string may be redefined into any format. The '%' character has to be delimited by another '%' if it is to be used in the text (i.e. "%%").

The year component (`%Y`) may be a 2 or 4 digit string, depending whether it includes the century. When the time stamping searches for the `%Y` component it searches for either variant and replaces appropriately.

## EXAMPLE

The startup file may define the time stamp required as follows:

```
set-variable $timestamp "Last Modified : %Y/%M/%D %h:%m:%s"
```

Time stamping is performed on the string :

```
Last Modified : 90/11/23 10:12:01
```

Where the time stamp is modified according to the file (buffer) type then the time stamp string may be modified within the buffer hooks. This allows different files to utilize different time stamping strings. The following example shows how the entry and exit buffer hooks are defined to modify the string:

```
0 define-macro bhook-nroff
    set-variable .timestamp $timestamp
    ; Buffer specific time stamp string.
    set-variable $timestamp "[%Y/%M/%D %h:%m:%s]"
!emacro
0 define-macro ehook-nroff
    ; Restore the existing time stamp.
    set-variable $timestamp .bhook-nroff.timestamp
!emacro
```

On entry to the buffer (buffer becomes current) the buffer hook **bhook-nroff** is executed which stores the current setting and then modifies the time stamp string. On exit from the buffer the buffer hook **ehook-nroff** is executed restoring the time stamp string.

## SEE ALSO

**buffer-mode**(2) **time**(2m).

Ref: $timestamp(5) File: m5var045.5 Date: 2022/06/12

## NAME

$trunc-scheme - Truncation color scheme.

## SYNOPSIS

**$trunc-scheme** *schemeNum*; Default is 0

## DESCRIPTION

**$trunc-scheme** sets the color scheme used when drawing a line truncation indicator. The left truncation character (usually a '$' char) drawn at the start of the line indicates that the line has been scrolled to the right and therefore the start of the line has been truncated. A right truncation char (also usually a '$') drawn at the end of the line indicates the remainder of the line is too long to fit onto the width of the window so the end has been truncated and the indicator drawn.

The *schemeNum* selected must be a color scheme defined with **add-color-scheme**(2), which identifies the foreground and background color schemes. A hilight scheme can define its own truncation color scheme, see **hilight**(2) for more information.

## NOTES

The truncation characters used are set by the **$window-chars**(5) variable.

## SEE ALSO

**$buffer-scheme**(5), **$global-scheme**(5), **add-color-scheme**(2), **hilight**(2), **$window-chars**(5).

## NAME

$user-name - MicroEmacs user name

## SYNOPSIS

**$user-name**

## DESCRIPTION

**$user-name** is the name of the MicroEmacs user. The name is used for identifying user configuration files such as *$user-name*.**emf** and *$user-name*.**erf**, the start-up user macro file and **erf**(8) registry file that contains saved user preferences.

The **$user-name** is determined as follows, in order of precedence:

- The **me**(1) command line option **-u** *username*.
- The environment variable **MENAME**(5) (alternatively via `$LOGNAME` on Microsoft Windows platforms ONLY).
- The name determined from the system. On UNIX then the `/etc/passwd` entry is used, on MicroSoft Windows then the login name is used.
- When no other name is located then `user` is used (MS-DOS and Microsoft Windows environments running without a login mechanism).

For most conventional *login* systems then the correct login name is retrieved and it is not necessary to over-ride the name, unless a special configuration of MicroEmacs is being used.

## NOTES

The user should not attempt to write to **$user-name**, the variable should be controlled through the aforementioned techniques.

**$user-name** was first introduced into MicroEmacs in the 2004 release.

## SEE ALSO

**me**(1), **$home**(5), **$search-path**(5), **$user-path**(5), **$MENAME**(5), **emf**(8), **erf**(8).

Ref: $user-name(5) File: m5var088.5 Date: 2022/06/12

## NAME

$user-path - MicroEmacs user directory

## SYNOPSIS

**$user-path**

## DESCRIPTION

**$user-path** is the name of the MicroEmacs user directory where user configuration files such as $user-name**.emf**, $user-name**.erf** and personal spelling dictionaries are stored. This is a single directory given as an absolute path and always with a terminating directory separator character '`/`'.

The value of the variable is the first suitable directory located using the following rules in order:

- The environment variable **$MEUSERPATH**(5) is defined, **$user-path** is set to this value. If the environment **$MEUSERPATH** is defined without a trailing '`/`' then it is automatically added. Note that the directory does not have to exist, but unexpected errors may occur if it does not.
- The environment variable **$MEPATH**(5) is defined, **$user-path** is set to the first directory within the **$MEPATH** variable. Again, this path may not exist but may cause unexpected problems if it doesn't.
- The environment variable **$MEINSTALLPATH**(5) is defined then $user-name`/` is appended to each directory given within the **$MEINSTALLPATH** and if the resultant directory exists, **$user-path** is set to this location. In this case, the user path must exist for it to be used.
- A system dependent user configuration/application data directory is checked for and used if it exists, this directory is defined as follows:
  **UNIX**: `.config/jasspa/$user-name/` is appended to the user's home directory, as defined in the `/etc/passwd` entry, i.e. `$HOME/.config/jasspa/$user-name/`, see variable **$home**(5).
  **Windows**: `jasspa/$user-name/` is appended to the user's application data directory, i.e. `$APPDATA/jasspa/$user-name/`.
  **MS-DOS**: `jasspa/$user-name/` is appended to the **$home**(5) directory, typically `C:/jasspa/$user-name/`.
- If the executable is located within a `bin/` or `bin/<plaform>/` sub-directory, the sub-directory is removed and **$user-name/** appended, if the resultant directory exists, **$user-path** is set to this location. In this case, the user path must exist for it to be used.
- If no configuration paths, i.e. `macros/` directory, have been located by this point, then any built in search paths may be use in the same way as **$MEINSTALLPATH** above, currently only UNIX is built with a single search path of `/usr/local/share/jasspa`.

If a suitable path was not located **$user-path** is set to a special value of `tfs://new-user/` and MicroEmacs runs a **new-user** command to setup the environment for you. This special path does not exist and is not writeable, ensuring that MicroEmacs does not create user configuration

files in unexpected places. While set to this, macros such as **user-setup**(3) will warn you that any configuration changes cannot be saved.

It is therefore strongly recommended you run this **new-user** command, when prompted, so that all your settings can be located somewhere that is easy for MicroEmacs to find next time.

## NOTES

The user should not attempt to write to **$user-path**, the variable should be controlled through the aforementioned techniques.

## SEE ALSO

**me**(1), **$home**(5), **$progname**(5) **$search-path**(5), **$user-name**(5), **$MEINSTALLPATH**(5), **$MENAME**(5), **$MEPATH**(5), **$MEUSERPATH**(5), **user-setup**(3), **emf**(8), **erf**(8).

Ref: $user-path(5) File: m5var089.5 Date: 2025/04/06

## NAME

$variable-names - Filtered variable name list

## SYNOPSIS

**$variable-names** *VariableName*

## DESCRIPTION

**$variable-names** must first be initialized to the required filter string, if the variable is evaluated before it is initialized the value will be set to "*ABORT*" and the command will fail.

The filter string can contain wild-card characters compatible with most file systems, namely:

**?** Match any character.

**[abc]** Match character only if it is *a*, *b* or *c*.

**[a-d]** Match character only if it is *a*, *b*, *c* or *d*.

**[^abc]** Match character only if it is not *a*, *b* or *c*.

***** Match any number of characters.

Note that these are not the same characters used by **exact**(2m) mode.

Once initialized, evaluating **$variable-names** returns the name of the next variable which matches the filter until no more variables are found, in which case an empty string is returned.

## EXAMPLE

The following example prints out the name of all variables to the massage line one at a time. Note that **&set**(4) is used on the **!while**(4) statement to avoid evaluating **$variable-names** twice per loop.

```
set-variable $variable-names "*"
!while &not &seq &set #l0 $variable-names ""
    100 ml-write &cat "variable: " #l0
!done
```

## NOTES

The list of variables is evaluated when the variable is initialized, variables defined after the initialization will not be included in the list. The list can contain the current buffer's buffer variables (See **Variables**(4) for more information on the different types of variables).

Using **unset-variable**(2) to delete a variable which are in the list, before it has be evaluated, will have undefined effects.

## SEE ALSO

**list-variables**(2), **$command-names**(5).

Ref: $variable-names(5) File: m5var068.5 Date: 2022/06/12

## NAME

$version - MicroEmacs version date-code
$build - MicroEmacs build information

## SYNOPSIS

**$version** "*YYYYMMDD*"
**$build** "*<platform> <platform-mjr-ver> <arch> <bit-size> <toolset> <toolset-mjr-ver>* "

## DESCRIPTION

**$version** is a system variable which is defined as the MicroEmacs build date code. This value is fixed at compile time and cannot be changed. The variable may be used in macros to identify incompatibility issues.

**$build** is a MicroEmacs list variable detailing the platform, architecture and toolset (compiler) used to build the running version. This variable is used to locate compatible binary components such as upgrades or **OpenSSL**(1) dynamic libraries.

## EXAMPLE

Given a macro that only operates with a MicroEmacs executable built on or after 1st August 2001 then this macro should check that `$version` is not less than `20010801`. The check may be performed as follows:

```
!if &les $version "20010801"
    ml-write "[Error: MicroEmacs executable is incompatible]"
    !abort
!endif
```

## NOTES

The **$version** variable was introduced in 2001-08-01, evaluating this variable on an earlier version of MicroEmacs would return the string "`ERROR`" unless an environment variable $version has been defined. "`ERROR`" evaluates to `0` hence the test still operates correctly.

This variable is used in the macro file `me.emf` to check for any macro - executable incompatibility issues.

The **$build** variable was introduced in 2024-07-01 along with the find-file pseudo key which together can be used to download and install missing or out-of-date components.

## NAME

$window-chars - Character set used to render the windows

## SYNOPSIS

**$window-chars** "*sting*"; Default is "=-#*%=^|#|v*==^^||##||vv**|<-#->*||<<--##-->>**x*[]>\.$$\"

## DESCRIPTION

**$window-chars** is a fixed length string that defines the set of characters used to render the windows. The characters have fixed indices defined as follows:

Index 0 The active window mode line separator character, This replaces all *Index 1* characters when the window is current. Default is '**=**'.

Index 1 The inactive window mode line separator character. This character is replaced by *Index 0* characters when the window becomes current. Default is '**-**'.

Index 2 UNIX based platforms only. The **root** or **superuser** indicator character that appears on the mode line. Default is '**#**'.

Index 3 The buffer changed indicator character that appears on the mode line. Default is '*'.

Index 4 The buffer in **view**(2m) mode indicator character that appears in the mode line. Default is '**%**'.

Index 5 Single column vertical scroll bar split window horizontally character. Default is '**=**'.

Index 6 Single column vertical scroll bar up-arrow character. Default is '^'.

Index 7 Single column vertical scroll bar upper-shaft character. Default is '—'.

Index 8 Single column vertical scroll box character. Default is '**#**'.

Index 9 Single column vertical scroll bar lower-shaft character. Default is '—'.

Index 10
Single column vertical scroll bar down-arrow character. Default is '**v**'.

Index 11
Single column vertical scroll bar corner character. Default is '*'.

Index 12-13
Double column vertical scroll bar split window horizontally character. Default is '**==**'.

Index 14-15
Double column vertical scroll bar up-arrow characters. Default is "^".

Index 16-17
Double column vertical scroll bar upper-shaft characters. Default is "——".

Index 18-19
Double column vertical scroll box characters. Default is "**##**".

Index 20-21
Double column vertical scroll bar lower-shaft characters. Default is "——".

Index 22-23
: Double column vertical scroll bar down-arrow characters. Default is "**vv**".

Index 24-25
: Double column vertical scroll bar corner characters. Default is "**".

Index 26-32
: Single column horizontal scroll bar. Default is "—<-#->*".

Index 33-46
: Double column horizontal scroll bar. Default is "——<<–##–>>**".

Index 47
: Osd title bar blank character. Default is ' '.

Index 48
: Osd title bar right corner kill character. Default is '**x**'.

Index 49
: Osd dialog bottom right corner resize character. Default is '*'.

Index 50-53
: Osd button characters. 50 & 52 are used for a normal button, default is ' *Button* '. 51 & 53 are used for the current default button, default is '>*Button*<'.

Index 54
: Displayed tab character (used when **$system**(5) bit 0x80000 is set). Default is '>'.

Index 55
: Displayed new-line character (used when **$system**(5) bit 0x80000 is set). Default is '\'.

Index 56
: Displayed space character (used when **$system**(5) bit 0x80000 is set). Default is '**.**'.

Index 57
: Displayed truncated text to left character (used when the current line is scrolled to the right). Default is '**$**'.

Index 58
: Displayed truncated text to right character (used when the current line is longer than the window width). Default is '**$**'.

Index 59
: Inserted end of wrapped line character in an **ipipe-shell-command**(2) buffer. Default is '\'.

## EXAMPLE

The **$window-chars** is typically platform dependent, it's setting is determined by the characters available in character set of the hosting platform. MS-DOS and Microsoft Windows use an OEM font might use the following value:

```
"=-#*%=\C^\xB1 \xB1\C_\CD==\C^\C^\xB1\xB1  \xB1\xB1\C_\C_\C[
\CZ|\CQ\xB1 \xB1\CP\CD||\CQ\CQ\xB1\xB1  \xB1\xB1\CP\CP\C[
\CZ x*  >\\.$$\\"
```

This utilizes character-set specific characters to render some of the window components.

## NOTES

- **$scroll-bar**(5) allows the scroll box to be rendered in reverse video allowing a space to be used for the scroll box.
- Use **insert-symbol**(3) to determine the displayable characters on the host platform.
- The use of MicroEmacs's extended character set on Windows and XTerm platforms can greatly improve the look and usability of MicroEmacs, see the Extend Char Set option in the Platform page of **user-setup**(3) and bit `0x10000` of variable **$system**(5).

## SEE ALSO

**split-window-horizontally**(2), **insert-symbol**(3), **$box-chars**(5), **$global-scheme**(5), **$mode-line**(5), **$mode-line-scheme**(5), **$scroll-bar**(5), **$scroll-bar-scheme**(5), **$system**(5).

Ref: $window-chars(5) File: m5var051.5 Date: 2022/06/12

## NAME

$window-col - Window cursor column (no expansion)
$window-line - Window cursor line (with narrows)
$window-acol - Window cursor actual column (expansion)
$window-aline - Window cursor actual line (ignore narrows)
$window-eline - Window end line
$window-mark-col - Window mark column
$window-mark-line - Window mark line

## SYNOPSIS

**$window-col** *integer* [0 <= *integer* <= 65535]
**$window-line** *integer* [1 <= *integer* <= n]
**$window-acol** *integer* [0 <= *integer* <= n]
**$window-aline** *integer* [1 <= *integer* <= n]
**$window-eline** *integer* [1 <= *integer* <= n]
**$window-mark-col** *integer* [0 <= *integer* <= 65535]
**$window-mark-line** *integer* [1 <= *integer* <= n]

## DESCRIPTION

**$window-col** is defined as the current position of the cursor in the current line in the current window. Column zero is the left hand edge. This differs from **$window-acol** in that TAB and special characters only count for 1 character. **$window-col** is valid in the range 0 - *n*.

**$window-acol** is defined as the current column of the cursor in the current window. Column zero is the left hand edge. This differs from **$window-col** in that tab and special characters may not count for 1 character.

**$window-line** is defined as the current buffer line number the cursor is on in the current window. Line numbering starts from 1. **$window-line** is valid in the range 1 - *n*.

**$window-aline** is identical to **$window-line** except when the current buffer contains narrowed out sections before the current line. In this case **$window-line** will be set to the line number without counting the number of lines in the narrow, whereas **$window-aline** will return the current line number including all lines narrowed out before it. When this variable is set, the line required may lie in a narrowed out section in which case the narrow is automatically removed. See **narrow-buffer**(2) for more information on narrowing.

**$window-eline** is the end line of the window, that is the line number of the last line of the buffer showing in the window. This is a useful value for mouse operations when the position of the mouse may be tested to ensure that the mouse position is within the valid text region of the buffer.

**$window-mark-line** is the current buffer line number of the mark position of **set-mark**(2) in the current window. Line numbering starts from 1. **$window-mark-line** is valid in the range 1 - *n*.

**$window-mark-col** is the current buffer column of the mark position of **set-mark**(2) in the current window. Note that this is the actual column position in the line which may be different from the window column position as TAB and special characters only count as a single character.

## NOTES

Variable **$window-wcol** was renamed to **$window-acol** in June 2000. Variable **$window-wline** was also removed and a new variable **$window-y-scroll** introduced at this time. The following macro code can be used to calculate the value of the original **$window-wline** variable: **$window-mark-line** and **$window-mark-col** first appeared in the 2006 release of MicroEmacs.

```
&sub &sub $window-line $window-y-scroll 1
```

## SEE ALSO

**$frame-depth**(5), **$window-depth**(5), **$window-width**(5), **$window-y-scroll**(5), **narrow-buffer**(2).

## NAME

$window-depth - Number of text lines in a window
$window-width - Number of character columns in a window

## SYNOPSIS

**$window-depth** *integer*

1 <= *integer* <= $frame-depth

**$window-width** *integer*

0 <= *integer* <= $frame-width - 1

## DESCRIPTION

**$window-depth** returns the depth (height) of the current window, excluding the mode line, specified in text lines. (i.e. the number of lines of text in the window). The returned value is an integer in the range:

**0** - ( $frame-depth - **3** )

**$window-width** returns the width, in characters, of the current window. The returned value is an integer in the range:

**0** - $frame-width.

## NOTES

These variables can not be set, any attempt to set them results in an error.

## SEE ALSO

**$frame-depth**(5), **$frame-width**(5), **$window-scroll-bar**(5), **$window-mode-line**(5),

## NAME

$window-flags - Current window setup flags

## SYNOPSIS

**$window-flags** *bitmask*; Default is `0`

## DESCRIPTION

The **$window-flags** variable is used to set or get various behavioural characteristic settings of the current window, it is a bit based flag where:

**0x001** If set the width of the window is locked, calls to **resize-all-windows**(2) will maintained the width of this window whenever possible.

**0x002** If set the depth of the window is locked, calls to **resize-all-windows**(2) will maintained the depth of this window whenever possible.

**0x004** If set the buffer being displayed by the window is locked, the user can still manually change the buffer being displayed (by using commands like **find-buffer**(2)) but commands that pop-up buffers (such as **help**(2) or **find-tag**(2)) will not use this window.

**0x008** When set the command **compare-windows**(2) will ignore this window.

**0x010** When set the commands like **previous-window**(2) and **next-window**(2) will skip this window unless the numeric argument given to the command is used to override the flag setting.

**0x020** When set the command **delete-other-windows**(2) will not delete this window unless the numeric argument given to the command is used to override the flag setting.

**0x040** When set the command **delete-window**(2) will not delete this window unless the numeric argument given to the command is used to override the flag setting.

**0x080** When set the window cannot be split using either the **split-window-horizontally**(2) or **split-window-vertically**(2) commands.

**0x100** If not set the window cannot be deleted if it is the only window without this bit set. This more esoteric feature is utilized by the toolbar, all toolbar windows have this bit set which means that the main user window cannot be delete.

## NOTES

The $window-flags setting is not preserved during a window splitting operation (i.e. using a command like **split-window-vertically**(2)) as the persistence of these settings can lead to unexpected behaviour.

The toolbar uses bit 0x1000 to indicate that the window is displaying a toolbar tool, this bit should not be used by users and its value should be maintained.

## SEE ALSO

**next-window**(2), **delete-other-windows**(2), **compare-windows**(2).

Ref: $window-flags(5) File: m5var084.5 Date: 2022/06/12

## NAME

$window-id - Current window identity
$window-id - Previous window identity

## SYNOPSIS

**$window-id** *integer*
**$window-pid** *integer*

## DESCRIPTION

**$window-id** retrieves the identity of the current window, this is a integer value (1..n) that is automatically allocated when a window is created using **split-window-horizontally**(2) or **split-window-vertically**(2).

The **$window-id** is guaranteed to be unique and is typically used in macros to locate a previous window.

The **$window-pid** stores the **$window-id** of the previous current window, defaulting to 0 if no other window has been visited. Note that the window identified by **$window-pid** may no longer exist so any macro making use of this variable must handle the *failed to find* case.

## SEE ALSO

**next-window**(2), **split-window-horizontally**(2), **split-window-vertically**(2), **$frame-id**(5).

Ref: $window-id(5) File: m5var087.5 Date: 2026/08/12

## NAME

$window-mode-line - Window mode line position
$window-scroll-bar - Window scroll bar (or separator) position

## SYNOPSIS

**$window-mode-line** *integer*

1 <= *integer* <= $frame-depth - 2

**$window-scroll-bar** *integer*

0 <= *integer* <= $frame-width - 1

## DESCRIPTION

**$window-mode-line** stores the screen line of the current windows mode-line, where screen lines are counted from 0 at the top of the screen. Often used in conjunction with **set-cursor-to-mouse**(2) and **$mouse-y**(5) to add more complex mouse functionality.

**$window-scroll-bar** stores the screen position of the right-hand horizontal window separator line or scroll-bar (see **split-window-horizontally**(2) and **$scroll-bar**(5)). A value of greater than **$frame-width**(5) indicates that there is no right-hand separator column or scroll bar present. Often used in conjunction with **$mouse-x**(5).

## EXAMPLE

In the following example the position of the mouse is checked to see if it is on the mode line of the window, if so then a different action is taken.

```
set-cursor-to-mouse
;   If we are on the mode line then interpret position of
;   the cursor on line to control the screen.
!if &equal $window-mode-line $mouse-y
    !if &less $mouse-x "2"
        menu-main     ; Inform buffer to pop up menu.
    !elif &equal $mouse-x "2"
        delete-window
    !elif &equal $mouse-x "3"
        delete-other-windows
    !elif &equal $mouse-x "4"
        backward-page
    !elif &equal $mouse-x "5"
        forward-page
    !elif &equal $mouse-x "6"
        recenter
    !elif &equal $mouse-x "7"
        undo
    !endif
!else
    .....
!endif
```

## SEE ALSO

**$mode-line**(5), **$mouse-x**(5), **$mouse-y**(5), **$scroll-bar**(5), **$mouse-pos**(5), **set-cursor-to-mouse**(2), **split-window-horizontally**(2).

Ref: $window-mode-line(5) File: m5var046.5 Date: 2022/06/12

## NAME

$window-x-sroll - Current window X scroll
$window-xcl-sroll - Current window current line X scroll
$window-y-sroll - Current window Y scroll

## SYNOPSIS

**$window-x-sroll** *integer*
**$window-xcl-sroll** *integer*
0 <= *integer* <= 65535
**$window-y-sroll** *integer*
0 <= *integer* <= n

## DESCRIPTION

**$window-x-sroll** defines the horizontal scroll position in the current window for all lines except the current line, **$window-xcl-sroll** defines the scroll position for the current line. The variables set how many characters are scrolled off the left hand edge of the current window, the variables are indirectly set by commands such as **scroll-left**(2), **forward-char**(2) etc.

**$window-y-sroll** defines the vertical scroll position in the current window. It sets the number of lines are scroll up off the top of the current window, it is indirectly set by commands such as **scroll-up**(2), **forward-line**(2) etc.

## EXAMPLE

The following example first stores the current window's buffer position and the window layout. The middle '`...`' section could be replaced with macro code performing any number of operations before the last section which restores the initial position:

```
set-variable #l0 $window-line
set-variable #l1 $window-col
set-variable #l2 $window-xcl-scroll
set-variable #l3 $window-x-scroll
set-variable #l4 $window-y-scroll
    .
    .
    .
set-variable $window-line #l0
set-variable $window-col #l1
set-variable $window-xcl-scroll #l2
set-variable $window-x-scroll #l3
set-variable $window-y-scroll #l4
```

## NOTES

If these variables are set by the user or a macro the value is validated against the **$scroll**(5) method and the current cursor position which may lead to the variable being reset if found to be invalid. For example, if the current line is 10 when the **$window-y-scroll** is set to 20 the variable will be reset to 0 as a value of 20 will mean the current line is not displayed in the current window.

## SEE ALSO

**scroll-left**(2), **scroll-up**(2), **$scroll**(5), **$window-line**(5), **$window-col**(5), **$window-acol**(5).

Ref: $window-x-scroll(5) File: m5var081.5 Date: 2022/06/12

## NAME

%compile-com - Default system compile command line

## SYNOPSIS

***%*compile-com** "*string*"; Default is "`make`"

## DESCRIPTION

Sets the default command-line inserted into the message line when the **compile**(3) command is executed. ***%*compile-com** does not need to be defined to run the **compile** command.

## SEE ALSO

**compile**(3), ***%*xgrep-com**(5).

Ref: %compile-com(5) File: m5var039.5 Date: 2022/06/12

## NAME

%ftp-flags - "Configure the FTP console"
%http-flags - "Configure HTTP and HTTPS support"

## SYNOPSIS

**%ftp-flags** "[c—d—i—p—s]" ; Default when undefined is "ci".
**%http-flags** "[C—c—d—h—i—p—s—v—w]" ; Default when undefined is "c".

## DESCRIPTION

The **%ftp-flags** and **%http-flags** modify the behavior of the editor during FTP and HTTP file transfers, respectively. (see **ftp**(3) and **find-file**(2)).

By default, the flags are disabled, the facilities outlined below are enabled by setting the variable in the user configuration. The flag values for both flags are defined as follows:

**C** HTTP(S) only: close socket after every request. When this flag is not given MicroEmacs sends `keep-alive` so the socket connection may be reused.

**c** Create a console buffer (`*ftp-console*` for ftp, `*http-console*` for http) into which the FTP/HTTP command interactions with the remote server are logged.

**d** Show details of commands executed and returned headers.

**h** HTTP(S) only: halt on a redirection. When this flag is not given MicroEmacs will automatically follow up to 5 redirections (`Location:` in the response header), note that a **%http-post-file**(5) or **%http-post-data**(5) is only sent on the initial request.

**i** FTPS, FTPE & HTTPS only: ignore certificate errors, useful when trying to load pages from a self-signed site.

**p** Show the download progress within the console window ('#' for every 2KB downloaded)

**s** Show the console whenever a FTP/HTTP operation is performed. The console is popped into the display pane and shows the current interaction status.

**v** FTPS, FTPE & HTTPS only: show verbose information such as certificate details.

**w** FTPS, FTPE & HTTPS only: show warnings of potential issues with the secure socket support layer.

Typically the following flags are enabled in the *user*.emf file:

```
set-variable %ftp-flags "cisp"
set-variable %http-flags "csp"
```

Once familiar with this facility the console pop-up becomes inconvenient and the flags are typically reduced to:

```
set-variable %ftp-flags "ci"
set-variable %http-flags "c"
```

This disables the pop-up feature of the console. Enabling the limited flag set allows some post mortem debugging to be performed if anything goes wrong. The console buffers are manually selected when these flags are set.

## NOTES

Note that ftp and http facilities are available on UNIX by default, but must be compiled in for Windows versions.

## SEE ALSO

**%http-cookies**(5), **%http-header**(5), **%http-post-data**(5), **%http-post-file**(5), **%http-proxy-addr**(5), **%sock-send-timeout**(5), **find-file**(2), **ftp**(3).

Ref: %ftp-flags(5) File: m5var049.5 Date: 2026/02/23

## NAME

%http-cookies - Set HTTP request cookies
%http-header - Set additional HTTP request headers

## SYNOPSIS

**%http-cookies** "*cookie=value; ...*"; Default is undefined
**%http-header** "*:VERB:*"; Default is undefined
**%http-header** "*Header1: value1\r\n ...*";
**%http-header** "*:VERB:Header1: value1\r\n ...*";

## DESCRIPTION

If the **%http-cookies** variable is set all HTTP file loading requests, using commands like **find-file**(2), are sent with a **Cookie:** request header (if not an empty string) and all response **Set-Cookie:** header values will be merged into the current value, i.e. if the named cookie already exists in the **%http-cookies** variable its value will be updated.

If the variable is not defined then no **Cookie:** request header is sent and all response **Set-Cookie:** headers are ignored.

If the **%http-header** variable is set, if the first character is a ':' then the string up to the next ':' is used as the HTTP request verb, e.g. POST, PUT, DELETE etc. If the value does not start with a ':' then the default verb, which is either GET or POST, is used. The remainder of the value, if any, is appended to the end of the standard request header, but before the terminating blank line.

When a header is appended, the value is prefixed and suffixed with a '\r\n' newline, therefore if the value starts, ends or contains double newline character sequences the URL request will be invalid.

## NOTES

The **%http-cookies** variable should take the form of "<name1>=<value1>; <name2>=<value2>; ...; <name#>=<value#>" with no trailing ';', names and values should be URL encoded.

As with variables like @wl 4 , the returned value can be longer than the maximum length of a user defined variable so care is needed if manipulating its value using macro code.

MicroEmacs cookie support is minimal, it does not support domains, expires, paths etc. it is up to the user or supporting macros to implement any additional features required.

**%http-header** can effect how a file is submitted using **%http-post-file**(5), see help for the variable for more information.

## SEE ALSO

**%http-flags**(5), **%http-post-data**(5), **%http-post-file**(5), **%http-proxy-addr**(5), **find-file**(2), **ftp**(3).

Ref: %http-cookies(5) File: m5var013.5 Date: 2024/09/02

## NAME

%http-post-data - Set HTTP request form data
%http-post-file - Set HTTP request file to submit

## SYNOPSIS

**%http-post-data** "*name=value&...*"; Default is undefined
**%http-post-file** "*file-name*"; Default is undefined

## DESCRIPTION

If either of these variables are defined the next HTTP file loading request, using a command like **find-file**(2), will be changed from a **GET** to a **POST**, submitting the required data in the first request. The variable at this point is unset so that if the request leads to a redirect, the data is only submitted on the first request.

When **%http-post-data** variable is set to a non-empty string its value is submitted as a simple URL encoded form data.

When **%http-post-file** variable is set to a non-empty string its value must be set to the absolute path of an existing local file that is to be submitted with the http request. If **%http-header**(5) is set and includes "`Content-type:`" then the given file becomes the full data body. However, if the variable is not set or does not contain the header string "`Content-type:`" then the file is submitted as part of a multi-part form submission (with "`Content-Disposition: form-data;`" and "`Content-Type: application/octet-stream`").

Both variables cannot be used at the same time.

## SEE ALSO

**%http-flags**(5), **%http-cookies**(5), **%http-header**(5), **%http-proxy-addr**(5), **%sock-send-timeout**(5), **find-file**(2), **ftp**(3).

Ref: %http-post-data(5) File: m5var014.5 Date: 2024/09/02

## NAME

%http-proxy-addr - Set HTTP proxy server address
%http-proxy-port - Set HTTP proxy server port

## SYNOPSIS

**%http-proxy-addr** "*proxy-addr*"
**%http-proxy-port** "*port-number*"; Default is `80`

## DESCRIPTION

If the **%http-proxy-addr** variable is set all HTTP file loading requests, using commands like **find-file**(2), are sent via the given proxy server. **%http-proxy-port** should be set to the proxy servers port number, defaulting to `80` if not set. These variables are typically set in your `<user>.emf` setup file, e.g.:

```
set-variable %http-proxy-addr "proxy.foobar.com"
set-variable %http-proxy-port "8080"
```

## NOTES

Note that http is available on UNIX by default, but must be compiled in for windows versions.

## SEE ALSO

**%http-flags**(5), **%http-cookies**(5), **%http-post-data**(5), **%http-post-file**(5), **%sock-send-timeout**(5), **find-file**(2), **ftp**(3).

Ref: %http-proxy-addr(5) File: m5var079.5 Date: 2024/07/16

## NAME

%os-chars - Characters used by the OS

## SYNOPSIS

***%*os-chars** "*string*"; Default is platform specific

## DESCRIPTION

***%*os-chars** defines the characters used by the current operating system in specific places. The characters have fixed indices defined as follows:

Index 0 Directory separator character, used when giving a full path, MicroEmacs always uses '/' internally but these many need to be converted for some operations.

Index 1 Character used after a drive letter, typically ':', if the current OS does not support drives this character will be a space (' ').

Index 2 Path separator character, typically used in the system's PATH environment variable to separate each path.

Index 3 Multiple command separator character, used to separate two commands on a system command-line where both commands are always run, regardless of the exit-code of the first command. Note that two ampersands ('**&&**') works in a similar way except if the exit code of the first (left hand) command is non-zero, indicating an error, the other command is not executed.

## SEE ALSO

**file-tool-setup**(3).

Ref: %os-chars(5) File: m5var096.5 Date: 2025/07/27

## NAME

%sock-send-timeout - Set socket send timeout
%sock-recv-timeout - Set socket receive timeout

## SYNOPSIS

***%*sock-send-timeout** "*length*"; Default is 20sec
***%*sock-recv-timeout** "*length*"; Default is 300sec

## DESCRIPTION

***%*sock-send-timeout** sets the timeout in seconds for sending data, via ftp or http protocols, down a socket. If the variable is set to less than or equal to zero the timeout is infinite.

***%*sock-recv-timeout** sets the timeout in seconds for receiving data, via ftp or http protocols, from a socket. Note that a socket connection may terminate for other reasons, such as the server closing the socket after a defined period of inactivity. If the variable is set to less than or equal to zero the timeout is infinite, i.e. MicroEmacs will hang until a response is received.

## NOTES

These variables must be set before the first URL, FTP or HTTP, is loaded, after which a value change to them is ignored.

## SEE ALSO

**%http-flags**(5), **%http-cookies**(5), **%http-post-file**(5), **%http-proxy-addr**(5), **find-file**(2), **ftp**(3).

Ref: %sock-send-timeout(5) File: m5var095.5 Date: 2024/07/16

## NAME

%tag-file - Tags file name
%tag-template - Tag file search string
%tag-option - Tag file search option

## SYNOPSIS

***%tag-file*** "*fileName*"
***%tag-template*** "*string*"
***%tag-option*** "*string*"

## DESCRIPTION

The **%tag-file** and **%tag-template** variables must be defined for **find-tag**(2) to work, they define the information required to locate tag references.
**%tag-file** is the name of the tag file to be used, usually set to "**tags**". **%tag-template** is a regular expression search string used to identify tags in a tag file. For example, a tag usually consists of a name "`%[^\t]`" followed by a tab "`\t`" followed by the file name that contains the function "`%[^\t]`" followed by another tab, followed by the search string and end of line "`%[^\n]\n`", i.e.

```
set-variable %tag-template  "%[^\t]\t%[^\t]\t%[^\n]\n"
```

This would match a **vi**(1) tag string definition, as created by the UNIX utility **ctags**(1). The tags file typically contains entries such as:

```
$auto-time    m5var000.5 /^.XI $auto-time - "Automatic buffer"$/
$buffer-bhook m5var002.5 /^.XI $buffer-bhook - "Buffer macro"$/
$buffer-ehook m5var002.5 /^.XI $buffer-ehook - "Buffer macro"$/
```

The **tag-template** definition is modified to match the output of the **ctags**(1) utility. The format of the tags file may differ from platform to platform, typically the differences are encountered in the line contents field which is usually defined as `/ .... /` for a forward search tag and `? .... ?` for a reverse search tag. Note that a tag's search string typically starts with the character '`^`' and ends with '`$`' which indicate the start and end of the line. The variable fields are expected to be in conventional order of *label*, *filename* and *lineText*.

**%tag-option** is a user defined variable that modifies the behavior of **find-tag**(2). This is defined as a string, where each character identifies an option, the options are defined as:

**m** - Enable multiple tags support
: Allows a single tag to be present multiple times in the tag file, typically used when a function is defined multiple times. When enabled **find-tag** can be used to loop through all definitions of a given tag.

**r** - recursive tags file
: By default, the **tags** file is assumed to reside in the current directory location. The **r** option enables an ascending search up the directory hierarchy from the current directory position in search of a recursively generated tags file.

**c** - Continue recursive tag search
: Used in conjunction with flag **r**; when not specified, the recursive searching of a tag stops at the first tag file found, regardless of whether the given tag was located in the found tag file. If this flag is given and the tag was not found in the first tag file, the recursive search continues. This allows local tag files to be created and regularly maintained, yet still being able to access a higher level tag file when required.

When not defined the default value is ”`mrc`”. Modifications to this variable should be made in the *user*.emf file, e.g. To enable multi recursive ascent tag searching define:

```
set-variable %tag-option  "mrc"
```

## NOTES

Note that GNU Emacs uses it’s own tag file format generated by **etags**(1) which does not contain the appropriate information to drive the MicroEmacs **find-tag** command.

The above settings should support the extended version 2 tag file format which has an extra tag type field at the end of each line.

## SEE ALSO

**ctags**(1), **ctags**(3f), **find-tag**(2), **vi**(1).

## NAME

%xgrep-com - Grep command line

## SYNOPSIS

**%xgrep-com** "*string*"; Default is "`grep `"

## DESCRIPTION

Sets the command line used to execute a **grep**(1) system command. The output of the **xgrep**(3) execution should include both file and line number information so that the command **get-next-line**(2) can be used properly. This is not defined by default and the **xgrep** command will not execute until it is defined.

**grep**(1) is typically used with the **-n** option which produced line numbering information which drives the **get-next-line**(2) command.

## EXAMPLE

The following example shows how the **xgrep** strings are defined.

```
set-variable %xgrep-com "grep -n "
0 add-next-line "*grep*"
add-next-line "*grep*" "%f:%l:"
```

This definition corresponds to a **grep** output such as:

```
m5var000.5:13:Sets the  number of seconds to wait
m5var000.5:14:temporary file to t seconds. A
m5var000.5:15:Note than the  temporary
m5var000.5:17:saving a buffer.  Backup  files are
m5var000.5:24:On unlimited  length  file  name  systems
```

where **grep** produces file and line number information for every match.
Use **add-next-line**(2) to define the line pattern produced by **grep**. Some versions of **grep** place the file name on a single line matches within the file occur on subsequent lines. In this case additional *add-next-line* patterns may be defined to cater for the **grep** output as follows:

```
set-variable %xgrep-com "grep /n "
0 add-next-line "*grep*"
add-next-line "*grep*" "File: %f:"
add-next-line "*grep*" "%l:"
```

This definition would be used with a **grep** output such as:

```
File:m5var000.5:
13:Sets the  number of seconds to wait
14:temporary file to t seconds. A
15:Note than the  temporary
17:saving a buffer.  Backup  files are
24:On unlimited  length  file  name  systems
File:m5var001.5:
```

## NOTES

**grep**(1) is a standard utility on UNIX systems. For Windows 95/NT a version of GNU **grep** may be found at:

*<ftp.winsite.com/ftp/pub/pc/winnt/misc/gnugrep.zip>*

For MS-DOS users, a DJGPP port of **grep** is also available on the net. A commercial version of **grep** is also available from MKS.

The MicroEmacs built in **grep**(3) provides similar functionality and does not rely on an external program.

## SEE ALSO

**add-next-line**(2), **grep**(1), **grep**(3), **xgrep**(3), **rgrep**(3), **add-next-line**(2).

Ref: %xgrep-com(5) File: m5var025.5 Date: 2025/08/28

## NAME

.about.argc, .about.arg0, .about.arg1, ... .about.arg# - Command line argument support variables

## SYNOPSIS

**.about.argc** *integer* - Command line argument count
**.about.arg0** *string* - First command line option
**.about.arg1** *string* - Second command line option
**.about.arg2** *string* - Third command line option
...
**.about.arg#** *string* - Last command line option (# = .about.argc-1)

## DESCRIPTION

These command line variables are only set if the **-f** command line option is used and they only reflect the options given after this option. For example, if MicroEmacs was launched as follows:

me -p -n @example.emf -f opt0 opt1 opt2 opt3

Then **.about.argc** would be set to `4`, **.about.arg0** to "`opt0`", **.about.arg1** to "`opt1`" etc. up to **.about.arg3**. Variable **.about.arg4** would not be set.

If the **-f** option is not given then **.about.argc** and all **.about.arg#** variables remain unset.

## EXAMPLE

The following example shows how MicroEmacs can be run as find or grep using the single *cfind.emf* start-up file:

me -p -n @cfind.emf find *.emf

me -p -n @cfind.emf grep -rmi "search-string" *.emf

The *cfind.emf* macro uses **.about.arg0**, which will be "`find`" or "`grep`", to determine the required functionality and how to interpret the subsequent options.

## SEE ALSO

**me**(1).

Ref: .about.argc(5) File: m5var094.5 Date: 2024/05/08

## NAME

.calc.answer - Last successful calc answer
.calc.result - Last calc calculation result

## SYNOPSIS

**.calc.result** *number*
**.calc.result** *string*

## DESCRIPTION

**.calc.answer** is used to store the answer of the last successful calculation made by **calc**(3). When defined the value will always be a number, but its format could be floating point, or a binary, octal, decimal or hex integer depending on the output format requested in the sum.

The "*Ans*" (Answer) in the next calculation is substituted with this value.

**.calc.result** is used to store the result of the last calculation made by **calc**(3). This can be the same as **.calc.answer**, a hybrid form of "<**floating-point**> (<**integer**>)" values when floating point and integer maths produce different answers and the required format is not specified, or, if there is an error in the calculation string a MicroEmacs list with the first element being the character offset to the point of error and the second element is an error message.

## SEE ALSO

**calc**(3).

## NAME

.Xdefaults - UNIX Window Manager defaults file

## SYNOPSIS

**MicroEmacs.font:** *font*
**MicroEmacs.geometry:** *char-width*x*char-height* [**+-**]*xoffset* [**+-**]*yoffset*
**MicroEmacs.name:** *name*
**MicroEmacs.iconname:** *name*

## PLATFORM

**UNIX environments only**

## DESCRIPTION

**.Xdefaults** is the UNIX X-Windows configuration file that permits some of the X-Windows start up settings to be configured. The file is typically loaded into the X database when the Window Manager starts and the values are read from the database by MicroEmacs at start up, over-riding any internal defaults.

The **.Xdefaults** file is used by other applications and MicroEmacs may be added to the end of any existing file.

The values interpreted by MicroEmacs are defined as follows:

**MicroEmacs.font:** *font*
: Specifies the font to use.

**MicroEmacs.geometry:** *char-width*x*char-height* [**+-**]*xoffset* [**+-**]*yoffset*
: The geometry of the X-Window at start-up. *char-width* and *char-height* define the width and depth of the frame specified in characters of the current font. *xoffset* and *yoffset* specify the position of the window with respect to the top left of the screen, defaults are used when specified as 0.

**MicroEmacs.name:** *name*
: Over-rides the name prefix used in the X-Window title bar, by default this is specified as *MicroEmacs*.

**MicroEmacs.iconname:** *name*
: Over-rides the name prefix used in a X-Window iconic display, by default this is specified as *MicroEmacs*.

## EXAMPLE

The following is an example of the **.Xdefaults** file:

```
!#####################################################################
!#                                                                   #
!# Defaults for MicroEmacs startup                                   #
!#                                                                   #
!#####################################################################
!# MicroEmacs.font: <fontname>
!# Set the default font.
MicroEmacs.font: -*-lucidatypewriter-medium-r-*-*-*-*-*-*-m-*-*-*
!# MicroEmacs.font: -*-clean-medium-r-*-*-*-130-*-*-*-*-*-*
```

```
!# MicroEmacs.geometry: <char-width>x<char-height> {+/-}<xoffset> {+-}<yoffset>
!# Set up the default screen size and position.
!# 102x65 characters please, 30 pixels from top left
MicroEmacs.geometry: 102x65 30 30

!# MicroEmacs.name: <name>
!# The name displayed in the title bar format is <name>: <file>
MicroEmacs.name: JASSPAs MicroEmacs

!# MicroEmacs: <icon-name>
!# The name displayed when MicroEmacs is iconized
MicroEmacs.iconname: JASSPAs MicroEmacs
```

## NOTES

The **.Xdefaults** file is typically read once at login and and any changes to the files are not enacted until the Window Manager is re-started or a new login session commences.

The settings in the **.Xdefaults** may be subsequently overridden by the settings defined in **user-setup**(3).

## SEE ALSO

**user-setup**(3).

Ref: .Xdefaults(8) File: m8fil007.8 Date: 2026/06/19

## NAME

eaf - MicroEmacs abbreviation file format

## SYNOPSIS

*<pattern> <insertionString>*
*<pattern> <insertionString>*
*<pattern> <insertionString>*
*<pattern> <insertionString>*

## DESCRIPTION

The MicroEmacs abbreviation file, typically given the extension **.eaf**, defines a set of shorthand expansion strings which are used by the command **expand-abbrev**(2). **buffer-abbrev-file**(2) defines the abbreviation file.

The abbreviation file line based, with one abbreviation per line, with no intervening blank lines. Each line comprises of two columns, the first column *<pattern>* identifies the source pattern to be expanded, the second column *<insertionString>* defines the replacement text. The two text columns are separated by a space character.

When **expand-abbrev**(2) is invoked and the expansion *<pattern>* is recognized, then *<pattern>* is deleted from the buffer and replaced with *<insertionString>*.

The fields are defined as follows:

*<pattern>*
: The source pattern to be expanded. The data commences in text column 0 and spans to the first white space character (`SPACE` or `tab`). The pattern may not include any white space characters.

*<insertionString>*
: The replacement string exists from the first non-white space character following the *<pattern>* to the end of the line. The replacement string may include special tokens, delimited by a backslash ('\') character which are interpreted as follows:

  **\b** Move cursor backwards
  : A positioning control. Allows the cursor to be moved backwards 1 character.

  **\d** Delete `tab` backwards
  : Back tab. Deletes a tab character backwards.

  **\m"***<string>***"** Macro execution
  : Takes the remainder of the line as a keyboard macro definition. The macro *string* is generated using **insert-macro**(2) and must be contained in double quotes. When invoked the keyboard macro is executed and the appropriate text is inserted into a buffer. This is typically only used for more complex operations.

  **\p** Position
  : The resultant position of the cursor following the expansion. If the cursor position is not specified, the cursor is placed at the end of the expansion string by default.

**\r** Carriage Return (Newline)
A newline in the replacement text. Note while **indent**(2m) is enabled a sequence a single "\r" retains the indent on the next line, however a sequence of two "\r\r" characters does not retain the tab position and returns the cursor to the start of the second line. If blank lines are required retaining tab positioning then a keyboard macro string should be used instead. (see "\m" above).

**\t** Tab A `tab` character in the replacement text.

## EXAMPLE

The following example provides abbreviations for the 'C' programming language, found in file **c.eaf**. All cursor positions in the examples are shown by <@>.

```
#i #include <\p>\r
#d #define \p
if if(\p)\r{\r\r}\r
ef else if(\p)\r{\r\r}\r
el else\r{\r\p\r}\r
wh while(\p)\r{\r\r}\r
sw switch(\p)\r{\rcase :\rdefault :\r}\r
```

Given that the abbreviation file has been declared then expansion of:

```
#d<@>       =>      #define <@>

if<@>       =>      if(<@>)
                    {

                    }

sw<@>       =>      switch(<@>)
                    {
                    case :
                    default :
                    }
```

Note, in all of the examples, the abbreviation replacement strings specify a resultant cursor position, typically where the next edit will take place.
The macros may alternatively be defined using keyboard macros. The aforementioned macros could have been re-written with the following definitions which are equivalent:

```
#i \m"#include <\CX\CAP>\CM\CXaP\CX)"
#d \m"#define \CX)"
if \m"if(\CX\CAP)\CM{\CM}\CXaP\CX)"
ef \m"else if(\CX\CAP)\CM{\CM\CM}\CM\CXaP\CX)"
el \m"else\CM{\CM\CX\CAP\CM}\CM\CXaP\CX)"
wh \m"while(\CX\CAP)\CM{\CM\CM}\CM\CXaP\CX)"
sw \m"switch(\CX\CAP)\CM{\CMcase :\CMdefault:\CM}\CM\CXaP\CX)"
```

Within a macro, the cursor positioning is generally achieved by setting a mark where the resultant cursor is to be positioned (see **set-mark**(2)), when the macro is finished then an **exchange-point-and-mark**(2) is initiated to move the cursor to the correct position; alternatively a sequence of cursor movements may be used.
The "\b" and "\d" are typically used for positioning the cursor on subsequent lines. "\d" is the inverse of "\t". Consider the following Pascal definition for an *else*, *begin* and *end* sequence:

```
el else\rbegin\r\t\p;\r\dend;
```

with **indent**(2m) mode enabled generates:

```
        else
        begin
            <@>;
        end;
```

Similarly the "\b" is typically used when **indent**(2m) is enabled, but when the tab spacing is known. Consider the following example used in the MicroEmacs **.emf** files to define a help entry. In this case the indent is known to be 5 characters. Hence to move the cursor back 5 characters then a sequence of **\b**'s are used:

```
!h def .. \rSEE ALSO\r     <cross references>\r\b\b\b\b\b!ehelp
```

the expansion in this case is:

```
define-help "<@>"

...

SEE ALSO
     <cross references>
!ehelp
```

## FILES

The default abbreviation files are located in the MicroEmacs *home* directory.

User's may specify their own abbreviation files by shadowing the *home* directory file with their own file located in a personal MicroEmacs directory. See **$user-path**(5).

## SEE ALSO

**expand-abbrev**(2), **buffer-abbrev-file**(2), **global-abbrev-file**(2), **iso-accents-mode**(3), **$user-path**(5).

Ref: eaf(8) File: m8fil000.8 Date: 2022/06/12

## NAME

edf - MicroEmacs spelling dictionary file

## SYNOPSIS

**lsdmenus.edf**
*user*.**edf**

## DESCRIPTION

The spelling dictionary files are given the extension **.edf**. These are binary files read by MicroEmacs and cannot be edited directly.

MicroEmacs is supplied with a dictionaries for various languages. It is recommended that these dictionaries are not modified, a personal dictionary is used and modified instead.

A personal dictionary, *user*.**edf**, is automatically created in the users directory for additional spelling information.

## FILES

The standard dictionary files lsdm<*language*><*country*>.**edf** are located in the MicroEmacs *home* directory.

User's may create their own dictionary files by shadowing the *home* directory file with their local dictionary(s) located in a personal MicroEmacs directory. See **$search-path**(5).

## SEE ALSO

**spell**(2), **add-dictionary**(2), **$search-path**(5).

Ref: edf(8) File: m8fil002.8 Date: 2025/08/28

## NAME

ehf - MicroEmacs help file

## SYNOPSIS

**!**<*helpTag*>
<*Text Description*>
...
**—**<*helpId*>
<*Text Description Line*>
...
**$** *?*
...
<*Text Description*>
**!**<*helpTag*>
**!**<*helpTag*>
<*Text Description*>
...

## DESCRIPTION

The on-line help information is retained in the file **me.ehf**, this is an ASCII text file which holds all of the on-line help information. The help file comprises of formatted text <*Text Description*> which is literally displayed to the user when help information is requested. Each text description is delimited into pages with a **!**<*helpTag*> which identifies the block of text with a help label.

The **!**<*helpTag*> is placed before the text description and is identified by a exclamation mark ('**!**') placed at the beginning of the line. The <*helpTag*> is the identifying name used by the help system and takes the following form:

`LSSNNNN...`

Where:

**L** Is the length of the "`NNNN...`" name which must be matched, a value of ' ' indicates that the whole name must be matched, otherwise the value must be in the range '`1`' - '`9`' indicating the number of characters to be match.

**SS** Is the section number of the page, the first character should be a numeric (i.e. '`3`' for a macro) and the second is an optional section letter. A value of ' ' indicates no section number and/or letter.

**NNNN...**
The page name, the length is unlimited but must be on one line.

Multiple <*helpTag*>'s may be associated with a common text description by proceeding a block of text with multiple tags, each on a separate line, with no intervening non-tag lines (i.e. lines that do not commence with **!**).

The <*Text Description*> that follows is the text associated with the tag. When the help system is invoked with the tag then the text is displayed. There are 2 types of

internal command lines, lines starting with a '|' indicate that the following line should only be displayed if the requested help page is <*helpId*>, where <*helpId*> is the the name used in the <*helpTag*>. This is a useful mechanism for pages with multiple <*helpTag*>s.

Lines which contain just "$?" are MicroEmacs command lines where ? can be:

a For a command help page display any global key bindings, for variables display its current value.

MicroEmacs uses a special hilighting scheme to control color schemes and hyper-text links, the special embedded tags all start with and escape character (0x1b or '^[') and are defined as follows:

^[c? Tag used to change color where ? can be:

- A white, used for main text.
- B red, used for underlining.
- C green, used for italic font.
- D cyan, used for bold font.
- E light yellow, used for a header.
- F light red, used for and image link.

^[s? Tag used to change hilighting scheme where ? can be:

- A Normal ehf hilight.
- B MicroEmacs macro (or emf) hilighting.

Note that other tags can only be used in the normal ehf hilighting scheme.

^[ls<link>^[lm<name>^[le
Used to create hyper-links, <link> is the help link name which can be omitted if it is the same as <name>. <name> should not contain any other tags, it is automatically displayed in the magenta color scheme.

## NOTES

When the help system is invoked for the first time, **me.ehf** is loaded into internal memory and fragmented into labeled pages using the <*helpTag*> information. Hence, any edits made to **me.emf** are not visible in the help system until the next session.

Macros and alike may add additional help information to the internal help database at run-time using the **define-help**(2) command.

The help hilighting is applied to the help buffer from the hilighting macro's defined in **hkhelp.emf**. The hilighting is NOT part of the help file.

Special hilighting keys may be included in **me.ehf** provided that they are interpreted by the help hilighting defined in **hkehf.emf**.

<*Text Description*> lines cannot commence with **!**, **—** or **$** in the first column.

## EXAMPLE

The following help entry defines the help for **global-mode**(2), **add-global-mode**(3) and **delete-global-mode**(3). It uses most features mentioned, namely multiple link names, color and scheme changes and several hyper-text links:

```
! 2 global-mode
! 3 add-global-mode
! 3 delete-global-mode
^[cE^[cENAME ^[cE^[cA


|global-mode
    global-mode - Change a global buffer mode
|add-global-mode
    add-global-mode - Set a global buffer mode
|delete-global-mode
    delete-global-mode - Remove a global buffer mode
$a


^[cE^[cESYNOPSIS ^[cE^[cA


    ^[cCn^[cA ^[cDglobal-mode^[cA "^[cCmode^[cA" (^[cDesc m^[cA)
    ^[cDadd-global-mode^[cA "^[cCmode^[cA"
    ^[cDdelete-global-mode^[cA "^[cCmode^[cA"


^[cE^[cEDESCRIPTION ^[cE^[cA


    ^[cDglobal-mode^[cA changes the state of one of the hereditary
    global modes. A buffer's modes are initialized to the global
    modes when first created. This command is very useful in changing
    some of the default behavior such as case sensitive searching (see
    the example below). See ^[ls^[lmOperating Modes^[le for a full list
    and description of modes. Also see ^[ls^[lmbuffer-mode(2)^[le for a
    full description of the use of the argument ^[cCn^[cA.

    The ^[ls^[lminfo(2)^[le command gives a list of the current global
    and buffer modes.

    ^[cDadd-global-mode^[cA and ^[cDdelete-global-mode^[cA are macros
    defined in me3_8.emf which use global-mode to add or remove a global
    mode. They are defined for backward compatibility and for ease of
    use; they are simple macros, add-global-mode is defined as follows:
^[sB
        define-macro add-global-mode
            ; Has the require mode been given as an argument, if so add it
            !force 1 global-mode @1
            !if &not $status
                ; No - use 1 global-mode to add a mode
                !nma 1 global-mode
            !endif
        !emacro

^[sA

^[cE^[cEEXAMPLE ^[cE^[cA


    The following example globally disables ^[ls^[lmexact(2m)^[le and
    ^[ls^[lmmagic(2m)^[le modes, if these lines are copied to the user
    setup file then are searches will be simple and case insensitive by
    default:
^[sB
        -1 global-mode "exact"
        -1 global-mode "magic"

^[sA
```

```
^[cE^[cENOTES ^[cE^[cA

    Globally adding ^[ls^[lmbinary(2m)^[le and ^[ls^[lmcrypt(2m)^[le
    modes is strongly discouraged as any file loaded would be assigned
    these modes. Instead the use of commands ^[ls^[lmfind-bfile(3)^[le
    and ^[ls^[lmfind-cfile(3)^[le are recommended.

    ^[ls^[lmauto(2m)^[le, ^[ls^[lmautosv(2m)^[le, ^[ls^[lmbackup(2m)^[le,
    ^[ls^[lmexact(2m)^[le, ^[ls^[lmmagic(2m)^[le, ^[ls^[lmquiet(2m)^[le,
    ^[ls^[lmtab(2m)^[le and ^[ls^[lmundo(2m)^[le modes are present on all
    platforms by default. On Windows and DOS platforms ^[ls^[lmcr(2m)^[le
    is also present and on DOS ^[ls^[lmctrlz(2m)^[le is also present.


^[cE^[cESEE ALSO ^[cE^[cA

    ^[ls^[lmOperating Modes^[le, ^[ls^[lmbuffer-mode(2)^[le,
    ^[ls^[lmfind-bfile(3)^[le, ^[ls^[lmfind-cfile(3)^[le,
    ^[ls^[lminfo(2)^[le.
```

## FILES

The help file **me.ehf** is located in the MicroEmacs *macros* directory and should exist on the **$search-path**(5).

## SEE ALSO

**define-help**(2), **$search-path**(5).

Ref: ehf(8) File: m8fil005.8 Date: 2025/08/28

## NAME

emf - MicroEmacs macro file

## FILES

**me.emf** - The start-up macro file
<*logname*>**.emf** - User start-up macro file

## EXTENSIONS

.emf - MicroEmacs macro file

## DESCRIPTION

MicroEmacs macro files are ASCII text files, given the file extension **.emf**. A number of special macro files exist as follows:

**me.emf** The start-up macro file. This file is the first macro file to be invoked and is used to bootstrap MicroEmacs into the correct configuration.

**hk**<*name*>**.emf**
Macro files prefixed with **hk** generally denote File Hook macro files which are automatically invoked when known file types are loaded.

<*logname*>**.emf**
The users start-up configuration file, typically used to configure the environment with the users preferences.

***term.emf**
Platform specific configuration files, used to configure the environment for a specific platform.

Macro files may be any name, the more prominent macro files are:

**color.emf**
Color definitions for the buffers.

**mouse.emf**
Mouse interaction macros.

**osd.emf**
OSD Menu configuration file.

## FILES

The default start-up file **me.emf** is located in the MicroEmacs *home* directory **$user-path**(5).

User's may create their own start-up and files in their local MicroEmacs directory. The users start-up file is called $user-name**.emf**, and may be used to execute other macro files defined by the user.

## SEE ALSO

File Hooks, **emftags**(3f), **$user-path**(5), **execute-file**(2).

Writing MicroEmacs Macros
Macro Glossary
Macro Debugging - A Crash Course
MicroEmacs Macro Language at a Glance

Ref: emf(8) File: m8fil003.8 Date: 2026/02/25

## NAME

erf - MicroEmacs registry file

## SYNOPSIS

*; Comment to the end of the line*
<*command*> ::= "<*identifier*>" [ = "<*string*>" ][ { <*command*> } ] *

## DESCRIPTION

MicroEmacs registry files are ASCII text files, given the file extension **.erf**. The registry file is a simple syntax that allows an *identifier* to be associated with a *string*. The *identifiers* are unique and allow a *string* value to be found when a search for a *identifier* is made. The *string* component is optional.

The syntax allows the *identifier*'s to be hierarchically nested, children of the *identifier* node are enclosed in a set of curly braces { ... }. The enclosure itself comprises a number of *identifiers*, which may have their own enclosures, and so on.

The backslash character '\' is the escape character, the following sequences of escape character are recognized:

\\ - Literal backslash
\" - Double quote (used within a quoted string)
\n - New line character.
\t - Tab character.

The semi-colon character ';' introduces a comment which exists to the end of the line.

## EXAMPLE

The following is an example of a registry file:

```
; -!- erf -!-
; Comment on this line
"dos"
{
    "file-ignore" = "~ ./ .o"
    "font" = "85"
    "mail-dir" = "c:/mail/"
    "mail-send" = "echo from \"%f\" file \"%o\""
    "mail-src" = "c:/mail/jon"
    "nested"="value"
    {
        "foo"="bar"
    }
}
```

The history file *username.erf* is a good example of the use of the registry. This file retains historical session information in The history registry file is automatically written at the end of a editing session when the editor is closed down (or may be saved explicitly using **save-session**(3)).

Every user should have their own personal history file in their personal MicroEmacs directory. The history file is located from the MicroEmacs search path defined by **$user-path**(5), and is named by the variable **$user-name**(5).

## NOTES

- The registry files are not currently written with a backup.
- Special care should be taken when editing registry files when they are loaded into MicroEmacs. It is recommended that the registry file is not loaded as a registry item when editing the registry text file.

To edit the history registry file within MicroEmacs then the following sequence of steps should be followed:

- Save the current session **save-session**(3).
- Load the history registry file *username*.erf.
- Edit the file.
- Save edits back to the file.
- Re-install the history **read-session**(3). This flushes the current session and restores it from the file. The new edits should now be in the registry.
- Examine the loaded registry using **list-registry**(2).

## SEE ALSO

**list-registry**(2), **read-registry**(2), **read-session**(3), **save-session**(3), **$user-name**(5), **$user-path**(5).

Ref: erf(8) File: m8fil004.8 Date: 2025/08/28

## NAME

esf - MicroEmacs session file

## DESCRIPTION

MicroEmacs session files are ASCII text files that retain all the buffer context for a previous editing session.

Every user has their own personal session file in their personal MicroEmacs directory. The session file is located from the MicroEmacs search path defined by **$user-path**(5), and is named by the variable **$user-name**(5).

## SEE ALSO

**list-registry**(2), **read-registry**(2), **read-session**(3), **save-session**(3), **$user-name**(5), **$user-path**(5).

Ref: esf(8) File: m8fil008.8 Date: 2025/08/28

## NAME

etf - MicroEmacs template file format

## SYNOPSIS

*<Free Form Text>*

## DESCRIPTION

The MicroEmacs template file, typically given the extension **.etf**, is a file template for a new file and defines common text that is automatically included when a new file is created.

The file inclusion is usually performed by macro **insert-template**(3), called from the File Hooks. The template file has no specific format, although **insert-template** replaces key strings with relevant information.

## EXAMPLE

The template file is inserted with the file hooks. If a file hook is called with an argument of `0` then the buffer has been created and the template file is inserted.

```
define-macro fhook-c
    ; if arg is 0 this is a new file so add template
    !if &not @#
        ; Is it an include h file or a c file?
        !if &seq &mid $buffer-bname &rsin "." $buffer-bname 1 "h"
            insert-template "h"
        !else
            insert-template "c"
        !endif
    !endif
    1 buffer-mode "time"
    .
    .
!emacro
```

See **insert-template**(3) for more information on how the template file is located and inserted into the buffer.
The default MicroEmacs 'C' mode template is defined as follows, but may be replaced with any other text:

```
/* -*- C -*- ****************************************************************
 *
 *                      Copyright $YEAR$ $COMPANY_NAME$.
 *                            All Rights Reserved
 *
 *
 *  System        :
 *  Module        :
 *  Object Name   : m8fil001.8
 *  Created By    : $USER_NAME$
 *  Created       : $ASCII_TIME$
 *  Last Modified : <260305.1602>
 *
 *  Description
 *
 *  Notes
 *
 *  History
 *
 ****************************************************************************
 *
 *  Copyright (c) $YEAR$ $COMPANY_NAME$.
```

```
 *
 *  All Rights Reserved.
 *
 * This  document  may  not, in  whole  or in  part, be  copied,  photocopied,
 * reproduced,  translated,  or  reduced to any  electronic  medium or machine
 * readable form without prior written consent from $COMPANY_NAME$.
 *
 ****************************************************************************/

static const char rcsid[] = "@(#) : $Id$";
```

## FILES

The default template files are located in the MicroEmacs *home* directory.

User's may specify their own template files by shadowing the *home* directory file with their own file located in a personal MicroEmacs directory. See **$user-path**(5).

## SEE ALSO

File Hooks.
**insert-template**(3), **&find**(4), **$user-path**(5).

Ref: etf(8) File: m8fil001.8 Date: 2026/02/22

## SYNOPSIS

Ada - Ada programming language

## FILES

**hkada.emf** - Ada programming language file hook definition
**ada.eaf** - Ada programming language abbreviation file
**ada.etf** - Ada programming language template file

## EXTENSIONS

**.ada**, **.adb**, **.ads** - Ada programming language

## DESCRIPTION

The **ada** file type template provides simple highlighting of Ada files.

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Highlighting

The highlighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Abbreviations

The installation comes with a default abbreviation file for Ada file editing which can be used by writing short abbreviations like "**fo**", **if**", "**el**" or "**fu**" - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the Ada file. The folds occur about functions, procedures or records as long as these declarations are on one line and and at a line starting with the **end;** keyword. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Ada mode offers beside folding as well support for displaying an outline of the functions, procedures or records for easy navigation using the **item-list**(3) command.

### Short Cuts

The short cut keys used within the buffer are:

**esc h** - Key binding help information.
**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hash (#).
**C-c C-o** - Format a comment
**f2** - (Un)fold the current region.

**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## BUGS

None reported.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **time**(2m).

Supported File Types

Ref: ada(9) File: m9typ043.9 Date: 2025/08/28

## SYNOPSIS

asciidoc, adoc - AsciiDoc Markup File.

## FILES

**hkadoc.emf** - AsciiDoc file hook definition
**adoc.eaf** - AsciiDoc abbreviation file
**adoc.etf** - AsciiDoc template file

## EXTENSIONS

**.asciidoc**, **.adoc**, **.tdoc** - AsciiDoc files

## MAGIC STRINGS

**-!- asciidoc -!-**

MicroEmacs recognizes the magic string on any of the first 4 lines of the file. The HTML files may be extension-less and are still recognized.

## DESCRIPTION

The **AsciiDoc** file hook provides a editing environment for AsciiDoc files. The file hook provides highlighting, folding of sections, document outline using item-list and abbreviations.

File recognition is performed using the standard file extension **.adoc**, **.adoc** or **.tdoc** or by the magic string. The **.tdoc** extension indicates that there are non-AscciDoc based extensions for instance to embed code of other programming languages like R or Tcl within the file which needs to be processed with tools like *tmdoc* before the document can be compiled with the *Asciidoc* compiler.

### Abbreviations

The installation comes with a default abbreviation file for AsciiDoc code editing which can be used by writing short abbreviations like "**fig**" for a figure environment, "**bib**" for a bibliography file, "**img**" for an image file, or "**co**" for a code chunk - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the AsciiDoc files. The folds occur about the level 1 and 2 headers, so headers with one or two equal symbols at the beginning of a line. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available for level 1 and 2 headers as well as for lines starting with TODO. These items of the file are displayed using the the *item-list* command.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.

**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close list items

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **md**(9), **typst**(9), **wiki**(9)

Supported File Types

Ref: adoc(9) File: m9typ066.9 Date: 2025/10/20

## SYNOPSIS

Apache - Apache httpd.conf configuration file

## FILES

**hkapache.emf** - Apache httpd.conf configuration file

## EXTENSIONS

**httpd.conf** - Apache configuration file

## DESCRIPTION

The **Apache** file type template provides simple highlighting of the Apache `httpd.conf` file.

### Hilighting

The hilighting features allow commands, variables and operators of the language to be differentiated and rendered in different colors.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph.

### Folding and Information Hiding

Generic folding is enabled, the folds occur about bracketed constructs <...>...</...> located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the key constructs of the file may be listed in the *item-list*.

### Short Cuts

The short cut keys used within the buffer are:

**esc h** - Key binding help information.
**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hash (#).
**C-c C-o** - Format a comment
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - List items.
**esc f6** - Close list items window.

## SEE ALSO

Supported File Types

Ref: apache(9) File: m9typ038.9 Date: 2022/06/12

## SYNOPSIS

asm, s - Assembler File

## EXTENSIONS

**.s** - Platform specific assembler file.
**.asm** - Platform specific assembler file.

## DESCRIPTION

The standard assembler file extensions **.s** and **.asm** are by default not bound to any hook functions as they are platform specific. The user should define a default binding for the assembler file types as appropriate to the current platform and assembler development. i.e. for the Windows environment the **x86**(9) file type would be conditionally bound to the file e.g.

```
!iif &seq $platform "windows"  major-mode "asm" ".asm .s .x86"
```

## SEE ALSO

**x86**(9).
File Hooks, Supported File Types.

Ref: asm(9) File: m9typ030.9 Date: 2026/01/17

## SYNOPSIS

asn.1 - ASN.1 File

## FILES

**asn1.emf** - asn.1 file hook definition
**ans1.etf** - Template file

## EXTENSIONS

**asn1** - ASN.1 files.

## MAGIC STRINGS

**-!- asn.1 -!-**

Recognized by MicroEmacs only, defines the file to be a asn.1.

## DESCRIPTION

The **asn1** file type template handles the hilighting of the asn.1 files.

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

By default, `TAB`'s are enabled as this is the syntactical feature of the file.

### Hilighting

The hilighting emphasizes the keywords and comments within the asn.1.

## BUGS

None reported.

## SEE ALSO

**time**(2m).
Supported File Types

Ref: asn.1(9) File: m9typ020.9 Date: 2022/06/12

## SYNOPSIS

asp - Active Server Pages.

## FILES

**hkasp.emf** - Active Server Pages file hook definition

## DESCRIPTION

The **asp** file type template provides simple hilighting of Active Server Pages embedded within **html**(9) files.

## SEE ALSO

Supported File Types

Ref: asp(9) File: m9typ045.9 Date: 2022/06/12

## SYNOPSIS

au3 - Autoit 3 Scripting Language File.

## FILES

**hkau3.emf** - Autoit 3 Scripting Language file hook definition
**au3.etf** - Autoit 3 Scripting Language header template file
**au3.eaf** - Autoit 3 Scripting Language abbreviation file

## EXTENSIONS

**.au3** - Autoit 3 file

## DESCRIPTION

The **autoit** file type template provides the hilighting, indentation, folding, item list and tools definitions for a perl file.

File recognition is performed using the standard file extension **.au3**.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow functions, commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Abbreviations**

The installation comes with a default abbreviation file for Autoit file editing which can be used by writing short abbreviations like "**if**", "**wh**" (while) or "**fu**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Auto Layout**

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

**Folding and Information Hiding**

Generic folding is enabled within perl files. The folds occur about *Func*tion definitions located on the left-hand margin and their closing *EndFunc* keyword which is located as well at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Autoit mode offers beside folding as well support for displaying an outline of the functions for easy navigation using the **item-list**(3) command.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-o** - Restyle comment.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close item list

## NOTES

The hilighting is typically extended using a file **myau3.emf**.

## BUGS

None which are currently known. As the author does not use this scripting language anymore, the highlighting and folding and item-list command might be not working as expected in all cicrumstances and support of more modern features of this scripting language on the Windows platform might be not available.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **indent**(2), **item-list**(3), **restyle-buffer**(3), **time**(2m).

Supported File Types

Ref: au3(9) File: m9typ059.9 Date: 2025/08/28

## SYNOPSIS

awk - AWK files

## FILES

**hkawk.emf** - AWK hook definition
**awk.etf** - AWK Language header template file
**awk.eaf** - AWK Language abbreviation file

## EXTENSIONS

**.awk** - AWK file
**.gawk** - GNU AWK file
**.nawk** - New AWK file

## MAGIC STRINGS

**#![ \t]*/.*awk**

MicroEmacs recognises the magic string on the first line of the file used to locate the executable. The awk files may be extensionless and are still recognised.

## DESCRIPTION

The **awk** file type template provides simple hilighting of AWK files, folding, item listing and abbreviations.

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow keywords, variables,comments, strings and characters of the language to be differentiated and rendered in different colors.

### Abbreviations

The installation comes with a default abbreviation file for AWK file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**fu**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within AWK files. The folds occur about *function* and *BEGIN* or *END* block definitions located on the left-hand margin and their closing brace which is located at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The AWK mode offers beside folding as well support for displaying an outline of the functions and the blocks for easy navigation using the **item-list**(3) command.

## BUGS

None reported. Template could probably benifit from some form of auto indentation.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **time**(2m).

Supported File Types

Ref: awk(9) File: m9typ015.9 Date: 2025/08/28

## SYNOPSIS

bat, btm - MS-DOS batch files

## FILES

**hkdos.emf** - MS-DOS hook definition

## EXTENSIONS

**.bat** - MS-DOS Batch file
**.btm** - 4-DOS Batch file

## MAGIC STRINGS

**-!- msdos -!-**

Recognized by MicroEmacs only. Denotes a MS-DOS batch file.

## DESCRIPTION

The **dos** file type templates provide simple hilighting of a MS-DOS batch file. The template provides minimal hilighting support of both standard and 4-DOS batch files.

The *Magic String* may be used within the **config.sys** file to force hilighting of the MS-DOS configuration file.

## BUGS

None reported.

## SEE ALSO

**ini**(9).

Supported File Types

Ref: bat(9) File: m9typ007.9 Date: 2022/06/12

## SYNOPSIS

bnf - Backus-Naur Form

## FILES

**hkbnf.emf** - Backus-Naur Form hook definition

## EXTENSIONS

**.bnf** - Backus-Naur Form file

## DESCRIPTION

The **bnf** file type template provides simple hilighting of text presented in Backus-Naur Form. The hilighting allows the components of the BNF to be differentiated quickly.

The syntactical tokens that are hilighted are:

*<[a-zA-Z].*>*
: Component language identifiers.

**— { } ::=**
: Meta symbols of the BNF format.

## BUGS

None reported.

## SEE ALSO

Supported File Types

Ref: bnf(9) File: m9typ012.9 Date: 2022/06/12

## SYNOPSIS

C, C++ - C and C++ programming language templates

## FILES

**hkc.emf** - C programming language hook definition
**hkcpp.emf** - C++ programming language hook definition
**c.etf** - C programming language template file
**h.etf** - C programming language header template file
**cpp.etf** - C++ programming language template file
**hpp.etf** - C++ programming language header template file

## EXTENSIONS

**.c**, **.h**, **.def** - ANSI C
**.cpp**, **.cc**, **.hpp**, **.rc .C** *(UNIX only)* - C++ programming language
**.l** - LEX
**.y** - YACC
**.i** - C (or C++) pre-processed file (i.e. output from pre-processor).
**.rc** - Microsoft Developer resource file.

## MAGIC STRINGS

**-*- c -*-** Recognized by GNU and MicroEmacs. Denotes a 'C' programming type file, may be used in **.c**, **.def** and **.h** files.

**-*- c++ -*-**
Recognized by GNU and MicroEmacs. Denotes a C++ programming type file, may be used in **.c**, **.def** and **.h** files.

## DESCRIPTION

The **C** and **C++** file type templates offer the most sophisticated editing features within the MicroEmacs environment.

**General Editing**
On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**
The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**
The C-Mode indentation mode of **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the

comment is automatically formatted to a box. If the comment commentces with `/**` then the comment is assumed to be a *Java Doc* comment.

**Tags** A C-tags file may be generated within the editor using the **Tools** -> **C Tools** -> **Create Tag File**. **find-tag**(2) takes the user to the file using the tag information.

On invoking the tag generator then the user is presented with a dialog box which specifies the generation option of the tags file. The base directory of the tags file search and tagging options may be specified to locate all of the definitions within the code space.

The **tags** file is extremely useful where the user is dealing with inherited source code spread over multiple directories. Generation of a recursive tag file with all searching options enabled allows labels to be located extremely rapidly (certainly faster than IDE environments).

**Folding and Information Hiding**

Generic folding is enabled within the C and C++ files. The folds occur about braces {...} located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region. Note that folding does not operate on K&R style code.

The **Tools** -> **C Tools** menu allows `#define`'s to be evaluated within the buffer. Where the state of a `#if` is established to be false (using the `#define` information) then the disabled region of code is grayed out indicating which regions of the code are active.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**rcs-file**(2) is automatically invoked if an RCS file is detected, the normal check-in/out operations may be performed through the editor.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (un)fold the current region
**f3** - (un)fold all regions

**Additional Commands**

A number of additional commands are available as follows:

- **c-slashify**(3) - Append backslash to end of lines of region.
- **c-deslashify**(3) - Remove backslash to end of lines of region.
- **c-to-cpp-convert**(3) - Convert a C file to C++
- **c-hash-eval**(3) - Evaluate C/C++ `#defines`.

- **c-hash-undo**(3) - Remove C/C++ `#define` evaluation.
- **c-hash-define**(3) - Set a C/C++ `#define`.
- **c-hash-undef**(3) - Unset a C/C++ `#define`

**Executing, Formatting and Linting C and C++ Code**

The C and the C++ file hooks supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools, compilers which are currently supported are ”gcc” and ”g++” as compilers, ”astyle” and ”clang-format” (formatters) and ”clang-tidy” (linter). The tools can be installed either using your standard package manager system wide or, for the current user, using the Python package manager ”pip”. Here an example on how to install astyle and the clang tools using pip:

```
pip install astyle-py --user
pip install clang-format --user
pip install clang-tidy --user
```

## NOTES

If the default language is C++, rather than ’C’ the extension list and priority of the `cpp` major-mode should be over-ridden in the users local setup, using:

```
4 major-mode "cpp" ".cc .cpp .hpp .h .def .rc .l .y .i"
```

This defaults all **.h** and **.def** files etc. to be C++ rather than C, and redefining it after `C` mode raises its relative priority.

The hilighting is typically extended using a file **myc.emf** (or **mycpp.emf**) i.e. to include the usual extended types of **int32** etc, **myc.emf** might be defined as:

```
hilight .hilight.c 1 "uint8"    .scheme.type
hilight .hilight.c 1 "int8"     .scheme.type
hilight .hilight.c 1 "uint16"   .scheme.type
hilight .hilight.c 1 "int16"    .scheme.type
hilight .hilight.c 1 "uint32"   .scheme.type
hilight .hilight.c 1 "int32"    .scheme.type
hilight .hilight.c 1 "float32"  .scheme.type
hilight .hilight.c 1 "float64"  .scheme.type
```

## BUGS

The ’C’ and ’C++’ templates have been thoroughly used, there are no known issues with the templates.

The **.rc** hilighting is a little bogus and should not really be mapped onto **.cpp**. Do not attempt to re-style.

## SEE ALSO

**c-deslashify**(3), **c-slashify**(3), **c-to-cpp-convert**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **ctags**(3f), **file-exec**(3), **file-format**(3), **file-lint**(3), **find-tag**(2), **rcs-file**(2), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: c(9) File: m9typ000.9 Date: 2026/06/19

## SYNOPSIS

C# - C# programming language (C sharp)

## FILES

**hkcs.emf** - C# programming language hook definition
**cs.eaf** - C# programming language abbreviation file
**cs.etf** - C# programming language template file

## EXTENSIONS

**.cs** - C#

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the C# file. The folds occur about function, class and enum definitions and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The C# mode offers beside folding as well support for displaying an outline of the functions and classes of the C# file for easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for C# file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**NOTES**

The hilighting or the addition of new item list matches is typically extended using a file **mycs.emf**

**SEE ALSO**

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **java**(9), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: C#(9) File: m9typ068.9 Date: 2026/01/09

## SYNOPSIS

C3 - C3 programming language

## FILES

**hkc3.emf** - C3 programming language hook definition
**c3.eaf** - C3 programming language abbreviation file
**c3.etf** - C3 programming language template file

## EXTENSIONS

**.c3** - C3

## DESCRIPTION

The **c3** file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

**Folding and Information Hiding**

Generic folding is enabled within the C3 file. The folds occur about functions defined with the **fn** keyword and a closing curly brace both at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The C3 mode offers beside folding as well support for displaying an outline of the functions of the C3 file for easy navigation using the **item-list**(3) command.

**Abbreviations**

The installation comes with a default abbreviation file for C3 file editing which can be used by writing short abbreviations like "**for**", "**if**", "**ef**" (else if), "**doc**" (documentation) or "**fn**" - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **myc3.emf**

## SEE ALSO

**abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).
Supported File Types

Ref: c3(9) File: m9typ053.9 Date: 2025/08/28

## SYNOPSIS

cbl - Cobol (85) files

## FILES

**hkcobol.emf** - Cobol (85) hook definition
**cobol.etf** - Cobol (85) template file

## EXTENSIONS

**.cbl** - Cobol file

## MAGIC STRINGS

**-*- cobol -*-**

Recognized by MicroEmacs and GNU Emacs, defines the file to be a cobol file.

## DESCRIPTION

The **cbl** file type templates provide simple hilighting of Cobol 85 files, the template provides minimal hilighting the language syntax.

## NOTES

No special language features are provided within the language syntax definition.

## BUGS

The Cobol hilight file is in it's infancy and a number of it's tokens may be misplaced.

## SEE ALSO

Supported File Types

Ref: cbl(9) File: m9typ028.9 Date: 2025/07/15

## SYNOPSIS

cfm - ColdFusion Markup Language.

## FILES

**hkcfm.emf** - ColdFusion file hook definition.
**hkcfms.emf** - ColdFusion script file hook definition.

## DESCRIPTION

The **cmf** file type template provides simple hilighting and indentation of Adobe Macromedia ColdFusion pages embedded within **html**(9) files. The keywords from **ColdFusion MX 7** are supported.

The template supports highlighting and indention of the tags `<cfxxx>` in addition to `<cfscript>` blocks and `<cfquery>` SQL blocks.

## SEE ALSO

**html**(9), Supported File Types

Ref: cfml(9) File: m9typ046.9 Date: 2022/06/12

## SYNOPSIS

Dart - Dart programming language

## FILES

**hkdart.emf** - Dart programming language hook definition
**dart.eaf** - Dart programming language abbreviation file
**dart.etf** - Dart programming language template file

## EXTENSIONS

**.dart** - Dart

## DESCRIPTION

The **dart** file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

**Folding and Information Hiding**

Generic folding is enabled within the Dart file. The folds occur about functions and classes and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Dart mode offers beside folding as well support for displaying an outline of the functions and classes of the Dart file for easy navigation using the **item-list**(3)

**Abbreviations**

The installation comes with a default abbreviation file for Dart file editing which can be used by writing short abbreviations like "**for**", "**if**", "**elif**" or "**class**" - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **my-dart.emf**

## SEE ALSO

**abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: dart(9) File: m9typ051.9 Date: 2025/08/28

## SYNOPSIS

tman, man - Tcl Doctools Markup File.

## FILES

**hkdtools.emf** - Tcl Doctools file hook definition
**dtools.eaf** - Tcl Doctools abbreviation file
**dtools.etf** - Tcl Doctools template file

## EXTENSIONS

**.tman** - Tcl Doctools files

## MAGIC STRINGS

**-[*!]- doctools**

MicroEmacs recognizes the magic string on the first line of the file used to indicate the file type. The files may be extension less or have an extension like .man and are still recognized.

## DESCRIPTION

The **Tcl Doctools** file hook provides a editing environment for Tcl doctools files. The file hook provides highlighting, folding of section, document outline using item-list and abbreviations.

File recognition is performed using the standard file extension **.tman**.

### Abbreviations

The installation comes with a default abbreviation file for Typst code editing which can be used by writing short abbreviations like "**se**" for a section header, "**ex**" for an example block, "**dl**" for a definition list, or "**li**" for an item list - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the Tcl doctools files. The folds occur about the section lines. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the section headers of the file are displayed in the *item-list*.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close list items

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **adoc**(9), **md**(9), **wiki**(9)

Supported File Types

Ref: doctools(9) File: m9typ070.9 Date: 2025/11/01

## SYNOPSIS

dot - Graphviz Dot Diagram Language File.

## FILES

**hkdot.emf** - Graphviz Dot diagram file hook definition
**dot.eaf** - Graphviz Dot diagram abbreviation file
**dot.etf** - Graphviz Dot diagram template file

## EXTENSIONS

**.dot** - Graphviz dot file

## DESCRIPTION

The **Graphviz** file macro provides simple hilighting of Graphviz dot diagram files.

File recognition is performed using the standard file extension **.dot**.

### Editing Environment

Beside of syntax highlighting the file hook offers as well abbreviations, folding and support for displaying an outline of the subgraphs of the Dot file script for easy navigation using the **item-list**(3) command. Further is is possible to expand abbreviations either via short strings using the **expand-abbrev-handle**(3) command and visual inserting abbreviations using the mouse by the **abbrev-list**(3) macro.

### Abbreviations

The installation comes with a default abbreviation file for Graphviz dot code editing which can be used by writing short abbreviations like "**gr**" (graph) "**dg**" (digraph), "**sg**" (subgraph) or **nd**" (node) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the Dot files. The folds occur about the **subgraph** keywords located on the left-hand margin at the beginning of the line and the closing curly brace for this subgraph. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the key constructs of the file may be listed in the *item-list*.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close list items

### NOTES

The hilighting is typically extended using a file **mydot.emf**.

### BUGS

No serious known so far. Indenting after the opening curly brace is not working sometimes.

### SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3),

Supported File Types

Ref: dot(9) File: m9typ056.9 Date: 2025/08/28

## SYNOPSIS

e, eu, ex, exw - Euphoria files

## FILES

**hkeuphor.emf** - Euphoria hook definition
**euphor.eaf** - Euphoria abbreviation file
**euphor.etf** - Euphoria template file.

## EXTENSIONS

**.e**, **.eu**, **.ex**, **.exw** - Euphoria file

## DESCRIPTION

The **Euphoria** file type mode provides simple hilighting of Euphoria files, abbreviations, folding and an item-list.

### Abbreviations

The installation comes with a default abbreviation file for Euphoria file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**fu**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Auto Layout

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

### Folding and Information Hiding

Generic folding is enabled within the Euphoria file. The folds occur about functions, procedures and types as long as these declarations are on one line and ending at a line starting with the **end** keyword. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Euphoria mode offers beside folding as well support for displaying an outline of the functions, macros, types and constants for easy navigation using the **item-list**(3) command.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (#).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.

**f6** - List items.
**esc f6** - Close item list.

## NOTES

The hilighting and other features are typically extended using a file **myeuphor.emf**.

## BUGS

None reported so far.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: e(9) File: m9typ061.9 Date: 2025/08/28

## SYNOPSIS

ehf - MicroEmacs help file

## FILES

**hkehf.emf** - MicroEmacs help file.

## EXTENSIONS

**.ehf**, ***help***

## DESCRIPTION

The **ehf** file type template performs the hilighting of the help file. The **ehf** file is a computer generated file and uses special embedded text markers to indicate the required color scheme.

The macro file includes special macros to locate help information.

## SEE ALSO

**help**(2).

Supported File Types

Ref: ehf(9) File: m9typ001.9 Date: 2025/08/28

## SYNOPSIS

emf - MicroEmacs Macro File

## FILES

**hkemf.emf** - MicroEmacs Macro File hook definition
**eaf.etf** - Abbreviation file
**emf.etf** - Template file

## EXTENSIONS

**.emf** - MicroEmacs Macro File

## DESCRIPTION

The **emf** file type template handles the hilighting of MicroEmacs macro files.

**General Editing**
On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**
The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**
The indentation mechanism is enabled which performs performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

**Tags** A C-tags file may be generated within the editor using the **Tools** -> **Emf-Tools** -> **Create Tag File**. **find-tag**(2) takes the user to the file using the tag information.

**Folding and Information Hiding**
Generic folding is enabled within the **emf** files. The folds occur about **define-macro** and **!emacro** text located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**
The Emf macro mode offers beside folding as well support for displaying an outline of the functions and help definitions of the Macro file for easy navigation using the **item-list**(3) command.

**Abbreviations**
The installation comes with a default abbreviation file for MicroEmacs macro file editing which can be used by writing short abbreviations like "**!w**" (while), "**!i**" (if) or "**de**" (define-macro)- and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Short Cuts**
The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c esc esc** - Command complete.
**A-C-i** - Restyle the current region.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - List items.
**esc f6** - Close item list,

## BUGS

No bugs reported

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **emftags**(3f), **expand-abbrev-handle**(3), **find-tag**(2), **indent**(2), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: emf(9) File: m9typ002.9 Date: 2025/08/28

## SYNOPSIS

erf - MicroEmacs registry file

## FILES

**hkerf.emf** - MicroEmacs registry file.

## EXTENSIONS

**.erf**, ***registry***

## DESCRIPTION

The **erf** file type template performs the hilighting of the registry file.

**Hilighting**

The hilighting features allows components of the language to be differentiated and rendered in different colors.

**Auto Layout**

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

## SEE ALSO

**list-registry**(2), **erf**(8).
Supported File Types

Ref: erf(9) File: m9typ021.9 Date: 2025/08/28

## SYNOPSIS

erl - Erlang programming language

## FILES

**hkerlang.emf** - Erlang programming language hook definition
**erlang.eaf** - Erlang programming language abbreviation file
**erlang.etf** - Erlang programming language template file

## EXTENSIONS

**.hx** - Erlang

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logicals, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph.

### Folding and Information Hiding

Generic folding is enabled within the Erlang file. The folds occur about functions definitions as long as they start with a lowercase letter and and final minus-greater symbols on that line and a new line starting with lowercase letters. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Erlang mode offers beside folding as well support for displaying an outline of the functions of the Erlang file for easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Erlang file editing which can be used by writing short abbreviations like "**if**", "**ca**" (case) or "**fu**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**NOTES**

The hilighting or the addition of new item list matches is typically extended using a file **myerlang.emf**

**SEE ALSO**

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

## SYNOPSIS

f, f77, f90 - Fortran files

## FILES

**hkf90.emf** - Fortran hook definition
**f90.etf** - Fortran 90 template file.
**f.etf** - Fortran (77) template file.
**f90.eaf** - Fortran 90 abbreviation file.

## EXTENSIONS

**.f** - Fortran file
**.f77** - Fortran 77 file
**.f90** - Fortran 90 file

## DESCRIPTION

The **f90** file type templates provide simple hilighting of Fortran 77 and Fortran 90 files, the template provides minimal hilighting of both language syntaxes, which are overloaded into the same file.

The major difference between the file types, apart from the new reserved words, is the comments. In Fortran 90 comments are introduced with **!**, while the other types use a **c** in column 0.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow functions, subroutines, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**

The indentation mechanism is enabled which performs performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

**Folding and Information Hiding**

Generic folding is enabled within the **f90** files. The folds occur about **program**, **function** or **procedure** keywords and their **end** text located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Fortran mode offers beside folding as well support for displaying an outline of the functions, subroutine and program definitions of the Fortran file for easy navigation using the **item-list**(3) command.

**Abbreviations**

The installation comes with a default abbreviation file for MicroEmacs macro file editing which can be used by writing short abbreviations like "**wh**" (while), "**if**" (if),

"**fu**" (function) or "**pg**" (program) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c esc esc** - Command complete.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - List items.
**esc f6** - Close item list,

**Executing, Formatting and Linting Fortran Code**

The Fortran file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools, compilers which are currently supported are "gfortran" as compilers and linter and "fprettify" as formatters. The formatter can be installed either using your standard package manager system wide or, for the current user, using the Python package manager "pip". Here an example on how to install fprettify using pip:

```
pip install fprettify --user
```

## BUGS

The Fortran hilight file is in it's infancy and a number of it's tokens may be misplaced.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **indent**(2), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

## SYNOPSIS

fu - Fusion transpiler programming language

## FILES

**hkfusion.emf** - Fusion programming language hook definition
**fusion.eaf** - Fusion programming language abbreviation file
**fusion.etf** - Fusion programming language template file

## EXTENSIONS

**.fu** - Fusion

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logicals, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Fusion file. The folds occur about functions and classes definition and their opening brace and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Fusion mode offers beside folding as well support for displaying an outline of the functions, classes and enums of the Fusion file for easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Fusion file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**Executing, Formatting and Linting Fusion Code**

The Fusion file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools, compilers which are currently supported are using the ”fut” command line transpiler and the ”astyle” source code formatter in C-sharp mode. For executing the transpiled code the python3 interpreter is used per default

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **myfusion.emf**

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **java**(9), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: fusion(9) File: m9typ076.9 Date: 2026/05/18

## SYNOPSIS

fvwm, fvwmrc - FVWM Window manager configuration files

## FILES

**hkfvwm.emf** - FVWM configuration file hook definition

## EXTENSIONS

**.fvwm**, **.fvwmrc** - FVWM configuration file

## MAGIC STRINGS

**-!- fvwm -!-**

The embedded fvwm string may be used with later versions of fvwm which use a different file extension to force the hilighting of the file.

## DESCRIPTION

The **fvwm** file type template provides simple hilighting of the FVWM files, the template provides minimal hilighting.

## BUGS

None reported.

## SEE ALSO

Supported File Types

Ref: fvwm(9) File: m9typ016.9 Date: 2022/06/12

## SYNOPSIS

go - Go programming language templates

## FILES

**hkgo.emf** - Go programming language hook definition
**go.eaf** - Go programming language abbreviation file
**go.etf** - Go programming language template file

## EXTENSIONS

**.go** - Go

## DESCRIPTION

The **go** file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Go language file. The folds occur about functions starting with the `func` keyword and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Go mode offers beside folding as well support for displaying an outline of the functions and types and modules of the Go file for easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Go file editing which can be used by writing short abbreviations like "**for**", "**if**", "**elif**" or "**func**" - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of

abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**Executing, Formatting and Linting Go Code**

The Go file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools, compilers which are currently supported are just the Go sub-commands usable for this task so ”go run”, ”go fmt” and ”go vet”.

## NOTES

The hilighting or the item-list matches are typically extended using a file **mygo.emf**

## SEE ALSO

**abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: go(9) File: m9typ049.9 Date: 2025/08/28

## SYNOPSIS

scala - Groovy programming language

## FILES

**hkgroovy.emf** - Groovy programming language hook definition
**groovy.eaf** - Groovy abbreviation file
**groovy.etf** - Groovy template file

## EXTENSIONS

**.gvy, .gy, .gsh, .groovy** - Groovy

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, functions, traits, classes, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Groovy file. The folds occur about lines with leading lowercase characters and an open curly brace at the end of a line and are continued until the next line with an closing curly brace. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Groovy mode offers beside folding as well support for displaying an outline of the functions, objects, classes and traits. These can be explored with easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Groovy file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hyphens (–).
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**NOTES**

The hilighting or the addition of new item list matches is typically extended using a file **my-groovy.emf**.

**SEE ALSO**

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **time**(2m).

Supported File Types

Ref: groovy(9) File: m9typ075.9 Date: 2025/12/29

## SYNOPSIS

hx - Haxe transpiler programming language

## FILES

**hkhaxe.emf** - Haxe programming language hook definition
**haxe.eaf** - Haxe programming language abbreviation file
**haxe.etf** - Haxe programming language template file

## EXTENSIONS

**.hx** - Haxe

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logicals, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Haxe file. The folds occur about functions, classes, enums and interface etc definition as long as they start with a lowercase letter and and a final opening brace on that line and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Haxe mode offers beside folding as well support for displaying an outline of the functions, classes, enums, typedef and interface definitions of the Haxe file for easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Haxe file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **my-haxe.emf**

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: haxe(9) File: m9typ078.9 Date: 2026/03/04

## SYNOPSIS

hs - Haskell programming language

## FILES

**hkhask.emf** - Haskell programming language hook definition
**hask.eaf** - Haskell programming language abbreviation file
**hask.etf** - Haskell programming language template file

## EXTENSIONS

**.hs** - Haskell

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Haskell file. The folds occur about functions definition containing the *::* character sequences are continued until the next empty line without any character except a newline. The macro **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Haskell mode offers beside folding as well support for displaying an outline of the functions, if the function declaration lines contain the *::* characters. The functions can be explored with easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Haskell file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hyphens (–).
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **myhask.emf**.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **time**(2m).

Supported File Types

Ref: hs(9) File: m9typ071.9 Date: 2025/11/09

## SYNOPSIS

html - HyperText Markup Language File.

## FILES

**hkhtml.emf** - HyperText Markup Language file hook definition
**html.eaf** - HyperText Markup Language abbreviation file
**html.etf** - HyperText Markup Language template file

## EXTENSIONS

**.htm**, **.html** - HyperText Markup Language File.
**.htp** - *[Special]* Super HTML Preprocessor file.
**.hts** - *[Special]* Super HTML file.

## MAGIC STRINGS

**<html>**

MicroEmacs recognizes the magic string on any of the first 4 lines of the file. The HTML files may be extension-less and are still recognized.

## DESCRIPTION

The **html** file type template provides simple hilighting of HTML files. Additionally, MicroEmacs is capable of rendering simple HTML files (without graphic content) into the current buffer and follow the hyper text links. The JASSPA HTML documentation may be viewed in this way.

**General Editing**

On creating a new file, a new header is automatically included into the file. HTML files may be edited or processed and rendered into the buffer. The **Use Author Mode** option in the **buffer-setup**(3) dialog determines the edit mode on loading a HTML file; when set to 'N' the page is rendered, 'Y' and the raw HTML file is presented. The default state is 'Y'.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Abbreviations**

The installation comes with a default abbreviation file for HTML file editing where the list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Navigation via Item List**

The Perl mode offers beside folding as well support for displaying an outline of the subroutines for easy navigation using the **item-list**(3) command.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-h** - Help information.
**C-c C-a** - Toggle the HTML author status.

**html-spell-check-word** - spell check the current word.
**f6** - List items.
**esc f6** - Close item list,

**Rendered Mode**

In the non-author mode, the HTML file is extracted and rendered to the buffer. The hypertext links may be followed by selecting them with the mouse or using the <RETURN> key.

The rendered mode is typically used to check HTML text after it has been authored from the editor. The rendered mode only caters for regular HTML 2.0 text. It does not handle tables or frames etc. *(use a browser)*.

Toggling between rendered and authoring mode, then the buffer should be killed as the translation is only performed when the file is read.

The non-author mode can be permanently enabled by setting the **Use Author Mode** option in the **buffer-setup**(3) dialog to 'N'. When set to N any HTML files loaded are automatically processed, and rendered according to their HTML content.

## NOTES

The print driver may be used to generate HTML from the contents of the buffer. Select the printer destination as *buffer*, and *HTML* as the driver. The buffer being printed is converted to HTML and dumped in the print buffer.

**.hts** and **.htp** are computer generated extended HTML files used in the MicroEmacs documentation system.

## BUGS

None reported.

## SEE ALSO

**abbrev-list**(3), **item-list**(3), **print-buffer**(2), **buffer-setup**(3).
Supported File Types

Ref: html(9) File: m9typ025.9 Date: 2025/08/28

## SYNOPSIS

imakefile - Make file

## FILES

**hkimake.emf** - Imakefile hook definition
**imake.etf** - Template file

## EXTENSIONS

**Imakefile**, **imakefile** - Imakefiles.

## DESCRIPTION

The **Imakefile** file type template handles the hilighting of the Imakefile files.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

By default, `TAB`'s are enabled as this is the syntactical feature of the file.

**Hilighting**

The hilighting emphasizes the keywords and comments within the Imakefile.

## BUGS

No attempt is made to hilight any embedded shell commands.

## SEE ALSO

**makefile**(9), **time**(2m).
Supported File Types

Ref: imakefile(9) File: m9typ006.9 Date: 2022/06/12

## SYNOPSIS

info - GNU Info File.

## FILES

**hkinfo.emf** - Info macro file.

## EXTENSIONS

No fixed extension, the root of the *info* tree is specified by `$INFOPATH`/dir. The default search paths on different platforms are:

`c:/info` - MS-DOS and MS-Windows (all).
`/usr/local/info` - All UNIX platforms.

## DESCRIPTION

The GNU *info* files are handled by the command **info**(3) which starts the info reader. This reads the initial info file `dir` and initializes the *info* file traversal. Where the *info* directory is not in the aforementioned locations then the `$INFOPATH` environment variable should specify the base directory.

The standard *info* navigation keys are in effect within the *info* buffers. The *mouse* may also be used to select the next info page.

## BUGS

There is no support within MicroEmacs to regenerate the *info* tags and indexes.

## SEE ALSO

**info**(3).

Supported File Types

Ref: info(9) File: m9typ027.9 Date: 2025/08/28

## SYNOPSIS

ini, hpj, reg, rgy - MS-Windows initialization and registry files
toml - Tom's obvious minimal language file

## FILES

**hkini.emf** - MS-Windows initialization and registry files.

## EXTENSIONS

**.ini** - MS-Windows Initialization File
**.hpj** - MS-Windows Help Project File
**.reg** - Registry File
**.rgy** - (Other) registry File
**.toml** - Tom's Obvious Minimal Language File

## DESCRIPTION

The **ini** file type templates provide simple hilighting of MS-Windows initialization and registry files. The file format is similar to a number of other registry type files and to toml configuration files which are also over-loaded into the same template.

**Hilighting**

The template provides minimal hilighting , but allows the different components of the file to be differentiated.

**Folding and Information Hiding**

Generic folding is enabled within the ini files. The folds occur about lines with leading square brackets **[**...**[** located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

## BUGS

None reported.

## SEE ALSO

**bat**(9).

Supported File Types

Ref: ini(9) File: m9typ008.9 Date: 2025/07/05

## SYNOPSIS

java - Java programming language templates

## FILES

**hkjava.emf** - Java programming language hook definition
**java.eaf** - Java programming language abbreviation file
**java.etf** - Java programming language template file

## EXTENSIONS

**.java**, **.jav** - Java

## DESCRIPTION

The **java** file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Abbreviations

The installation comes with a default abbreviation file for Java file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Auto Layout

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Java files. The folds occur about braces {...} located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region. Note that folding does not operate on K&R style code, with the trailing open brace.

### Navigation via Item List

The Java mode offers beside folding as well support for displaying an outline of the functions and classes for easy navigation using the **item-list**(3) command.

**Tags** A c-tags file may be generated within the editor using the **Tools** -> **Java-Tools** -> **Create Tag File**. **find-tag**(2) takes the user to the file using the tag information.

On invoking the tag generator then the user is presented with a dialog box which specifies the generation option of the tags file. The base directory of the tags file search and tagging options may be specified to locate all of the definitions within the code space.

The **tags** file is extremely useful where the user is dealing with inherited source code spread over multiple directories. Generation of a recursive tag file with all searching options enabled allows labels to be located extremely rapidly (certainly faster than IDE environments).

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**rcs-file**(2) is automatically invoked if an RCS file is detected, the normal check-in/out operations may be performed through the editor.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting is typically extended using a file **myjava.emf**.

## SEE ALSO

**c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **find-tag**(2), **item-list**(3), **javatags**(3f), **rcs-file**(2), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: java(9) File: m9typ018.9 Date: 2025/08/28

## SYNOPSIS

j, jl - Julia files

## FILES

**hkjulia.emf** - Julia hook definition
**julia.eaf** - Julia abbreviation file
**julia.etf** - Julia template file.

## EXTENSIONS

**.jl** - Julia file

## DESCRIPTION

The **Julia** file type editing provides simple hilighting of Julia files, abbreviations, folding and an item-list.

### Abbreviations

The installation comes with a default abbreviation file for Julia file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**fu**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Auto Layout

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

### Folding and Information Hiding

Generic folding is enabled within the Julia file. The folds occur about functions, macros and structs as long as these declarations are on one line and ending at a line starting with the **end** keyword. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Julia mode offers beside folding as well support for displaying an outline of the functions, macros and structs for easy navigation using the **item-list**(3) command.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (#).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list.

### Executing, Formatting and Linting Julia Code

The Julia file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The Julia packages which are currently supported for formatting and linting are ”Runic” and ”ReLint”. The procedure to install the ”runic” script as a standalone application on a Unix system is like this:

```
# Download the script into ~/.local/bin
curl -fsSL -o ~/.local/bin/runic https://raw.githubusercontent.com/fredrikekre/Runic.jl/refs/he
# Make the script executable
chmod +x ~/.local/bin/runic
# Verify the installation
runic --version
```

The procedure to install the ”ReLint” Julia linter in the julia shell is like this:

```
import Pkg
Pkg.add(url="https://github.com/RelationalAI-oss/ReLint.jl")
```

## NOTES

The hilighting and other features are typically extended using a file **myjl.emf**.

## BUGS

None reported so far.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: jl(9) File: m9typ057.9 Date: 2025/08/28

## SYNOPSIS

js - Javascript programming language

## FILES

**hkjs.emf** - Javascript programming language hook definition
**js.eaf** - Javascript programming language abbreviation file
**js.etf** - Javascript programming language template file

## EXTENSIONS

**.js** - Javascript language file

## DESCRIPTION

The **js** file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

**Folding and Information Hiding**

Generic folding is enabled within the Javascrip file. The folds occur about functions defined with the **function** keyword and a closing curly brace both at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Javascript offers beside folding as well support for displaying an outline of the functions and classes of the Javascript file for easy navigation using the **item-list**(3) command.

**Abbreviations**

The installation comes with a default abbreviation file for Javascript language file editing which can be used by writing short abbreviations like "**for**", "**if**", "**elif**" (else if) or "**func**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**List Items**

Item listing is available and the key constructs of the file may be listed in the *item-list*.

**Tags** A C-tags file may be generated within the editor using the **Tools** -> **Javascript-Tools** -> **Generate Tag File**. **find-tag**(2) takes the user to the file using the tag information.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **myjs.emf**.

## SEE ALSO

**abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **find-tag**(2), **item-list**(3), **jstags**(3f), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).
Supported File Types

Ref: js(9) File: m9typ055.9 Date: 2026/02/24

## SYNOPSIS

jst - JASSPA Structured Text

## FILES

**hkjst.emf** - Jasspa structured text file hook definition.

## DESCRIPTION

JASSPA Structured Text (**jst**) is an ASCII text format which includes some reserved text constructs that allow quite complex word processor type documents to be constructed with the minimal amount of additional annotation and which remains readable in its raw ASCII form. The motivation for doing this is to quickly construct documents and notes for publication whilst editing within MicroEmacs.

**JST** is a richer mark-up than the simpler **doc**(9) file type.

The main requirements of JST are:

- Easy to remember.
- Easily readable and can be highlighted well in MicroEmacs.
- Well defined, i.e. not ambiguous so formating errors can be generated.
- Extendable.
- May be automatically reformatted.
- Handle things like figure references and internal anchor and references.
- Html and LaTeX output, possibly rtf and text.

The syntax definition used by JST is defined in the next section.

The JST template extensions include a number of converters that will syntax check the text and then format and export the text to a new presentation format that may be imported into a word processor or used directly in the generated form.

## BASIC SYNTAX

The syntax of JST reserves basic character sequences which may be unambiguously interpreted into document styling. The basic syntax set allows a simple formatted document to be produced with the minimal amount of work from the author.

JST supports more advanced extensions which allow tables and hypertext tags to be defined, these are discussed in the **Advanced Syntax** section later.

### Quoting

To ensure that all characters may be input as typed then there is no single escape character. The character sequence **&[**<*char*>**]** may be used to escape a character <*char*>. Our experience is that escaping is not usually necessary.

### Basic Styling

The basic character styling is performed with a pair of escape characters at the start and end of a string sequence. The style end token does not have to appear on the same line as the start token and an error is reported if the sequence is not closed properly. Composite styles may be built buy combining the different tokens by inclusion of a new style within another style block. Where styles are nested then they should be closed in the reverse order that they were opened.

`""Raw Text""`
: A pair of double quotes surrounding a text block is treated as verbatim and is not subject to escaping, that is the character format characters are not translated.

****Bold Text****
: A pair of star characters denote a emboldened text region.

*//Italic Text//*
: A forward slash pair denotes an italic text region.

__Underline Text__
: A underscore pair denotes an underlined region.

`@@Mono Text@@`
: A pair of at symbols denotes a fixed text region rendered in a fixed width font.

“Superscript Text“
: A pair of grave accents denotes superscript text region.

,,Subscript Text,,
: A pair of commas denotes a subscript text region.

Styles may be nested i.e. `//Italic **Bold Italic** Italic Again//` To get raw text with a double quote use: `""raw with a &["]" in it""`

### Styling Paragraphs and Blocks

Blocks of text may be demarked by a single token at the start of the line. A exclamation mark '`!`' at the start of a paragraph may be followed by one or more style letters to enable the style for the paragraph, for consistency with the in-line tokens then double characters may be used.

```
!* As an example this baragraph is bold as it started with a '!*'.
```

produces

**As an example this baragraph is bold as it started with a '!*'.**

The exclamation mark may be followed by multiple different styles which are applied together. e.g.

```
!"@ This paragraph  started with '!"@' so it is quoted and mono-spaced
style, i.e. @@, "" or ** is ignored.
```

You can enable a style over a block of text by using a '`!`' style definition with no further text to the right. The block is terminated by a '`!end!`' or '`>end>`' tag on its own line. It is recommended that you insert a blank line before and after the '`!end!`' definition so that it does not affect the filling of the following paragraph.

```
!end!
```

Paragraphs may be associated with a literal block of text from a different programming language. These are introduced with a '`>type>`' tag and end with a '`>end>`' tag. The tag types that are currently supported include:

`emf` - JASSPA MicroEmacs Macro file.
`pseudo` - Pseudo code block.
`bnf` - BNF block.
`c` - ANSI C code block.
`cpp` - C++ code block.
`perl` - Perl code block.
`tcl` - TCL code block.
`shell` - UNIX shell block.
`java` - Java code block.
`ini` - MS-DOS/Windows .ini code block
`php` - PHP code block.
`make` - Makefile fragment.

The aforementioned blocks support syntax language highlighting and indentation. As an example:

```
Some example 'C' code:
>c>
/* This is C code */
int main (int argc, char *argv[])
{
    printf ("Hello World\n");
    return 0;
}
>end>
```

### Justification of Paragraphs and Blocks

A paragraph is a block of text delimited by an empty line, by default all paragraphs are left justified. Where the layout of the paragraph is literal and should not be re-formatted then the paragraph should be proceeded by a '>' symbol e.g.

```
> This paragraph will not be reformatted
because it starts with a '>', the
output will respect the position of the new lines.
```

Paragraphs may be center justified with a leading '`>c`' e.g.

```
>c This is centered text
```

With centered text then the indentation of the '`>c`' centers the text from the indent position. e.g.

```
>c Note the initial indent level is important
as the text is
centered w.r.t. the remainder.
```

Text may be right justified with a leading '`>r`'.

### Indentation

Indentation is implicitly defined by the indentation of the first line of the paragraph, each paragraph is effectively independent with respect to determining the indentation level.

The indentation level is defined to be the the column of the first non-white-space character of the first line divided by the *Indent Width* (see indent-setup) rounded up. For example this the first non-white of this paragraph is the 'F' at column 4, with an indent width of 4 (the default) this is a an indent level of 1.

A line starting with 6 spaces so it has an indent level of 2 (6/4 == 1.5 which is rounded up to 2). Rounding up allows item lists to work with little effort, without introducing ambiguity, for example.
The indentation of a code fragment or literal text block are determined by the indentation of the initial style tag, for example:

```
    >"@
    >c>
    int myVariable ;
    >end>
>end>
```

The code block is indented from the left paragraph by the starting block indent.

### Headings

Headings may be defined and commence with a equals ('=') character as the first character of the line. The number of equals determines the heading level.

```
= Top Level Heading
== Sub-heading
=== Sub-sub-heading
```

The position of the heading may be controlled and may be left, centered or right justified using `l`, `c` or `r` respectively. Left justified is the default when not defined.

```
=c Centered Top Level Heading
==r Right justified sub-heading
```

### Item Lists

Items lists are bullet or number lists of items which are introduced by a *formatChar***.** to start the list, subsequent list items are introduced with a period '**.**' character only or optionally with the bullet character. Where *formatChar* identifies the type of list as follows:

- `*` - bullet list.
- `1` - Numeric item list.
- `a` - Lowercase alphabetic list.
- `A` - Uppercase alphabetic list.
- `i` - Roman numeral list.
- `I` - Upper case Roman numeral list.
- `:.` - Text item list.

Indentation and sub-paragraphs are supported within the paragraph. e.g.

```
 1. Start or a numbered list.

    Sub-paragraph of numbered list.

 2. 2nd item in numbered list

    *. Contained bullet list.

     . 2nd item of bullet list.

 . 3rd item in numbered list.
```

The text item, or definition, requires some further explanation. This allows a definition item to be defined (e.g. glossary of terms at start of document or note points within a document). These are defined at column 0 e.g.

```
:. MyDefinition

   Description of the term.
```

## ADVANCED SYNTAX

This section defiens the advanced syntax extensions of JST which allow more complex documents to be generated. These extensions may be used to author more complex documents for export.

### Tables

Quite complex tables may be supported through JST. A table is introduced with the closing square brace '**]** character at the start of the line which define the ghlobal properties and dimensions of the table. Each entry in the table is introduced by '**]** and is followed by the assigned content.
The initial table definition has the syntax

```
]crh <# columns> <# rows>
```

Where '**c**' and '**r**' are optional and center or right justify the table, '**h**' hides the table lines i.e. no border. The dimensions of the table then follow.
A simple definition of a 3x2 table might be defined as:

```
] 3 2

] Row 1 col 1 content.

] Row 1 col 2 content.

] Row 1 col 3 content.

] Row 2 col 1 content.

] Row 2 col 2 content.

] Row 2 col 3 content.
```

The content may include

> `*.` Use item lists
> `.` Like this
> `!c` Center text
> `!r` Right justify text
> *also embedded images, see later.*
> Columns and rows may be joined both horizontally and vertically by specifying the join in the cell definition. '**]>**' performs a horizontal cell join of two cells, a single '>' should be added for each additional cell so '**]>>**' joins 3 cells horizontally. A vertical join is similarly specified with '**]v**'.

```
] 3 3

]>> Single top row with 3 columns joined content.

]v 2 rows joined in the first column content.

]> 2nd row 2nd and 3rd column joined content.

] 3rd row 2nd column content.

] 3rd row 3rd column content.
```

### Special Characters and Symbols

Special characters are introduced with '**&[***special***]**' and are defined as follows:

`&[nl]` or `&[new-line]`
: Newline.

`&[np]` or `&[new-page]`
: New page. This is not supported in HTML or text.

`&[nf]` or `&[new-file]`
: New file, the effect is dependent on the output format:

| | |
|---|---|
| **html** | Starts a new html file, typical use of this is when the first file contains the TOC with links to the subsequent files. |
| **latex** | Just starts a new page, i.e. same as **[[np]]**. |
| **rtf** | Just starts a new page, i.e. same as **[[np]]**. |
| **text** | Ignored. |

`&[toc]`
: Inserts an automatically generated table of contents.

`---`
: Inserts a horizontal line. This must be the only text on the line.

### Tags and Hyperlinks

Most of the more complex formatting is performed with tags. The general format of a tag is defined as:

**&[***<tag-id><options> <args>***]**

If a argument contains a space or a '**]**' then it must be quoted, e.g.:

```
&[c this-is-okay]
&[c "this is okay"]
&[c "this [\"is\"] okay"]
```

A tag may support numerous options, the order of the options is not important but the order of their arguments is as specified in the docs, not the order given with the tag therefore **&[idw <image> <depth> <width>]** is still **&[iwd <image> <depth> <width>]**. The text that actually appears in any generated document should not appear in a tag as the spelling check should skip the contents of a tag. This means that a link should take the HTML form of encasing the text label with start and end tags rather than being in the tag itself. One exception would be the setting of a variable.

**Comment Tag**
: Inserts a comment.

  **&[c** *<0+args>***]**

  There may be any number of arguments, the whole tag is ignored. As the comment ends with a '**]**' and '**]**' char in the comment must be quoted, i.e.:

  ```
  &[c this is a comment with a "]" "char[23]" in it]
  ```

**Anchor Tag**
: Internal document reference.

**&[a** <*name*>**]**

An anchor tag can be placed anywhere and referenced by a link .

**Foreground and Background Color Tag**

Modifies the color of the text foreground and background.

**&[f** <*color*>**]** .... **&[/f]**

Sets the current foreground color where <*color*> is either a direct HTML format color definition (e.g. "`#ff0000`" for **red**) or a previously defined color name.

**&[b** <*color*>**]** .... **&[/b]**

Sets similarly sets the current background color.

A number of options are supported by the color tag as follows:

**&[fs** <*name*> <*color*>**]**
: Define color '<*name*>' to be '<*color*>', the <*name*> may be references in the **f/b** color references.

**&[fd** <*color*>**]**
: Sets the default foreground color to <*color*>.

**&[bd** <*color*>**]**
: Sets the default background color to <*color*>.

**Link Tag**

Link to another part of the document or URL.

**&[l** <*url*>**]** .... **&[/l]**

Creates a hyperlink with the text with '`....`' going to the <*url*>, The format of the url uses the standard HTML syntax, a link starting with a hash '`#`' is assumed to be local and a warning will be issued if the corresponding anchor point is not found. An error is generated if the **&[/l]** is missing.

Urls appearing in the text like '`http://www.jasspa.com`' are automatically turned into links. To avoid this quote the string, i.e. ""http://www.jasspa.com""

**Image Tag**

Insert an image in the document.

**&[i** <*image-name*>**]**

Inserts an image <*image-name*>, for example **&[i me.png]**

Additional options allow the depth and width of the image to be specified:

**&[idw** <*image-name*> <*depth*> <*width*>**]**

Where option **d** specifies the depth of image and **w** specifies the width of the image. e.g. `&[iw test.jpg 60]`. The size is given in pixels and the conversion variable '*image-ppi*' is used to convert is a physical paper size. The width may also be specified as a percentage, e.g. **50%**.

**Use Tag**

Insert a file at the current point.

**&[u** <*file-name*>**]**

Inserts <*file-name*> into the current location and processes it as if part of the original docuement. e.g. `&[u insert.jst]` inserts the file *insert.jst* at the current position in the file.

**Variable Tag**

Insert a variable.

**&[v** <*variable-name*>**]**

Inserts the current value of variable <*variable-name*>, variable names may only contain alphanumeric characters and '–'.

The variable supports a set of simple options that allow the variable value to be manipulated prior to insertion e.g. for handling numbers etc. The option syntax is defined as follows:

**&[vdinrs** <*variable-name*> <*value*> <*increment*> <*reference-name*>**]**

**d** Define the reference <*reference-name*> for the current value.

**i** Pre-increment the variable value by <*increment*> prior to insertion.

**n** Do not insert the variable value.

**r** Insert the <*reference-name*> value.

**s** Set the value to <*value*> first.

As an example, consider the following sequence to implement figure labels and reference them:

```
&[vsn figure 0]

&[i figure1.jpg]
Figure &[vid figure 1 figA] - my first figure

some more text

&[i figure2.jpg]
Figure &[vid figure 1 figB] - my second figure

To see  **figure &[vr figure  figA]** //my first  figure// and **figure
&[vr figure figB]** //my second figure//.

To see  **figure &[vr figure  figA]** //my first  figure// and **figure
&[vr figure figB]** //my second figure//.
```

An undefined variable has a value of "`ERROR`", it can be defined by using either option '**s**' or '**i**' in which case the value is `0`, i.e. the following can be used

```
= Chapter 1:  Will be 1 at the top of the doc
```

A set of special variables are automatically defined:

**title** The document title.

**author** The author

**image-ppi**

Image pixels per inch resolution setting, this is the resolution of the generated document in inches.

**Output Dump Tag**

Dump output text.

**&[o** <*out-type*> <*raw-text*>**]**

Dumps <*raw-text*> into the output if the target output type is <*out-type*>, where <*out-type*> can be **all**, **html**, **latex**, **rtf** and **text**.

**Dump Tag**

Dumps the tag for processing.

**[d** <*out-type*> <*text*>**]**

This command is similar to Output Dump Tag except this is done first so the <*text*> is processed, this can be used to define output specific formatting, e.g. (note that the \ at the end of the line is used here to show that the content should be placed on a single line).

```
&[d html "&[vsn large-font-start \"..html def..\"] \
          &[vsn large-font-end \"..html def..\"]"]
&[d latex "&[vsn large-font-start \"..latex def..\"] \
           &[vsn large-font-end \"..latex def..\"]"]
&[d text "&[vsn large-font-start \"\"]&[vsn large-font-end \"\"]"]
```

so in the main body of the text:

```
&[v large-font-start] Large Text &[v large-font-end]
```

This is a complex concept and is beyond most users requirements, but making this kind of thing possible means that users should be able to do whatever they need to.

## TEMPLATE

The template supports highlighting and indention including commands for exporting JST to another format:

**jst-to-htm**

Export the JST definition to an HTML file with **.htm** extension.

**jst-to-html**

Export the JST definition to an HTML file with **.htm** extension.

**jst-to-rtf**

Export the JST definition to an RTF file, for this export then the file name is **.rtf** but the content is HTML. **Word**(1) and **StarOffice**(1) know how to convert this when imported.

**jst-to-latex**

Convert to a LaTeX file format.

## FILES

JST is controlled by the macro files `hkjst.emf` which contains the basic template. `jstutl.emf` includes the utilities used for conversion to other formats.

## INTERNAL FORMAT

**JST** uses an internal neutral format which is used in the conversion process, this allows different back end output format drivers to be written against the neutral format without dealing with the input syntax. This intermediate format is not visible to the user and does not exist in the output. The internal format is defined as follows:

**\bF?**....**\bf?**
Font style change where **?** is **B**,**I**,**U**,**M**,**p**,**b**).

**\bDD\bDF** *<color>* **\bDB** *<color>* **\bdD**....**\bDd**
Foreground or background color change where *<color>* = hexadecimal value `RRGGBB`.
Note that **\bDF** *<c>* or **\bDB** *<c>* may be be omitted.

**\bIT** *<image-name>* **\biT**
Insert an image without scaling.

**\bIT** *<image-name>* **\bID** *<depth>* **\biD\bIW** *<width>* **\biW\biT**
Insert an image with scaling.

**\bTA** *<anchor-name>* **\btA**
A local anchor label.

**\bTL** *<link-url>* **\btL** *<link-text>* **\bTl**
URL Link.

**\bTV** *<variable-name>* _ *<variable-ref>* **\btV**
A variable name (temporary).

**\bSF** Start new file.

**\bSL** Start new line.

**\bSP** Start new page.

**\bSN** No-text symbol.

**\bST** Insert the table of content (temporary)

**\bPI** Indent level.

**\bPT** Indent tab level (used only in generated TOC).

**\bPB**....**\bpB**
Paragraph - justification both.

**\bPC**....**\bpC**
Paragraph - justification center.

**\bPF**....**\bpF**
Paragraph - File style block.

**\bPL**....**\bpL**
Paragraph - justification left.

**\bPN**....**\bpN**
Paragraph - justification none.

**\bPR**....**\bpR**
Paragraph - justification right.

**\bH#**....**\bh#**
Heading, e.g. **\bH1**....**\bh1**

**\bBI\bB?**...**\bb?\bB?**...**\bb?\bbI**
Item list start and end tags.

**\bBB**....**\bbB**
Bullet item.

**\bBN#\bBn**....**\bbN**
Item # in numbered item list.

**\bBU#\bBu**....**\bbU**
Item # in uppercase letter item list

**\bBL#\bBl**....**\bbL**
Item # in lowercase letter item list.

**\bBR#\bBr**....**\bbR**
Item # in uppercase roman numeral item list.

**\bBS#\bBs**....**\bbS**
Item # in lowercase roman numeral item list.

**\bCT**<*flags*>**\bCt**<*widths*>**\bct**....**\bcT**
Table - Entire table (contains rows).
<*flags*> can be any of **h**, **c** or **r**.
<*widths*> is an ME list `|no col|width table|width of col 1|width of col 2|...|`

**\bCR**....**\bcR**
Table - Row (contains columns)

**\bCC**....**\bcC**
Table - column (single cell)

## SEE ALSO

**doc**(9), Supported File Types

Ref: jst(9) File: m9typ047.9 Date: 2025/07/23

## SYNOPSIS

Kotlin - Kotlin programming language

## FILES

**hkkotlin.emf** - Kotlin programming language hook definition
**kotlin.eaf** - Kotlin programming language abbreviation file
**kotlin.etf** - Kotlin programming language template file

## EXTENSIONS

**.kt**, **.ktm**, **.ktx** - Kotlin

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Kotlin file. The folds occur about functions and classes definition and their opening brace and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Kotlin mode offers beside folding as well support for displaying an outline of the functions and classes of the Kotlin file for easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Kotlin file editing which can be used by writing short abbreviations like "**for**", "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

### NOTES

The hilighting or the addition of new item list matches is typically extended using a file **mykotlin.emf**

### SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **java**(9), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: kotlin(9) File: m9typ067.9 Date: 2025/09/28

## SYNOPSIS

latex - TeX Documentation

## FILES

**hklatex.emf** - TeX File hook definition
**latex.eaf** - Abbreviation file
**latex.etf** - Template file
**hkbibtex.emf** - TeX Bibliography Files

## EXTENSIONS

**.tex** - TeX Documentation
**.bib** - TeX Bibliography Files
**.sty** - TeX Style file

## DESCRIPTION

The **latex** file type template handles the hilighting of the TeX files. The hilighting is minimal, hilighting the key words and comments.

**General Editing**
On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**
The hilighting emphasizes the TeX embedded command strings and comments. No special recognition of the command strings is performed.

**Outline Hilighting**
The LaTeX content may be viewed with synthetic hilighting such that headers, text in bold and italic are displayed, removing the LaTeX control sequences.

**Folding and Information Hiding**
Generic folding is enabled, the folds occur about **\section{...}** and **\subsection{...}** constructs. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Abbreviations**
The installation comes with a default abbreviation file for LaTeX file editing which can be used by writing short abbreviations like "**sec**" (section), "**bf**" (textbf) or "**verb**" (verbatim environment)- and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Spelling**
Buffer spell checking is LaTeX aware.

**List Items**
Item listing is available and the key constructs of the file may be listed in the *item-list*.

**Buffer Commands**
**latex-verify** verifies the syntax of the (La)TeX from the current position forward, reporting braces that are not matched.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hash (%).
**C-c C-o** - Format a comment
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - List items.
**esc f6** - Close list items window.

The command **latex-compile** is available within the buffer which invokes an external process to build the text.

## BUGS

No bugs reported

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **item-list**(3), **time**(2m).
Supported File Types

Ref: latex(9) File: m9typ004.9 Date: 2026/02/25

## SYNOPSIS

lua - Lua files

## FILES

**hklua.emf** - Lua hook definition
**lua.eaf** - Lua abbreviation file
**lua.etf** - Lua template file.

## EXTENSIONS

**.lua** - Lua file

## DESCRIPTION

The **Lua** file type editing provides simple hilighting of Lua files, abbreviations, folding, item-list support and lua function tagging using the **luatags**(3f) command.

**Abbreviations**

The installation comes with a default abbreviation file for Lua file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**fu**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Auto Layout**

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

**Folding and Information Hiding**

Generic folding is enabled within the Lua file. The folds occur about functions as these declarations are starting at the beginning of one line and ending at a line starting with the **end** keyword. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Lua mode offers beside folding as well support for displaying an outline of the functions for easy navigation using the **item-list**(3) command.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (#).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list.

### Executing, Formatting and Linting Lua Code

The Lua file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The default commands and tools are "*luac*" for linting, "*lua*" for execting and the "*luafmt*" application for formatting. The procedure to install the "*luafmt*" application on a Unix using the npm package manager system is like this:

```
sudo npm install -g lua-fmt
```

If the application *"luafmt"* is not available, the macro **restyle-buffer**(3), will be used instead.

## NOTES

The hilighting and other features are typically extended using a file **mylua.emf**.

## BUGS

None reported so far but many features are added just recently and have not been tested throughly.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3), **luatags**(3f), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: lua(9) File: m9typ062.9 Date: 2025/09/11

## SYNOPSIS

m - GNU Octave / Matlab files

## FILES

**hkoctave.emf** - Octave hook definition
**octave.eaf** - Octave abbreviation file
**octave.etf** - Octave template file.

## EXTENSIONS

**.m** - GNU Octave / Matlab file

## DESCRIPTION

The **Octave** file type editing provides simple hilighting of GNU Octave / Matlab files, abbreviations, folding and an item-list.

### Abbreviations

The installation comes with a default abbreviation file for Octave file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**fu**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the Octave file. The folds occur about function declarations as long as these are at the beginning of the line and ending at a line starting with the **endfunction** keyword. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Octave mode offers beside folding as well support for displaying an outline of the functions for easy navigation using the **item-list**(3) command.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with comment symbols (#).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

### Executing, Formatting and Linting Octave Code

The Octave file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. For formatting and linting the *octave* interpreter itself is used using the *"-W"*

option for running the terminal version. As Octave does not support mentioning the filename in case of errors, currently there is no support to click on the error messages for and jump to the problematic code **file-exec**. For linting a trick is used catching the error message for the current Octave file. For formatting the code the Python tool *"matlab-formatter"* can be used. This tool can be installed on an Unix system like this:

```
## Download the script into ~/.local/bin
curl -fsSL -o ~/.local/bin/matlab-formatter \
  https://raw.githubusercontent.com/mittelmark/matlab-formatter-vscode/refs/heads/master/format

### Make the script executable
chmod +x ~/.local/bin/matlab-formatter

#### Verify the installation
matlab-formatter
```

## NOTES

The hilighting is typically extended using a file **mym.emf**.

## BUGS

Currenlty the next-line feature for interactive browsing of errors and warning for file-lint and file-exec does not work due to Octave limits on mentioning the filename for errors.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: m(9) File: m9typ058.9 Date: 2025/08/28

## SYNOPSIS

me - M4 Macro Processor

## FILES

**hkm4.emf** - M4 macro processor macro file.
**m4.etf** - M4 macro processor header template file.

## EXTENSIONS

**.m4**

## DESCRIPTION

The **M4 macro processor** template performs simple hilighting of **.m4** files. The file type is recognized by the standard extension only.

### Hilighting

The hilighting features allows components of the language to be differentiated and rendered in different colors.

## NOTES

The M4 hilighting is minimal, no other features have been implemented.

## SEE ALSO

Supported File Types

Ref: m4(9) File: m9typ032.9 Date: 2022/06/12

## SYNOPSIS

makefile - Make file

## FILES

**hkmake.emf** - Make file hook definition
**make.etf** - Template file

## EXTENSIONS

**Makefile**, **makefile**, **.mak** - Makefiles.

## MAGIC STRINGS

**-!- makefile -!-**

Recognized by MicroEmacs only, defines the file to be a makefile.

## DESCRIPTION

The **make** file type template handles the hilighting of the makefile files.

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

By default, `TAB`'s are enabled as this is the syntactical feature of the file.

### Hilighting

The hilighting emphasizes the keywords and comments within the makefile. No special support for Microsoft **nmake**(1) is provided because of the number of oddities in their implementation of make.

## BUGS

No attempt is made to hilight any embedded shell commands.

## SEE ALSO

**imakefile**(9), **time**(2m).
Supported File Types

Ref: makefile(9) File: m9typ005.9 Date: 2022/06/12

## SYNOPSIS

man - UNIX Manual page

## FILES

**hkman.emf** - UNIX manual page hook definition

## EXTENSIONS

**.man** - UNIX manual page file

## DESCRIPTION

The **man** provides the hilighting of UNIX manual pages, generally acquired through the **man**(3) command, via a pipe. *man* references within the displayed manual page may be accessed using the mouse in a hypertext fashion.

### Hilighting

The hilighting commands recognize the manual page *bold* and *underline* character sequences and transpose these into the appropriate character hilighting. The hilighting sequences are generally unpleasant because they also remove the characters for display.

The multitude of different platforms causes problems as different vendors produce different character sequences for *bold/italic* text, hence on some platforms it may be necessary to add additional hilighting rules to cater for any local variations.

### Short Cuts

Selecting a link node within the manual page using the mouse (i.e. a reference to another manual page) then MicroEmacs attempts to find the manual page in the text and invokes **man**(3) to render the page. This provides a crude hyper text mechanism simply using the manual page information itself.

The **man-clean**(3) command can be used to remove all of the hilighting characters from the current manual page. This is the typical method of reducing a manual page to plain text.

## SEE ALSO

**man**(3), **man-clean**(3),
Supported File Types

Ref: man(9) File: m9typ022.9 Date: 2025/08/28

## SYNOPSIS

md - Markdown Markup File.

## FILES

**hkmd.emf** - Markdown file hook definition
**hkmdview.emf** - Markdown browser hook definition
**md.eaf** - Markdown abbreviation file
**md.etf** - Markdown template file
**qd.etf** - Quarkdown template file
**rmd.etf** - R-Markdown template file

## EXTENSIONS

**.md**, **.qd** **.rmd**, **.tmd** - Markdown, R-Markdown and Quarkdown files

## DESCRIPTION

The **Markdown** file hook provides a editing environment for Markdown files. The file hook provides highlighting, folding of section, document outline using item-list and abbreviations. Fenced code blocks within triple backticks can be highlighted using the declared code block language. Further the **mdview** macros allows to browse a set of related Markdown files in an interactive way.

File recognition is performed using the standard file extension **.md**, **.tmd**, **.rmd** or **.qd**. The **.qd** file extension is reserved for Quarkdown files which provided a few more highlightings for quaurkdown commands at the beginning of a line starting with a dot.

### Abbreviations

The installation comes with a default abbreviation file for Markdown code editing which can be used by writing short abbreviations like "**h1**" for a default level 1 header, "**bo**" for bold text, "**url**" for an URL, or "**cc**" for a code chunk - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the Markdown files. The folds occur about the level 2 headers, so headers with two hash symbols at the beginning of a line. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the level 1 and 2 headers of the file are displayed in the *item-list*.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**f2** - (un)fold the current region
**f3** - (un)fold all regions

**f6** - list items
**esc f6** - close list items

**Code Chunk Highlighting**

Inside code chunks as well syntax hilghlighting for a few programming languages like Tcl (tcl), R (r), GraphViz Dot (dot), Python (py), SQL (sql), MicroEmacs macro code (emf)or Javascript (js) is available. Here an example which should be highlighted within a Markdown file using the R file hook:

```
```{r}
x = 1
test = function () {
    print("Test")
}
```
```

**mdview macro**

**mdview** provides a mechanism to display a set of related Markdown files in a folder in an interactive way within MicroEmacs. On invoking **mdview** if the user is not currently editing a Markdown file the user is prompted for the name of a Makrdown file to display:

```
Open file:
```

The command the loads the Markdown file and calls the **mdview**(3f) macro, entering browse mode for the file.

Another Markdown file may be selected in the generated browse page by either moving the cursor to the link and pressing return or double clicking on it with the left mouse button. MicroEmacs will then attempt to load and display the selected Markdown file. If the file does not yet exists the user is asked to create a new file.

## SEE ALSO

**abbrev-list**(3), **adoc**(9), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **mdview**(3f), **typst**(9), **wiki**(9).

Supported File Types

Ref: markdown(9) File: m9typ060.9 Date: 2026/08/26

## SYNOPSIS

MetaFont/MetaPost - Meta Font and Post File.

## FILES

**hkmeta.emf** - MetaFont/MetaPost file hook definition

## EXTENSIONS

**.mf** - MetaFont file
**.mp** - MetaPost file

## DESCRIPTION

The **Meta** file type template provides simple hilighting of **MetaFont** (`.mf`) and **MetaPost** (`.mp`) files, the template provides minimal hilighting. The same hilighting definition is used for both file types.

File recognition is performed using the standard file extensions.

## NOTES

JASSPA have no idea as to the state of this file hook definition.

## SEE ALSO

Supported File Types

Ref: MetaFont(9) File: m9typ035.9 Date: 2022/06/12

## SYNOPSIS

nim - Nim programming language

## FILES

**hknim.emf** - Nim programming language hook definition
**nim.eaf** - Nim programming language abbreviation file
**nim.etf** - Nim programming language template file

## EXTENSIONS

**.nim** - Nim

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Nim file. The folds occur about functions and classes definition and their opening brace and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Nim mode offers beside folding as well support for displaying an outline of the functions and classes, if the classes package is installed of the Nim file for easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Nim file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hashes (#).
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**Executing, Formatting and Linting Nim code**

The Nim file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The default commands and tools are

```
nim compile --verbosity:0
```

for linting,

```
nim run --verbosity:0
```

for executing and the **nimpretty** application for formatting. The user may install the **nph** application for better formatting on a Unix system using the nim package manager **nimble** as follows:

```
nimble install nph
```

If the application **nph** is not available, the default **nimpretty** will be used instead.

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **mynim.emf**. Support for classes can be added by adding the **classes** package using the nimble package manager like this:

```
nimble install classes
```

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3), **time**(2m).

Supported File Types

Ref: nim(9) File: m9typ069.9 Date: 2026/02/25

## SYNOPSIS

0-9, tni, so - UNIX t/nroff file.

## FILES

**hknroff.emf** - UNIX t/nroff file.
**nroff.etf** - UNIX t/nroff template file
**ntags.emf** - t/nroff tags generator macro definition.

## EXTENSIONS

**1**, **2**, **3**, **4**, **5**, **6**, **7**, **8**, **9** - UNIX t/nroff files.
**tni**, **so** - UNIX t/nroff include files.
**sm** - *[Special]* Superman t/nroff file.

## MAGIC STRINGS

**-*- nroff -*-**

Recognized by GNU and MicroEmacs. Denotes a t/nroff type file, may be used in **.1/.9**, **.tni** and **.so** files.

## DESCRIPTION

The **nroff** file type templates handle UNIX n/troff type files.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Tags**

A C-tags file may be generated within the editor using the **Tools** -> **Nroff-Tools** -> **Create Tag File**. **find-tag**(2) takes the user to the file using the tag information. The tags are generated using the **.XI** keyword, this may not be standard for all nroff pages.

**Folding and Information Hiding**

Generic folding is enabled within the C and C++ files. The folds occur about sections **.S**[**HS**]....**S**[**HS**] located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region. Note that folding does not operate on K&R style code.

**Tools**

The nroff buffer provides a facility to toggle the hilighting of the buffer on and off. If font change inserts are used (**fB**, **fR**, etc), then the enclosed **bold** and *italic* regions are hilighted, hiding the escape sequences. This allows the nroff text to be viewed in a more representative rendered format.

The local buffer command **aman** invokes, the following command sequence (defined in `hkman`) to render a nroff **man** file into a buffer window;-

```
soelim <file> | tbl -TX | neqn | nroff -man | col -x
```

The command **tex2nr** attempts to convert a **latex**(9) file into an nroff file. The *latex* escape sequences are converted into their nroff equivalents. The command is only made available when an Nroff file is loaded (as the command is defined in the `hknroff.emf` file).

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-s** - Insert a font size escape character **S0**.
**C-c C-r** - Insert a roman font escape character **fR**.
**C-c C-b** - Insert a bold font escape character **fB**.
**C-c C-i** - Insert a italic font escape character **fI**.
**C-c C-c** - Insert a courier font escape character **fC**.
**C-c C-p** - Insert a previous font escape character **fP**.
**esc o**, **esc q** - **fill-paragraph**(2) fills paragraph to next `.XX` command.
**C-c b** - Bold region by inserting **fB** .. **fR**.
**C-c c** - Courier region by inserting **fC** .. **fR**.
**C-c c** - Italic region by inserting **fI** .. **fR**.
**C-c C-h** - Toggle hilighting on/off.
**C-c C-&** - Adds nroff padding **f&** about words.
**C-x C-&** - Removes nroff padding **f&** about words.
**esc h** - Nroff help.

**f2** - (un)fold the current region
**f3** - (un)fold all regions

## BUGS

The nroff language template is heavily biased towards the **man** macros only and includes all of the extension macros used for generating the JASSPA hypertext documentation.

The template in the current form has been used entirely by JASSPA in generating all of the documentation (**HTML**, **Winhelp**, **ehf**, **PostScript**) used by MicroEmacs. It does not include all of the troff/nroff keywords, or keywords for any of the standard macro packages.

The JASSPA documentation preparation tools are proprietary and have not been made publicly available.

## SEE ALSO

**collapse-all**(3), **collapse-current**(3), **fill-paragraph**(2), **find-tag**(2), **ntags**(3f), **time**(2m).
Supported File Types

Ref: nroff(9) File: m9typ026.9 Date: 2025/08/28

## SYNOPSIS

p, pas - Pascal files

## FILES

**hkpascal.emf** - Pascal hook definition
**pascal.eaf** - Pascal abbreviation file
**pascal.etf** - Pascal template file.

## EXTENSIONS

**.p**, **.pas** - Pascal file

## DESCRIPTION

The **pascal** file type template provides simple hilighting of Pascal files, the template provides minimal hilighting.

### Abbreviations

The installation comes with a default abbreviation file for Pascal file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**be**" (begin/end) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the Pascal file. The folds occur about functions, procedures, record or classes as long as these declarations are on one line and and at a line starting with the **End;** keyword. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Pascal mode offers beside folding as well support for displaying an outline of the functions, procedures, records and classes or objects for easy navigation using the **item-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**Executing, Formatting and Linting Pascal Code**

The Pascal file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools and compilers which are currently the Free Pascal provided ones, so ”fpc” as compiler, used as well as linter and ”ptop” the formatter which comes with the Free Pascal installation.

## BUGS

None reported.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

## SYNOPSIS

perl - Practical Extraction and Report Language File.

## FILES

**hkperl.emf** - Practical Extraction and Report Language file hook definition
**perl.etf** - Practical Extraction and Report Language header template file
**perl.eaf** - Practical Extraction and Report Language abbreviation file

## EXTENSIONS

**.pl**, **.pm** - Perl file

## MAGIC STRINGS

**#![ \t]*/.*perl**

MicroEmacs recognizes the magic string on the first line of the file used to locate the executable. The Perl files may be extension-less and are still recognized.

**# -*- perl -*-**

MicroEmacs recognizes the standard GNU Emacs magic string on the first line of the file. The Perl files may be extension-less and are still recognized.

## DESCRIPTION

The **perl** file type template provides the hilighting, indentation and tools definitions for a perl file.

File recognition is performed using the standard file extension **.pl**, **.pm** or by the magic string.

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Abbreviations

The installation comes with a default abbreviation file for Perl file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**sub**" (subroutine) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Auto Layout

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

### Folding and Information Hiding

Generic folding is enabled within perl files. The folds occur about *sub*routine definitions located on the left-hand margin and their closing brace which is located at

the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Perl mode offers beside folding as well support for displaying an outline of the subroutines for easy navigation using the **item-list**(3) command.

**Tags** A C-tags file may be generated within the editor using the **Tools** -> **Perl-Tools** -> **Create Tag File**. **find-tag**(2) takes the user to the file using the tag information.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c esc esc** - Command complete.
**A-C-i** - Restyle the current region.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close item list

**Executing, Formatting and Linting Perl Code**

The Perl file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools which are currently used are the ”perl” interpreter, ”perltidy” for formatting and ”perlcritic” for linting. The latter two should be installable either via your default package manager or using the Perl ”cpan” package manager.

**Debugging**

Debugging a perl script can be done inside MicroEmacs by using the **perldb**(3) command.

## NOTES

The hilighting or the item-list matches are typically extended using a file **myperl.emf**.

## BUGS

The flexibility of the perl language does cause some hilighting anomalies from time to time, typically with unbalanced quote characters. Most of the common exceptions have been caught, however there are a few syntax sequences that involve quotation marks that can cause problems.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **indent**(2), **find-tag**(2), **item-list**(3), **perldb**(3), **perltags**(3f), **restyle-buffer**(3), **time**(2m).

Supported File Types

Ref: perl(9) File: m9typ024.9 Date: 2025/08/28

## SYNOPSIS

php - PHP: Hypertext Preprocessor Language File.

## FILES

**hkphp.emf** - PHP: Hypertext Preprocessor file hook definition

## EXTENSIONS

**.php**, **.php3**, **.php4**, **.phphtml** - PHP: Hypertext Preprocessor Language Language File.

## DESCRIPTION

The **php** file type template provides simple hilighting of the Open Source general-purpose scripting language PHP files. PHP is typically used embedded within **html**(9) files.

**Abbreviations**

The installation comes with a default abbreviation file for Php file editing which can be used by writing short abbreviations like "**fo**", **if**", "**el**" or "**fu**" - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**List Items**

Item listing is available and the key constructs of the file may be listed in the *item-list*.

**Tags** A C-tags file may be generated within the editor using the **Tools** -> **PHP-Tools** -> **Create Tag File**. **find-tag**(2) takes the user to the file using the tag information.

**Short Cuts**

The short cut keys used within the buffer are:

**esc h** - Key binding help information.
**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hash (#).
**C-c C-o** - Format a comment
**f6** - List items.
**esc f6** - Close list items window.

## BUGS

None reported.

## SEE ALSO

**abbrev-list**(3), **expand-abbrev-handle**(3), **find-tag**(2), **item-list**(3), **phptags**(3f).
Supported File Types

Ref: php(9) File: m9typ044.9 Date: 2025/08/28

## SYNOPSIS

ps1, psm1 - PowerShell files

## FILES

**hkps1.emf** - PowerShell hook definition
**ps1.eaf** - PowerShell abbreviation file
**ps1.etf** - PowerShell template file.

## EXTENSIONS

**.ps1** - PowerShell file
**.psm1** - PowerShell module file

## DESCRIPTION

The **PowerShell** file type editing provides simple hilighting of PowerShell files, abbreviations, folding, item-list support

### Abbreviations

The installation comes with a default abbreviation file for PowerShell file editing which can be used by writing short abbreviations like "**if**", "**el**" (else) or "**fu**" (function) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Auto Layout

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

### Folding and Information Hiding

Generic folding is enabled within the PowerShell file. The folds occur about functions, classes and enums if these declarations are starting at the beginning of one line and ending at a line starting with a curly brace. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The PowerShell mode offers beside folding as well support for displaying an outline of the functions, classes and enums for easy navigation using the **item-list**(3) command.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (#).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.

**f6** - List items.
**esc f6** - Close item list.

## NOTES

The hilighting and other features are typically extended using a file **myps1.emf**.

## BUGS

None reported so far but many features are added just recently and have not been tested throughly as our main development platform is Linux.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: ps1(9) File: m9typ063.9 Date: 2025/09/11

## SYNOPSIS

puml - Graphviz Dot Diagram Language File.

## FILES

**hkpuml.emf** - PlantUML diagram file hook definition
**puml.eaf** - PlantUML diagram abbreviation file
**puml.etf** - PlantUML diagram template file

## EXTENSIONS

**.puml**, **.pml** - PlantUML file

## DESCRIPTION

The **PlantUML** file macro provides simple hilighting of PlantUML diagram files.

File recognition is performed using the standard file extension **.puml** or **.pml**.

### Editing Environment

Beside of syntax highlighting the file hook offers as well abbreviations, folding for the style section of the file. Further is is possible to expand abbreviations either via short strings using the **expand-abbrev-handle**(3) command and visual inserting abbreviations using the mouse by the **abbrev-list**(3) macro.

### Abbreviations

The installation comes with a default abbreviation file for PlantUML code editing which can be used by writing short abbreviations like "**um**" (start/enduml environment) "**mm**" (mindmap environment) or "**st**" (style section) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the PlantUML files. The folds occur about the **style** sectionss located on the left-hand margin at the beginning of the line and the closing style tag. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**f2** - (un)fold the current region
**f3** - (un)fold all regions

## NOTES

The hilighting is typically extended using a file **mypuml.emf**.

## BUGS

No serious known so far.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **dot**(9), **expand-abbrev-handle**(3), **restyle-buffer**(3), **restyle-region**(3),

Supported File Types

Ref: puml(9) File: m9typ077.9 Date: 2026/01/21

## SYNOPSIS

python, py, **py-doc** - Python Language File.

## FILES

**hkpython.emf** - Python Language file hook definition and py-doc macro implementation
**python.eaf** - Python abbreviation file
**python.etf** - Python template file

## COMMANDS

**py-doc** - Command to display a Python manual page (*esc-x py-doc*)

## EXTENSIONS

**.py** - Python file

## MAGIC STRINGS

**^#![ \t]*/.*env[ \t]+python**

MicroEmacs recognizes the magic string on the first line of the file used to locate the executable. The Python files may be extension-less and are still recognized.

## DESCRIPTION

The **python** file type template provides simple hilighting of Python files, the template provides minimal hilighting.

File recognition is performed using the standard file extension **.py**, or by the magic string.

It offers as well folding and support for displaying an outline of the functions of the Python script for easy navigation using the **item-list**(3) command. Further is is possible to expand abbreviations either via short strings using the **expand-abbrev-handle**(3) command and visual inserting abbreviations using the mouse by the **abbrev-list**(3) macro.

### Folding and Information Hiding

Generic folding is enabled within the Python files. The folds occur about the **def** or **class** keywords located on the left-hand margin at the beginning of the line. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the key constructs of the file may be listed in the *item-list*.

### Abbreviations

The installation comes with a default abbreviation file for Python code editing which can be used by writing short abbreviations like "**main**" for a default Python application outline,"**for**", "**if**" or "**def**" - for a Python function, "**class**" for a Python class - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close list items

**Executing, Formatting and Linting Python Code**

The Python file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools which are currently supported are Python 3 and 2 interpreters, the formatters ruff, yapf and black and linters ruff and pylint. All of these tools can usually be installed using the default package manager of your system or using the pip3 package manager of Python. Here an example on how to install the Python linter and formatter ruff using the pip package manager:

```
pip3 install ruff --user
```

**py-doc macro**

**py-doc** provides a mechanism to display a Python manual page within MicroEmacs. On invoking **py-doc** the user is prompted for the name of the manual page to display:

```
Python help on ?
```

The command invokes **Python**(1) using the **pydoc** module to generate the requested manual page in a `*py-doc*` buffer. This buffer supports **item-list**(3) to help browse the content generated.

## NOTES

The hilightings or the item-list matches are typically extended using a file **mypython.emf**.

## BUGS

There would appear to be too much applied hilighting in this file, it could probably do with rationalizing.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3).

Supported File Types

Ref: python(9) File: m9typ033.9 Date: 2026/02/25

## SYNOPSIS

**r**, **r-doc** - R Language File.

## FILES

**hkr.emf** - R Language file hook and r-doc definition
**r.eaf** - R language abbreviation file
**r.etf** - R language template file

## EXTENSIONS

**.r** - R file

## MAGIC STRINGS

**^#![ \t]*/.*env[ \t]+Rscript**

MicroEmacs recognizes the magic string on the first line of the file used to locate the executable. The Rscript files may be extension-less and are still recognized.

## DESCRIPTION

The **R** macro file hook provides simple hilighting of R files, inserting abbreviations and browsing the structure of the file using item-list and folding..

File recognition is performed using the standard file extension **.r**, **.R** or by the magic string.

### Abbreviations

The installation comes with a default abbreviation file for R code editing which can be used by writing short abbreviations like "**main**" for a default Rscript application outline,"**for**", "**if**" or "**fun**" - for a R function - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the R files. The folds occur about the **function** keywords located on the left-hand margin at the beginning of the line if the smaller-minus assignment operator is used. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the key constructs of the file may be listed in the *item-list*.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close list items

**Executing, Formatting and Linting R Code**

The R file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The R packages which are currently supported are ”formatR” and ”lintr”. They can be installed either using your standard package manager system wide or, for the current user, using the ”install.packages” function. Here an example on how to install these packages from within R:

```
$ R
> install.packages(c("formatR","lintr"))
```

**r-doc macro**

**r-doc** provides a mechanism to display an R manual page within MicroEmacs. On invoking **r-doc** the user is prompted for the name of the manual page to display:

```
R help on ?
```

The command invokes the R **help** function, **Rscript**(1), for the R manual page entered in the message line.

Another R documentation page may be selected in the generated help page by either moving the cursor to the link and pressing return or double clicking on it with the left mouse button. MicroEmacs will then attempt to load and display the selected R manual page.

## NOTES

The hilighting or the item-list matches are typically extended using a file **myr.emf**

## BUGS

Functions with using the equal operator for assignments are not shown in the function outline, this should be seen as a feature which allow you to write hidden internal functions within other functions.

## SEE ALSO

**abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **py**(9), **rdoc**(3f), **sh**(9).

Supported File Types

## SYNOPSIS

Ruby - Ruby programming language

## FILES

**hkruby.emf** - Ruby programming language file hook definition
**ruby.etf** - Ruby programming language template file

## EXTENSIONS

**.rb** - Ruby programming language

## MAGIC STRINGS

**#![ \t]*/.*ruby**
: MicroEmacs recognizes the magic string on any of the first line of the file.

**-[*!]-[ \t]*ruby.*-[*!]-**
: GNU Emacs magic string.

The ruby files may be extension-less and are still recognized.

## DESCRIPTION

The **ruby** file type template provides simple highlighting of Ruby files.

**General Editing**
: On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Highlighting**
: The highlighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Folding and Information Hiding**
: Generic folding is enabled, the folds occur about **module** and **class** definitions located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**List Items**
: Item listing is available and the key constructs of the file may be listed in the *item-list*.

**Indentation**
: Automatic indentation is provided, including a buffer re-style.

**Short Cuts**
: The short cut keys used within the buffer are:

  **esc h** - Key binding help information.
  **C-c C-c** - Comment out the current line.
  **C-c C-d** - Uncomment the current line.
  **C-c C-e** - Comment to the end of the line with hash (#).

**C-c C-o** - Format a comment
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - List items.
**esc f6** - Close list items window.

## BUGS

In line redirection (i.e. `<< EOL`) can cause rendering and indent anomalies.
The block documentation remains under indent control, this should be disabled.

## SEE ALSO

Supported File Types

Ref: ruby(9) File: m9typ042.9 Date: 2025/08/28

## SYNOPSIS

rul - Install Shield Rules

## FILES

**hkrul.emf** - Install Shield hook definition
**rul.etf** - Install Shield template file.

## EXTENSIONS

**.rul** - Install Shield Rules file

## DESCRIPTION

The **rul** file type template provides simple hilighting of Install Shield Rules files.

### Hilighting

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

### Folding and Information Hiding

Generic folding is enabled within the rul file. The folds occur about the keywords **function**...**end** located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**f2** - (un)fold the current region
**f3** - (un)fold all regions

## BUGS

None reported.

## SEE ALSO

**collapse-all**(3), **collapse-current**(3), **indent**(2), **restyle-buffer**(3), **restyle-region**(3).
Supported File Types

Ref: rul(9) File: m9typ011.9 Date: 2022/06/12

## SYNOPSIS

Rust - Rust programming language

## FILES

**hkrust.emf** - Rust programming language hook definition
**rust.eaf** - Rust programming language abbreviation file
**rust.etf** - Rust programming language template file

## EXTENSIONS

**.rs** - Rust

## DESCRIPTION

The **rs** file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

**Folding and Information Hiding**

Generic folding is enabled within the Rust file. The folds occur about functions starting with the `fn`, `type`, `impl` or `mod` keywords and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Rust mode offers beside folding as well support for displaying an outline of the functions and types and modules of the Rust file for easy navigation using the **item-list**(3) command.

**Abbreviations**

The installation comes with a default abbreviation file for Rust file editing which can be used by writing short abbreviations like "**for**", "**if**", "**elif**" or "**fn**" - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of

abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting is typically extended using a file **myrust.emf**

## SEE ALSO

**abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: rust(9) File: m9typ050.9 Date: 2025/08/28

## SYNOPSIS

Samba - Samba smb.conf configuration file

## FILES

**hksamba.emf** - Samba smb.conf configuration file

## EXTENSIONS

**smb.conf** - Samba configuration file

## DESCRIPTION

The **Samba** file type template provides simple highlighting of the Samba `smb.conf` file.

### Hilighting

The hilighting features allow commands, variables, operators and comments of the language to be differentiated and rendered in different colors.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph.

### Folding and Information Hiding

Generic folding is enabled, the folds occur about bracketed constructs <...>...</...> located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the key constructs of the file may be listed in the *item-list*.

### Short Cuts

The short cut keys used within the buffer are:

**esc h** - Key binding help information.
**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hash (#).
**C-c C-o** - Format a comment
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - List items.
**esc f6** - Close list items window.

## SEE ALSO

Supported File Types

Ref: samba(9) File: m9typ039.9 Date: 2022/06/12

## SYNOPSIS

scala - Scala programming language

## FILES

**hkscala.emf** - Scala programming language hook definition
**scala.eaf** - Scala abbreviation file
**scala.etf** - Scala template file

## EXTENSIONS

**.scala** - Scala

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, functions, object,s traits, classes variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Scala file. The folds occur about lines with leading characters and an open curly brace at the end of a line and are continued until the next line with an closing curly brace. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Scala mode offers beside folding as well support for displaying an outline of the functions, objects, classes and traits. These can be explored with easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Scala file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hyphens (–).
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

### NOTES

The hilighting or the addition of new item list matches is typically extended using a file **myscala.emf**.

### SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **time**(2m).

Supported File Types

Ref: scala(9) File: m9typ074.9 Date: 2025/12/21

## SYNOPSIS

scheme - Scheme File.

## FILES

**hkscheme.emf** - Scheme file hook definition

## EXTENSIONS

**.scm**, **.sch** - Scheme file

## DESCRIPTION

The **scheme** file type template provides simple hilighting of Scheme files, the template provides minimal hilighting.

File recognition is performed using the standard file extensions **.scm** or **.sch**.

## NOTES

JASSPA have no idea as to the state of this file hook definition.

## SEE ALSO

Supported File Types

Ref: scheme(9) File: m9typ034.9 Date: 2022/06/12

## SYNOPSIS

*sh - UNIX shell files

## FILES

**hkshell.emf** - UNIX shell file hook definition
**shell.eaf** - UNIX shell abbreviation file
**shell.etf** - UNIX shell programming language template file

## EXTENSIONS

**.sh** - Bourne shell file
**.ksh** - Korn shell file
**.csh** - C-Shell file
**.zsh** - Z-Shell file
**.login** - Shell user login file
**.profile** - Shell user profile
**.tcshrc** - T-Shell start up file

## MAGIC STRINGS

**#![ \t]*/.*sh**

MicroEmacs recognizes the magic string on the first line of the file used to locate the executable. The shell files may be extension less and are still recognized. Note that this is the typical method of identifying shell files and will recognize other files not mentioned above i.e. **bash** shells.

## DESCRIPTION

The **shell** file type template provides simple hilighting of the shell files. It offers as well folding and support for displaying an outline of the functions of the shell script for easy navigation using the **item-list**(3) command. Further is is possible to expand abbreviations either via short strings using the **expand-abbrev-handle**(3) command and visual inserting abbreviations using the mouse by the **abbrev-list**(3) macro.

### Folding and Information Hiding

Generic folding is enabled within the Shell files. The folds occur about the function keyword and the closing braces **function...{...}** located on the left-hand margin at the beginning of the line. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the key constructs of the file may be listed in the *item-list*.

### Abbreviations

The installation comes with a default abbreviation file for shell editing which can be used by writing short abbreviations like "**for**", "**if**" or "**fun**" - for a Bourne shell or Z-shell function - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close list items

### NOTES

The file hook is implemented in the `hkshell.emf` file, the abbreviations are placed within the `shell.eaf` file, the template for the initialization of the new file is defined in the `shell.etf` file. To add your own abbreviations or your own template copy the files to your config folder and edit them within this folder.

### BUGS

Functions defined without using the function keyword can't currenly folded and as well do not appear in the item-list outline.

There is a heavy bias towards Bourne, Korn and Zsh shells. The authors are not csh shell users so have probably missed a lot of csh features.

### SEE ALSO

**fvwm**(9).

Supported File Types

Ref: sh(9) File: m9typ017.9 Date: 2025/08/28

## SYNOPSIS

sql - SQL files

## FILES

**hksql.emf** - SQL hook definition
**sql.etf** - SQL template file.

## EXTENSIONS

**.sql** - SQL file

## DESCRIPTION

The **sql** file type template provides simple hilighting of SQL files, the template provides minimal hilighting.

## BUGS

None reported.

## SEE ALSO

**sql-format**(3), Supported File Types

Ref: sql(9) File: m9typ013.9 Date: 2024/05/21

## SYNOPSIS

swift - Swift programming language

## FILES

**hkswift.emf** - Swift programming language hook definition

## EXTENSIONS

**.swift** - Swift

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, functions, classes variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Swift file. The folds occur about lines with leading characters and an open curly brace at the end of a line and are continued until the next line with an closing curly brace. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Swift mode offers beside folding as well support for displaying an outline of the functions, classes and enums. These can be explored with easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Swift file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hyphens (–).
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **myswift.emf**.

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **time**(2m).

Supported File Types

Ref: swift(9) File: m9typ073.9 Date: 2025/12/21

## SYNOPSIS

tcl, tk - TCL Programming language templates

## FILES

**hktcl.emf** - TCL/TK programming language hook definition
**tcl.eaf** - TCL/TK programming language abbreviation file
**tcl.etf** - TCL/TK programming language template file

## EXTENSIONS

**.tcl**, **.tk** - TCL/TK file

## MAGIC STRINGS

**^#![ \t]*/.*wish**

MicroEmacs recognizes the magic string on the first line of the file used to locate the executable. The tcl files may be extension less and are still recognized.

## DESCRIPTION

The **tcl** provides hilighting and automatic formatting features, in addition to a number of tools to handle the file type.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Abbreviations**

The installation comes with a default abbreviation file for Tcl file editing which can be used by writing short abbreviations like "**if**", "**else**" (else) or "**proc**" (procedure) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Auto Layout**

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

**Tags** A Tcl-tags file may be generated within the editor using the **Tools** -> **Tcl-Tools** -> **Generate Tag File**. **find-tag**(2) takes the user to the file using the tag information.

**Folding and Information Hiding**

Generic folding is enabled within the C and C++ files. The folds occur about braces {...} located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region. Note that folding does not operate on K&R style code.

**Navigation via Item List**

The Tcl mode offers beside folding as well support for displaying an outline of the procedures, methods and classes for easy navigation using the **item-list**(3) command.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hashes (#).
**A-C-i** - Restyle the current region.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - List items
**esc f6** - Close item list

**Executing, Formatting and Linting Tcl Code**

The Tcl file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools, compilers which are currently supported are ”tclsh” the Tcl interpreter and ”tclint” a Python package to lint Tcl code and ”tclfmt” which is a formatter as well provided by the ”tclint” Python package. The formatter and the linter can be installed using the Python package manager ”pip”. Here an example on how to install them for the current user:

```
pip install tclint --user
```

**SEE ALSO**

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **find-tag**(2), **indent**(2), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **tcltags**(3f), **time**(2m).

Supported File Types

Ref: tcl(9) File: m9typ019.9 Date: 2026/05/18

## SYNOPSIS

texinfo - GNU Texinfo documentation file.

## FILES

**hktexi.emf** - Texinfo file hook definition

## EXTENSIONS

**.texi** - Texinfo file

## MAGIC STRINGS

**-*- texinfo -*-**

Recognized by GNU Emacs and MicroEmacs.

## DESCRIPTION

The **texinfo** file type template provides simple hilighting of GNU **Texinfo** files (`.texi`), the template provides minimal hilighting.

File recognition is performed using the standard file extensions, or the magic string.

## NOTES

This template file could benefit from some of the `hklatex.emf` technology for generating the *info* file.

## SEE ALSO

Supported File Types

Ref: texinfo(9) File: m9typ036.9 Date: 2022/06/12

## SYNOPSIS

ts - Typescript programming language

## FILES

**hkts.emf** - Typescript programming language hook definition
**ts.eaf** - Typescript programming language abbreviation file
**ts.etf** - Typescript programming language template file

## EXTENSIONS

**.ts** - Typescript

## DESCRIPTION

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, functions, classes variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The command **indent**(2) performs automatic layout of the text.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the Typescript file. The folds occur about function, class, enums, interface and type definitions and are continued until the next line with an closing curly brace. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The Typescript mode offers beside folding as well support for displaying an outline of the functions, classes, methods, enums, types and interfaces. The functions can be explored with easy navigation using the **item-list**(3)

### Abbreviations

The installation comes with a default abbreviation file for Typescript file editing which can be used by writing short abbreviations like "**fo**" (for), "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Working Environment

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hyphens (–).
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**NOTES**

The hilighting or the addition of new item list matches is typically extended using a file **myts.emf**.

**SEE ALSO**

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **time**(2m).

Supported File Types

Ref: ts(9) File: m9typ072.9 Date: 2025/12/15

## SYNOPSIS

txt, doc - Plain text document file

## FILES

**hkdoc.emf** - Plain text hook definition

## EXTENSIONS

**.txt** - ASCII plain text file
**.doc** - ASCII plain text document file

## DESCRIPTION

The **doc** file type template handles the hilighting and text formating of a plain text file. Within the text document justification and word wrapping are typically enabled. The template allows the user to format text as left, right, center or no justification.

**Auto Layout**

The automatic layout of the text is restricted to justification and wrapping and the detection of bulleted lists. **fill-bullet**(5) may be used to determine the character set used for bullet points, on encountering a bullet the left-hand justification might be modified.

**Formatting rules**

The default mode of operation is automatic mode which attempts to retain the document style whenever a paragraph is re-formatted. This allows rapid entry of text into a reasonable format with no special formating character embedded in the text.

The automatic formatting rules used by **fill-paragraph**(2) in an automatic text mode are defined as follows:

Text on column 0
: Text appearing in the first column is always assumed to be left justified, and non-wrapping, provided that the text does not extend to the buffer fill column. This is typically used for headers and addresses.

Text on right edge
: Text ending at the right edge (the **$buffer-fill-col**(5)), which commences from more that 50% of the page width is assumed to be right justified, non-wrapping. Typically used for addresses.

Text centered
: Text which is centered on the page is assumed to be centered, this is non-wrapping.

Indented
: All other text, not covered by the cases above is assumed to be available for filling. In this case the text is filled by the paragraph and left/right justification is applied, the fill default may be changed with **$buffer-fill-mode**(5) and changed with **buffer-setup**(3).

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-h** - Help information on current mode.
**C-c C-s** - Spell the buffer.
**C-c C-b** - Fill both; perform left and right justification on the margins.
**C-c C-b** - Fill center; center the text on the current line.
**C-c C-l** - Fill left; fill the text on the paragraph (ragged right edge).
**C-c C-r** - Fill right; place text on right margin.
**C-c C-o** - Reduce a paragraph to a single line.
**C-c a** - Move to automatic formatting mode (default).
**C-c l** - Change mode to left formating
**C-c r** - Change mode to right formating
**C-c r** - Change mode to both formating
**C-c c** - Change mode to center formating
**C-c n** - Change mode to no formating

## NOTES

To move text to a word processor then it is advised that all paragraphs are reduced to single lines, leading white space should be deleted (any possibly blank lines) and then import to the word processor. This saves considerable time as the word processor styles may be applied without handling spaces and band end of line characters.

## MAGIC STRINGS

**-!- document -!-**

MicroEmacs specific tag, recognizes the file as a plain text document. No hilighting of the document is performed.

**-!- document; sectioned -!-**

MicroEmacs specific tag, recognizes the file as a document that contains sections. A crude section hilighting is enabled as follows:

Lines commencing with > are assumed to be comments, typically used at the head of the document .

```
> -!- document; sectioned -!-
>
> Author:       My Self
> Created:      11/11/97
> Modified:     <211197.1003>
> Location:     /xx/yy/zz.doc
```

All lines commencing with start (*) are assumed to be bullet lists. Bullet is hilighted.

All lines commencing with `[a-zA-Z])` or `[0-9])` are assumed to be minor sections. The section number is hilighted. e.g.:

```
a) text
1) text
```

All text in single or double quotes is hilighted, assumed to be literal text. and are hilighted i.e. This is a **"double quote"** or **'a'** single quote.

Lines commencing with underscore (_) are hilighted to the end. typically used as demarcation breaks or for section underlining

**-!- Document; pseudo-code -!-**

The document contains pseudo code, and the pseudo code is hilighted. The pseudo-code tokens are defined as follows:

**//** introduces a comment to the end of the line.

Command words comprise:

`BEGIN`, `BREAK`, `CASE`, `CLEAR`, `CONTINUE`, `DO`, `DONE`, `ELIF`, `ELSE`, `END`, `ENDIF`, `FOR`, `FUNCTION`, `GOTO`, `IF`, `ONEVENT`, `ONINTERRUPT`, `PROCEDURE`, `REPEAT`, `RETURN`, `SET`, `SWITCH`, `THEN`, `TO`, `UNTIL`, `WHILE`,

Pseudo logical operators include

`AND`, `FALSE`, `MOD`, `NOT`, `OR`, `TRUE`, `XOR`,

**-!- document; sectioned; pseudo-code -!-**

A combination of both of the above.

## BUGS

The automatic mode sometimes mistakes an indented paragraph for a centered paragraph. This only typically occurs when the first line of the paragraph is not filled to the right. When the formatting error occurs, simply pad the line out so that it extends past the fill column and re-apply the formatting.

Unfortunately there is nothing that can be done to alleviate this problem, but it occurs infrequently.

## SEE ALSO

**$buffer-fill-col**(5), **$buffer-fill-mode**(5), **$fill-col**(5), **$fill-mode**(5), **buffer-setup**(3), **fill-paragraph**(2), **spell-buffer**(3).

Supported File Types

Ref: txt(9) File: m9typ003.9 Date: 2022/06/12

## SYNOPSIS

typ, typst - Typst Markup File.

## FILES

**hktypst.emf** - Typst file hook definition
**typst.eaf** - Typst abbreviation file
**typst.etf** - Typst template file

## EXTENSIONS

**.typ**, **.typst**, **.ttyp**, **.ttypst** - Typst files

## DESCRIPTION

The **Typst** file hook provides a editing environment for Typst files. The file hook provides highlighting, folding of section, document outline using item-list and abbreviations. Fenced code blocks within triple backticks can be highlighted using the declared code block language.

File recognition is performed using the standard file extension **.typ**, **.typst** or the extensions **.ttyp** and **.ttypst**. The latter two file extensions indicate there are non-Typst based extensions for instance to embed code of other programming languages like R or Tcl within the file which needs to be processed with tools like *tmdoc* before the document can be compiled with the *Typst* compiler.

### Abbreviations

The installation comes with a default abbreviation file for Typst code editing which can be used by writing short abbreviations like "**fig**" for a figure environment, "**bib**" for a bibliography file, "**img**" for an image file, or "**co**" for a code chunk - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

### Folding and Information Hiding

Generic folding is enabled within the Typst files. The folds occur about the level 1 and 2 headers, so headers with one or two equal symbols at the beginning of a line. The following commands are supported to show and hide the function bodies. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### List Items

Item listing is available and the level 1 and 2 headers of the file are displayed in the *item-list*.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**f2** - (un)fold the current region
**f3** - (un)fold all regions
**f6** - list items
**esc f6** - close list items

### Code Chunk Highlighting

Inside code chunks as well syntax hilghlighting for a few programming languages like Tcl (tcl), R (r), GraphViz Dot (dot), Python (py), SQL (sql), MicroEmacs macro code (emf)or Javascript (js) is available. Here an example which should be highlighted within a Typst file using the R file hook:

```
```r
x = 1
test = function () {
    print("Test")
}
```
```

## SEE ALSO

**abbrev-list**(3), **collapse-all**(3), **collapse-current**(3), **expand-abbrev-handle**(3), **item-list**(3), **adoc**(9), **md**(9), **wiki**(9)

Supported File Types

Ref: typst(9) File: m9typ065.9 Date: 2025/10/20

## SYNOPSIS

Vala - Vala programming language

## FILES

**hkvala.emf** - Vala programming language hook definition
**vala.eaf** - Vala programming language abbreviation file
**vala.etf** - Vala programming language template file

## EXTENSIONS

**.vala** - Vala

## DESCRIPTION

The **vala** file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

**Folding and Information Hiding**

Generic folding is enabled within the Vala file. The folds occur about functions and classes and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Vala mode offers beside folding as well support for displaying an outline of the functions and classes of the Vala file for easy navigation using the **item-list**(3)

**Abbreviations**

The installation comes with a default abbreviation file for Vala file editing which can be used by writing short abbreviations like "**for**", "**if**", "**ei**" (else if) or "**cl**" (class) - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **my-vala.emf**

## SEE ALSO

**abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

Ref: vala(9) File: m9typ064.9 Date: 2025/09/15

## SYNOPSIS

bas, cls - Visual Basic file

## FILES

**hkvb.emf** - Visual Basic macro file.

## EXTENSIONS

**.bas**, **.cls**

## MAGIC STRINGS

**-!- msvb -!-**

## DESCRIPTION

The **Visual Basic** template performs the hilighting of Visual Basic files. The file type is recognised by the standard extension, or by the inclusion of the magic string.

**Hilighting**
: The hilighting features allows components of the language to be differentiated and rendered in different colors.

**Auto Layout**
: The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively. The default indentation setting is 2.

**Folding and Information Hiding**
: Generic folding is enabled within VisualBasic files. The folds occur about *Function* and *Sub*routine definitions located on the left-hand margin and their *End* mark which is located at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**
: The VisualBasic mode offers beside folding as well support for displaying an outline of the Functions and Subroutines for easy navigation using the **item-list**(3) command.

**Tags**
: A C-tags file may be generated within the editor using the **Tools** -> **VB-Tools** -> **Generate Tag File**. **find-tag**(2) takes the user to the file using the tag information.

**Short Cuts**
: The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c esc esc** - Command complete.
**A-C-i** - Restyle the current region.
**f2** - (Un)fold the current region.
**f3** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list.

## SEE ALSO

**collapse-all**(3), **collapse-current**(3), **find-tag**(2), **item-list**(3), **restyle-region**(3), **vbtags**(3f), Supported File Types

Ref: vb(9) File: m9typ031.9 Date: 2025/07/23

## SYNOPSIS

Verilog - Verilog hardware description language

## FILES

**hkverilg.emf** - Verilog hardware description language hook definition

## EXTENSIONS

**.v** - Verilog file

## DESCRIPTION

The **Verilog** file type template provides simple highlighting of the Verilog hardware description language. **.v** files are automatically recognized as Verilog files. For the documentation of the V programming language look at **vlang**(9).

### Hilighting

The hilighting features allow commands, variables and operators of the language to be differentiated and rendered in different colors.

### Comments

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph.

### Short Cuts

The short cut keys used within the buffer are:

**esc h** - Key binding help information.
**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hash (#).
**C-c C-o** - Format a comment

## SEE ALSO

**vlang**(9).
Supported File Types

Ref: verilog(9) File: m9typ040.9 Date: 2025/07/20

## SYNOPSIS

vhdl - VHDL hardware simulation files

## FILES

**hkvhdl.emf** - VHDL hook definition
**vhdl.eaf** - VHDL abbreviation file.
**vhdl.etf** - VHDL template file.

## EXTENSIONS

**.vhdl**, **.vhd** - VHDL file

## DESCRIPTION

The **vhdl** file type template provides simple hilighting of VHDL files, and a few abbreviations like **if**, **be** (begin-end) or **wh** (while) and the template provides minimal hilighting.

## BUGS

None reported. Template could probably benifit from some form of auto indentation.

## SEE ALSO

Supported File Types

Ref: vhdl(9) File: m9typ014.9 Date: 2025/07/13

## SYNOPSIS

vlang - V programming language

## FILES

**hkvlang.emf** - V programming language hook definition
**vlang.eaf** - V programming language abbreviation file
**vlang.etf** - V programming language template file

## EXTENSIONS

**.v** - V-lang

## DESCRIPTION

The **v**lang file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

### General Editing

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

### Hilighting

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

### Folding and Information Hiding

Generic folding is enabled within the V file. The folds occur about functions defined with the **fn** keyword and a closing curly brace both at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

### Navigation via Item List

The V mode offers beside folding as well support for displaying an outline of the functions and structures of the V file for easy navigation using the **item-list**(3) command. Public functions are highlighted with color in the item list.

### Abbreviations

The installation comes with a default abbreviation file for V language file editing which can be used by writing short abbreviations like "**for**", "**if**", "**elif**" (else if), "**str**" (public structure) or "**fn**" (function) - and then pressing the key-binding for the

**expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

**Executing, Formatting and Linting V Code**

The V file hook supports the **file-exec**(3), the **file-format**(3) and the **file-lint**(3) macros. The tools, compilers which are currently supported are just the V sub-commands usable for this task so ”v run”, ”v fmt” and ”v vet”.

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **myvlang.emf**. As this language shares the same extension (`.v`) as Verilog files there can be a conflict between these two major-modes. MicroEmacs attempts to detect which major mode to use based on the content of the file, but this can go wrong. The best solution to this problem is to add a file content identifier, i.e. add a line like the following to one of the first 2 lines of the file:

```
// -*- vlang -*-
```

Alternatively, if there are no verilog files and you do not wish to edit them, add the following function to your **myvlang.emf** file:

```
define-macro ftest-vlang
  set-variable .result 100
!emacro
```

This will ensure all `.v` files, without a content identifier, will have vlang major-mode applied.

## SEE ALSO

**major-mode**(2), **abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **file-exec**(3), **file-format**(3), **file-lint**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).

Supported File Types

## SYNOPSIS

vrml - VRML File

## FILES

**hkvrml.emf** - VRML File hook definition

## EXTENSIONS

<*none*> - Uses the *Magic String* only.

## MAGIC STRINGS

**#VRML**

A generic tag that appears on the first line at the top of a **VRML** (or **wrl**) file. MicroEmacs automatically recognises the tag and adopts the appropriate mode.

## DESCRIPTION

The **vrml** file type template handles the hilighting of **VRML** files.

**Hilighting**

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

**Short Cuts**

The short cut keys used within the buffer are:

**A-C-tab** - Restyle a region.

## BUGS

No bugs reported

## SEE ALSO

**indent**(2), **restyle-buffer**(3), **restyle-region**(3).

Supported File Types

Ref: vrml(9) File: m9typ023.9 Date: 2022/06/12

## SYNOPSIS

Wiki - GNU Wiki File.

## FILES

**hkwiki.emf** - GNU Wiki file hook definition

## MAGIC STRINGS

**-[*!]-[ \t]*mode: wiki.*-[*!]-**

## DESCRIPTION

The **Wiki** file type template provides a editing environment for GNU Wiki files. The template provide highlighting and an Edit and View mode to allow the Wiki contents to be viewed.

File recognition is performed using the GNU Emacs magic string.

## SEE ALSO

Supported File Types

Ref: wiki(9) File: m9typ037.9 Date: 2022/06/12

## SYNOPSIS

x86 - Intel .x86 Assembler File

## FILES

**hkasmx86.emf** - Intel .x86 Assembler hook definition
**asmx86.etf** - Intel .x86 Assembler template file.

## EXTENSIONS

**.x86** - Intel .x86 Assembler File

## MAGIC STRINGS

**-!- asmx86 -!-**

Recognized by MicroEmacs only, defines the file to be a Intel x86 assembler file.

## DESCRIPTION

The **x86** file type template provides simple hilighting of Intel x86 assembler files.

### Hilighting

The hilighting features allow commands, variables, logical, comments, strings and characters of the language to be differentiated and rendered in different colors.

### Auto Layout

The indentation mechanism is enabled which performs automatic layout of the text. **restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

### Short Cuts

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.

## BUGS

None reported.

## SEE ALSO

**indent**(2), **restyle-region**(3) **restyle-buffer**(3) **asm**(9)
Supported File Types

Ref: x86(9) File: m9typ029.9 Date: 2022/06/12

## SYNOPSIS

xml - Extensible Markup Language File.

## FILES

**hkxml.emf** - Extensible Markup Language file hook definition

## EXTENSIONS

**.xml** - Extensible Markup Language File.
**.xsl** - Extensible Stylesheet Language file.
**.xul** - XML User Interface Language file.

## MAGIC STRINGS

**<?xml>**

MicroEmacs recognizes the magic string on any of the first line of the file. The XML files may be extension-less and are still recognized.

## DESCRIPTION

The **xml** file type template provides simple hilighting of XML files.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Embedded Languages**

Support is provided for embedded code fragments in Java and Visual Basic.

**Folding and Information Hiding**

Generic folding is enabled, the folds occur about bracketed constructs **<...>**...**</...>** located on the left-hand margin. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Buffer Commands**

**xml-forward-tag** and **xml-backward-tag** are provided to advance to the next < . . > tag in the buffer.

**xml-goto-matching-tag** finds the tag that matches the current tag.

**xml-verify** verifies the syntax of the XML reporting tags that are not matched in addition to syntax errors in the tag attributes.

**Spelling**

Buffer spell checking is XML aware.

**Completion**

Completion is enabled, </**esc** **esc** will expand to the closing tag for the last tag that remains open.

### Short Cuts

The short cut keys used within the buffer are:

**esc h** - Key binding help information.
**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with hash (#).
**C-c C-o** - Format a comment

### BUGS

None reported.

### SEE ALSO

Supported File Types

Ref: xml(9) File: m9typ041.9 Date: 2025/08/28

## SYNOPSIS

Zig - Zig programming language

## FILES

**hkzig.emf** - Zig programming language hook definition
**zig.eaf** - Zig programming language abbreviation file
**zig.etf** - Zig programming language template file

## EXTENSIONS

**.zig** - Zig

## DESCRIPTION

The **zig** file type templates share much with the **c**(9) template definitions, utilising the electric 'C' features for automatic layout of text.

**General Editing**

On creating a new file, a new header is automatically included into the file. **time**(2m) is by default enabled, allowing the modification time-stamp to be maintained in the header.

**Hilighting**

The hilighting features allow commands, variables, logical, preprocessor definitions, comments, strings and characters of the language to be differentiated and rendered in different colors.

**Auto Layout**

The C-Mode **indent**(2) performs automatic layout of the text.

**restyle-region**(3) and **restyle-buffer**(3) are available to reformat (re-layout) selected sections of the buffer, or the whole buffer, respectively.

Comments may be formatted using `esc o`, which reformats the comments according to the current fill paragraph. If a comment commences with `/***...` then the comment is automatically formatted to a box.

**Folding and Information Hiding**

Generic folding is enabled within the Zig file. The folds occur about functions and test code and a closing curly brace at the beginning of a line. **collapse-all**(3) (un)folds all regions in the file, **collapse-current**(3) (un)folds the current region.

**Navigation via Item List**

The Zig mode offers beside folding as well support for displaying an outline of the functions, enums, structs, unions and test code of the Zig file for easy navigation using the **item-list**(3) command.

**Abbreviations**

The installation comes with a default abbreviation file for Zig file editing which can be used by writing short abbreviations like "**for**", "**if**", "**elif**", "**fn**" or "**struct**" - and then pressing the key-binding for the **expand-abbrev-handle**(3) command. The list of abbreviations can be used interactively in the GUI version of MicroEmacs using the **abbrev-list**(3) command.

**Working Environment**

**compile**(3) may be invoked to rebuild the source, the user is prompted to save any files.

**Short Cuts**

The short cut keys used within the buffer are:

**C-c C-c** - Comment out the current line.
**C-c C-d** - Uncomment the current line.
**C-c C-e** - Comment to the end of the line with stars (*).
**A-C-i** - Restyle the current region.
**esc q** - Format a comment.
**esc o** - Format a comment.
**f2** - (Un)fold the current region.
**C-f2** - (Un)fold all regions.
**f6** - List items.
**esc f6** - Close item list,

## NOTES

The hilighting or the addition of new item list matches is typically extended using a file **myzig.emf**

## SEE ALSO

**abbrev-list**(3), **c**(9), **collapse-all**(3), **collapse-current**(3), **compile**(3), **expand-abbrev-handle**(3), **item-list**(3), **restyle-buffer**(3), **restyle-region**(3), **time**(2m).
Supported File Types

Ref: zig(9) File: m9typ052.9 Date: 2025/08/28

## INSTALLATION

This page describes introductory notes for the installation and setup of MicroEmacs 2026.

### Quick Install

The quickest way to install MicroEmacs for testing without reading the rest of this document is to:

- Create a new directory i.e. `jasspa`.
- Unpack the macros archive into this directory.
- Unpack the executable into this directory.
- Run `me` from this directory.

On starting, you will be prompted for a default directory, accept the defaults. Use the mouse and configure the user from the menu bar:

```
Tools -> User Setup
```

This allows the user and screen settings to be altered. On becoming more accustomed to the editor then a fuller installation may be performed.

### Getting Help

See Contact Information for full contact information. A mail archive exists at:

```
http://groups.yahoo.com/group/jasspa/
```

If you wish to participate in the list then you must first register by sending an empty mail message body to:

```
jasspa-subscribe@yahoogroups.com
```

You will then be able to mail any questions into the group. Registration is required in order to prevent *spam* mailings from entering into the lists.

### Distribution

MicroEmacs is distributed in the following files:

### Complete Installations

The Microsoft 95/98/NT/2K/XP, Solaris, HP-UX, Linux and the more popular platforms may be installed using a package and do not require the components specified in later sections as these are all bundled together (these packages do not include spelling).

`jasspa-me-YYYYMMDD-1.i386.rpm` - Linux RedHat 9.x package.
`jasspa-me-YYYYMMDD-1.src.rpm` - Linux RedHat 9.x source package.
`jasspa-mepkg-hpux-pa-10.20-YYYYMMDD.depot.gz` - HP-UX 10 package.
`jasspa-mepkg-hpux-pa-11.00-YYYYMMDD.depot.gz` - HP-UX 11 package.
`jasspa-mepkg-ms-win32-2004030.zip` - 95/98/NT/2K/XP Install Shield
`jasspa-mepkg-sun-i386-58-YYYYMMDD.zip` - Sun 2.8(i386) package.

`jasspa-mepkg-sun-i386-59-YYYYMMDD.zip` - Sun 2.9(i386) package.
`jasspa-mepkg-sun-sparc-56-YYYYMMDD.zip` - Sun 2.6(Sparc) package.
`jasspa-mepkg-sun-sparc-58-YYYYMMDD.zip` - Sun 2.8(Sparc) package.
`jasspa-mepkg-sun-sparc-59-YYYYMMDD.zip` - Sun 2.9(Sparc) package.

**Executable Source Code**

The source code release for MicroEmacs contains makefiles (`*.mak`) for all supported platforms. Microsoft '95/'98/NT/2K/XP makefiles contain options at the top of the makefile to enable/disable console and URL support.

`jasspa-mesrc-YYYYMMDD.zip` - Source code for all platforms
`jasspa-mesrc-YYYYMMDD.tar.gz` - Source code

**Executable Images**

`jasspa-me-cygwin-YYYYMMDD.exe.gz` - Cygwin Executable (X-Windows)
`jasspa-me-hpux-pa-10.20-YYYYMMDD.gz` - Hewlett Packard HP-UX 10 Executable
`jasspa-me-hpux-pa-11.00-YYYYMMDD.gz` - Hewlett Packard HP-UX 11 Executable
`jasspa-me-ibm-aix-4-YYYYMMDD.gz` - IBM's AIX 4.3 Executable
`jasspa-me-linux-i386-2.4-YYYYMMDD.gz` - Linux 2.2.x/2.4.x Executable
`jasspa-me-linux-i386-2.6-YYYYMMDD.gz` - Linux 2.6.x Executable
`jasspa-me-ms-win32-YYYYMMDD.zip` - Windows 32' (95/98/NT/2K/XP) Executable
`jasspa-me-ms-win32s-YYYYMMDD.zip` - Windows win32s (Win3.1/3.11) Executable
`jasspa-me-msdos-djgpp1-YYYYMMDD.zip` - DOS Executable (DJGPP v1.x)
`jasspa-me-msdos-djgpp2-YYYYMMDD.zip` - DOS Executable (DJGPP v2.x)
`jasspa-me-openbsd-i386-3.4-YYYYMMDD.gz` - Open BSD 3.4 Executable
`jasspa-me-sgi-irix-6-YYYYMMDD.gz` - Silicon Graphics Irix 6 Executable
`jasspa-me-sun-i386-58-YYYYMMDD.gz` - Sun Solaris 2.8 Intel Platform Executable
`jasspa-me-sun-i386-59-YYYYMMDD.gz` - Sun Solaris 2.9 Intel Platform Executable
`jasspa-me-sun-sparc-56-YYYYMMDD.gz` - Sun OS 5.5 Executable
`jasspa-me-sun-sparc-58-YYYYMMDD.gz` - Sun OS 5.8 Executable
`jasspa-me-sun-sparc-59-YYYYMMDD.gz` - Sun OS 5.9 Executable
`jasspa-mec-apple-darwin-6.6-YYYYMMDD.gz` - Apple console Executable
`jasspa-mec-cygwin-YYYYMMDD.exe.gz` - Cygwin Executable (Console)

**Help File Images** (all platforms)

`jasspa-mewinhlp-YYYYMMDD.zip` - Windows Help file
`jasspa-mehtm-YYYYMMDD.zip` - HTML Help files for 8.3 file systems (.htm)
`jasspa-mehtml-YYYYMMDD.tar.gz` - HTML Help files (.html)

**Macro File Images** (all platforms)

`jasspa-memacros-YYYYMMDD.tar.gz` - Macro files
`jasspa-memacros-YYYYMMDD.zip` - Macro files

jasspa-metree-YYYYMMDD.tar.gz - Complete install tree + help
jasspa-metree-YYYYMMDD.zip - Complete install tree + help

**Spelling Dictionaries** (all platforms)

One of the following base dictionaries is required for spelling. The extended dictionaries require the base dictionary and are recommended for a more comprehensive spelling list. Other languages are supported.

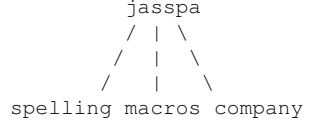
ls_dede.zip - German rules and base + extended dictionary.
ls_engb.zip - British rules and base + extended dictionary.
ls_enus.zip - American rules and base + extended dictionary.
ls_eses.zip - Spanish rules and dictionary.
ls_frfr.zip - French rules and dictionary.
ls_itit.zip - Italian rules and dictionary
ls_plpl.zip - Polish rules and dictionary.
ls_ptpt.zip - Portuguese rules and dictionary.
ls_ruye.zip - Russian YE rules and dictionary.
ls_ruyo.zip - Russian YO rules and dictionary.

ls_dede.tar.gz - German rules and base + extended dictionary.
ls_engb.tar.gz - British rules and base + extended dictionary.
ls_enus.tar.gz - American rules and base + extended dictionary.
ls_eses.tar.gz - Spanish rules and dictionary.
ls_frfr.tar.gz - French rules and dictionary.
ls_itit.tar.gz - Italian rules and dictionary
ls_plpl.tar.gz - Polish rules and dictionary.
ls_ptpt.tar.gz - Portuguese rules and dictionary.
ls_ruye.tar.gz - Russian YE rules and dictionary.
ls_ruyo.tar.gz - Russian YO rules and dictionary.

**NOTE:** The binary versions of the executables held on the site include the platform name as part of the executable name i.e. **me** for DOS is called **medos.exe**. On installing the binaries onto the target machine, you should rename the executable to **me** or **me.exe**, whatever is appropriate. The ONLY exception to this rule is the Microsoft Windows executable where **mewin32.exe** should be renamed to **me32.exe**. Our reason for this naming is to allow the executables to be unpacked in the same directory and not be confused with each other.

**Installation Directories**

The suggested installation structure for all platforms is defined as:




This directory structure is suggested to facilitate upgrades in the future and to keep user data separate from the system macros. MicroEmacs understands this directory structure.

The directory structures are defined as:

**macros** The standard macro distribution, as supplied by JASSPA.

**spelling** Spelling dictionaries.

**company**
Company wide local definitions for the local site. This directory may be omitted if not applicable. Header templates (`.etf` are typically copied from the `macro` directory to the `company` directory and modified for the local site.

The search order is defined as `root`, `company`, `macros`, `spelling`.

### UNIX Installation (System Wide)

A system wide installation on UNIX for multiple users adheres to conventional package installation methods.

**Distribution Components Required:**
`jasspa-me-<`*unix*`>-YYYYMMDD.gz`
`jasspa-metree-YYYYMMDD.tar.gz`
<*spelling*>`.gz`
`jasspa-mehtml-YYYYMMDD.tar.gz` (optional)
**Installation:**
It is recommended that the macro files are placed in a standard shared system directory. The standard directory locations searched by MicroEmacs are defined on a per-platform basis, only one of the directory locates is selected, as follows:

| | HP-UX | | SunOS | | AIX | | Cygwin | | |
|---|---|---|---|---|---|---|---|---|---|
| | IRIX | | UnixWare | | ***BSD | | Linux | | NeXT |
| /opt/jasspa: | y | y | y | y | y | y | n | n | n |
| /usr/share/jasspa: | y | y | y | y | y | y | y | n | n |
| /usr/local/jasspa: | y | y | y | y | y | y | y | y | y |
| /usr/local/share/jasspa: | n | n | n | n | y | n | n | n | n |

- Create the directory `/usr/local/jasspa`.
- Unpack and install the `jasspa-metree-YYYYMMDD.tar.gz` into this directory.
- If spelling dictionaries are required then place them in the *spelling* sub-directory created when *metree* was unpacked. Unpack and install the `spelling` into this directory.
- If the standard icons are required for Gnome, KDE or some other window manager then create the sub-directory `/usr/local/jasspa/icons` and unpack and install the icons into the new directory.
- Unpack (or build) the executable and place in "`/usr/local/bin`"

The executable may now be run by all users. If the macros are placed in a non-standard directory location then the environment variable **MEINSTALLPATH**(5) to define the root jasspa tree.

For **csh**(1) users execute a "`rehash`" command and then **me**(1) can be executed from the command line.

By default a X-Windows terminal is displayed, ensure that `$DISPLAY` and `$TERM` are correctly configured. To execute a terminal emulation then execute **me** with the `-n` option i.e. "`me -n`". Note that this is not required if you are using a `vt100` emulation.

When MicroEmacs is executed the local directory `/.config/jasspa` is created for the user settings. The user home directory may be changed by setting the environment variable **$MEPATH**(5).

### UNIX Installation (Single User)

Where the user is not permitted to install 3rd party applications in the system directories then MicroEmacs may be installed locally as follows:

- Create the directory `/.config/jasspa`.
- Unpack and install the `jasspa-metree-YYYYMMDD.tar.gz` into this directory.
- If spelling dictionaries are required then place them in the *spelling* sub-directory created when *metree* was unpacked. Unpack and install the `spelling` into this directory.
- If the standard icons are required for Gnome, KDE or some other window manager then create the sub-directory `/.config/jasspa/icons` and unpack and install the icons into the new directory.
- The user files will be placed in the `/.config/jasspa` directory by default. To keep the user macros separate then the directory `/.config/jasspa/$LOGNAME` may be created and will identified as the users home directory.
- Unpack (or build) the executable and place in " `/bin`"

MicroEmacs may be executed after it has been re-hashed (depending on the shell).

### DOS Installation

MS-DOS has no concept of a standard installation directory. The location of the MicroEmacs macros is generally determined from the directory location of the executable.

**Executable:**
Compiled with DJGPP V1.0

Distribution components required:

`jasspa-me-msdos-djgpp1-YYYYMMDD.zip`
`jasspa-metree-YYYYMMDD.zip`
<*spelling*>`.zip`

`mewinhlp.zip` if you are using windows 3.1/3.11

Recommended installed components:

`grep` - Version of grep (djgpp recommended)
`make` - Version of make (djgpp recommended)
`diff` - Version of diff (djgpp recommended)

**Installation:**
This installation relies on the executable position.

- Create the directory `c:\jasspa` (or other location)
- Unpack and install the `jasspa-metree-YYYYMMDD.zip` into this directory.
- If spelling dictionaries are required then place them in the *spelling* sub-directory created when *metree* was unpacked. Unpack and install the `spelling` into this directory.
- Unzip the executable and place it in the jasspa root directory `c:\jasspa`.

- Edit "`c:\autoexec.bat`" and add the following lines:

```
SET PATH=%PATH%;c:\jasspa
```

- Reboot the system.
- MicroEmacs may be run from the command line using **me**.

**Graphics Cards:**
MicroEmacs may be configured to the text modes of your graphics card. Refer to you graphics card DOS text modes to identify the text modes supported by your monitor. The text mode number may be entered into the user monitor configuration, defined in **Tools** -> **User Setup**.

Running DOS Version From Windows (3.x)

The MS-DOS version of MicroEmacs may be executed from a **.pif** file. Use the pif editor to create a new **.pif** file to launch MicroEmacs. The size of the DOS window may be configured from the command line, set the terminal size using one of the following command lines:

```
me -c -v$TERM=E80x50        - 80 x 50 window
me -c -v$TERM=E80x25        - 80 x 25 window.
```

We usually add the `-c` option so that MicroEmacs is executed with history information. This may be omitted if required.

**Microsoft Windows 3.1/3.11**

**Executable:**
Compiled with Microsoft Developer 2.0

**Helper DLL:**
Under **Win32s** a helper DLL **methnk16.dll** is required to perform the **pipe-shell-command**(2) in a synchronous manner. This should be installed into the `C:\WINDOWS\SHELL` directory. This (rather inelegantly) gets around the problems of spawning a process under **win32s** due to a number of Microsoft bugs in the operating system. Note: that on a spawn operation a `MS-DOS` window is visible, this is due to the nature of the command shell on this platform which has a tendency to prompt the user at every opportunity, hence a certain amount of interaction (which is out of our control) is necessary.

The helper DLL is compiled with a 16-bit Windows compiler - MSVC 1.5.

**Distribution components required:**

`jasspa-me-ms-win32s-YYYYMMDD.zip`
`jasspa-metree-YYYYMMDD.zip`
`jasspa-mewinhlp-YYYYMMDD.zip`
<*spelling*>`.zip`

**Recommended installed components:**

`4dos` - command shell (giving *stderr* redirection)
`grep` - Version of grep (GNU port of grep recommended)
`diff` - Version of diff (GNU port of grep recommended)
`make` - use nmake or GNU port of make.

If**win32s** is a requirement on this platform, typically taken from **pw1118.exe** which freely available on the Internet.

**Installation:**
This version of Windows does not have a *install* directory as '95/'98 and it is expected that the `MS-DOS` version will coexist. No *Install Shield* installation is provided. Install in a directory structure similar to `MS-DOS`. Install the helper DLL **methnk16.dll** in the `C:\WINDOWS\SHELL` directory.

If the macros are placed in a non-standard directory location then the environment variable **MEINSTALLPATH**(5) to define the root jasspa tree.

**Support Status:**
The **win32s** release has not been used with vengeance, although no specific problems have been reported with this release.

**Windows 95/98/NT/2K/XP**

**Executable:**

Compiled with Microsoft Developer 6.x

**Install Shield**

An **Install Shield** version of MicroEmacs is available which includes all of the distribution components.

**Distribution components required:**

`jasspa-mepkg-ms-win32-2004030.zip`

**Recommended installed components:**

`grep` - Version of grep (GNU port of grep recommended)
`diff` - Version of diff (GNU port of grep recommended)
`make` - use nmake or GNU port of make.

A good source of UNIX native tools is

`http://unxutils.sourceforge.net/` - for native windows grep, diff, zip etc.
`http://www.cygwin.com` - for a shell environment.

**Installation:**

- Create the directory "`c:\Program Files\Jasspa\MicroEmacs`" (or other location)
- Create the sub-directory `c:\Program Files\Jasspa\MicroEmacs\macros`. Unpack and install the `memacros.zip` into this directory
- If spelling dictionaries are required then create the directory `c:\Program Files\Jasspa\MicroEm` Unpack and install the `spelling` into this directory.
- Unzip the executable components into "`C:\Program Files\Jasspa\MicroEmacs`"

- Create a short cut to MicroEmacs for the Desktop

  Right click on the desk top

  ```
  => New
  => Short
  => Command Line:  "c:\Program Files\Jasspa\MicroEmacs\me.exe
  -c"
  => Short Cut Name:  "MicroEmacs"
  ```

  MicroEmacs may be executed from the shortcut.

**Open Actions** Microsoft Windows 95/98/NT provide short cut actions, assigning an open action to a file. The short cuts may be installed from the **Install Shield** installation, but may alternatively be explicitly defined by editing the registry file with **regedit**(1).
A file open action in the registry is bound to the file file extension, to bind a file extension *.foo* to the editor then the following registry entries should be defined:

```
[HKEY_CLASSES_ROOT\.foo]
"MicroEmacs_foo"
[HKEY_CLASSES_ROOT\MicroEmacs_foo\DefaultIcon]
"C:\Program File\JASSPA\MicroEmacs\meicons,23"
[HKEY_CLASSES_ROOT\MicroEmacs_foo\Shell\open]
"&Open"
[HKEY_CLASSES_ROOT\MicroEmacs_foo\Shell\open\command]
"C:\Program File\JASSPA\MicroEmacs\me32.exe -o "%1""
```

In the previous example the *DefaultIcon* entry is the icon assigned to the file. This may be an icon taken from `meicons.exe` (in this case icon number 23), or may be some other icon. The open action in the example uses the **-o** option of the *client-server*, which loads the file into the current MicroEmacs session, alternatively the **-c** option may be used to retain the previous context, or no option if a new session with no other files loaded is started.
A generic open for ALL files may be defined using a wild-card, this may be used to place a *MicroEmacs* edit entry in the right-click pop-up menu, as follows:

```
[HKEY_CLASSES_ROOT\*\shell]
[HKEY_CLASSES_ROOT\*\shell\MicroEmacs]
"&MicroEmacs"
[HKEY_CLASSES_ROOT\*\shell\MicroEmacs\command]
"C:\Program File\JASSPA\MicroEmacs\me32.exe -o "%1""
```

## ORGANIZING A LOCAL USER PROFILE

MicroEmacs uses local user configuration profiles to store user specific information. The user information may be stored in the MicroEmacs directory, or more typically in a users private directory. On login systems then the user name and user private directory location is automatically determined from the system.

The user name identified by MicroEmacs is **$user-name**(5) this value may be set or re-defined using the environment variable **$MENAME**(5). On MS-DOS and early windows systems where there is no user login then then **$MENAME**(5) may be defined in the environment.

The location of the user information depends upon your platform and installation configuration. The private directory identified by MicroEmacs is **$user-path**(5) this location may be changed by defining the environment variable **$MEUSERPATH**(5).

### User Profile Files

Files contained in the user profiles typically include:

<*name*>`.emf` - The users start up profile - <*user*>.emf.
<*name*>`.edf` - The users spelling dictionary.
<*name*>`.erf` - The users registry configuration file.

These files are automatically set up by MicroEmacs. These files are established from the menu "**Tools** -> **User Setup**". The "**Setup Path**" item defines the location of the files, but must be MANUALLY included in the `$MEPATH` environment.

### Company Profiles

Company profiles include standard files and extensions to the standard files which may be related to a company, this is typically <*company*>.emf where <*company*> is the name of the company.

The directory may also include template files **etf**(8) files which defines the standard header template used in the files. Files in the "`company`" directory would over-ride the standard template files.

## SEE ALSO

**$user-name**(5), **$user-path**(5), **$MEINSTALLPATH**(5), **$MENAME**(5), **$MENAME**(5), **$MEPATH**(5), **$MEUSERPATH**(5), Company Profiles, File Hooks, File Language Templates, User Profiles.

Ref: Installation(1) File: m2nar001.2 Date: 2025/08/28

## PATCHING

This page describes how to patch a MicroEmacs installation with a later release without re-installing.

### Windows

Windows requires careful patching in order not to disturb the registry settings The following procedure should be followed.

- Move to the jasspa directory, `c:\ProgramFiles\JASSPA\Microemacs`.
- COPY the executable `me32.exe` to a backup, you MUST copy the file otherwise the registry settings will change. DO NOT MOVE THE FILE.

  ```
  copy me32.exe me32_2002_12.exe
  ```

- Save the macros directory, these may be moved i.e.

  ```
  rename macros macros_2002_12
  ```

- Unpack and install the new Windows executable and COPY over the original executable that was previously backed up i.e.

  ```
  copy c:\path-to-executable\me32-win32-20040108-rc1.exe me32.exe
  ```

- Create a new macros directory and install the macros into this new directory, do this manually or using a utility like WINZip.

  ```
  mkdir macros
  cd macros
  unzip c:\path-to\memacros-20040108-rc1.zip
  ```

- The new version of MicroEmacs will now operate with the old installation environment. You may return to the old release by reversing the steps to patch the installation.

### UNIX/Linux

For ***NIX** package installations then patching is a little easier. The steps are defined as follows:

- Move to the jasspa directory.

  ```
  cd /opt/jasspa/bin
  ```

- Save the original executable.

  ```
  mv me me_200212
  ```

- Unpack the new executable and make it executable.

  ```
  gunzip me-sun-solaris-20040108-rc1.gz
  mv me-sun-solaris-20040108-rc1 me
  chmod a+x me
  ```

- Save the existing macros directory.

```
cd /opt/jasspa
mv macros macros_200212
```

- Create a new macros directory and install the new macros:

```
mkdir macros
cd macros
unzip /path-to/memacros-20040108-rc1.zip
```

- Installed packages are built with the install path built into the executable, this may be different from the patch version of the executable so the environment variable **$MEINSTALLPATH**(5) may be defined in the environment with the macro installation search paths.

## SEE ALSO

**$MENAME**(5), **$MEINSTALLPATH**(5), Company Profiles, Installation, File Hooks, File Language Templates, User Profiles.

## BUILD

MicroEmacs 2026 may be compiled from the source files using the command shell build scripts *build.sh* (UNIX Bourne Shell) or *build.bat* (DOS/Windows). A default compile sequence may be achieved with a simple:

```
build.sh
```

from the command line. The build script attempts to detect the host system and available compiler and build the editor.

The build script recognizes the following options:

**-32** Build 32 bit based program.

**-64** Build 64 bit based program.

**-a** *architecture*
Specifics the target CPU chip or architecture, typically one of `intel`, `arm`, `aarch` or `apple`.

**-C** Build clean. Delete all of the object files.

**-D** *<define>[=<value>]*
Sets build define **<define>** with value "`<value>`" or empty string if value not given.

**-d** Build a debugging version, the output is to a platform specific `debug` build directory rather than a `release` one.

**-h** Display a simple help page.

**-l** *logfile*
Redirect all compilation output to the *logfile*, this may not work on DOS or Windows systems.

**-la** *logfile*
Append all compilation output to the end of *logfile*, this may not work on DOS or Windows systems.

**-m** *makefile*
Build using the specified makefile. over-riding the auto system detect. The supplied makefiles include:

- `linuxgcc.mak` Linux using GNU tools
- `macoscc.mak` macOS using Xcode command-line tools
- `winmsvc.mak` Windows using MS Visual C++
- `winmingw.mak` Windows using MinGW and GNU tools
- `cygwingcc.mak` Cygwin (on Windows) using GNU tools
- `msyswingcc.mak` MSYS (on Windows) using GNU tools
- `dosdj2.mak` Microsoft DOS using djgpp version 2.

The following makefiles are also included for historical record, but will likely need significant updating to make them work with the current release:

- `aix4cc.mak` IBM AIX 4.x native
- `aix4gcc.mak` IBM AIX 4.x using GNU tools
- `aix5cc.mak` IBM AIX 5.x native
- `aix5gcc.mak` IBM AIX 5.x using GNU tools
- `darwingcc.mak` Apple Darwin (console and X-Windows)
- `dosdj1.mak` Microsoft DOS build using djgpp version 1
- `freebsdcc.mak` Free BSD native
- `freebsdgcc.mak` Free BSD using GNU tools
- `hpux9cc.mak` HP-UX 9.x native
- `hpux9gcc.mak` HP-UX 9.x using GNU tools
- `hpux10cc.mak` HP-UX 10.x native
- `hpux10gcc.mak` HP-UX 10.x using GNU tools
- `hpux11cc.mak` HP-UX 11.x native
- `hpux11gcc.mak` HP-UX 11.x using GNU tools
- `irix5cc.mak` Silicon Graphics IRIX 5.x native
- `irix5gcc.mak` Silicon Graphics IRIX 5.x using GNU tools
- `irix6cc.mak` Silicon Graphics IRIX 6.x native
- `irix6gcc.mak` Silicon Graphics IRIX 6.x using GNU tools
- `openbsdgcc.mak` OpenBSD using GNU tools
- `openstepcc.mak` Openstep 4.2 on NeXTstep (BSD 4.3)
- `osf1gcc.mak` OSF1 using GNU tools
- `sunos5cc.mak` Sun Solaris 5.6/7/8/9/10 native
- `sunos5gcc.mak` Sun Solaris 5.5/6/7/8/9/10 using GNU tools
- `sunosx86gcc.mak` Sun Solarais 2.6/7/8/9/10 (Intel) using GNU tools.
- `zaurusgcc.mak` Zauras palm using GNU tools

**-ne** Build NanoEmacs (a cut down version aimed as a vi replacement), the output is `ne`.

**-S** Build spotless. Deletes all of the object files and any backup files, tag files etc.

**-t** *type* Set the build type, where *type* can be one of the following:

**-v** <*variable*>=<*value*>
Build with the given makefile <**variable**> override, e.g. `-v OPENSSLPATH=<path-to-openssl`

- `c` Build a console only version (i.e. no window support), the output is `mec`.
- `w` Build a windows only version (i.e. no console support), the output is `mew`.
- `cw` Build a version which supports both console and windows, the output is `mecw`.

### Makefiles

The supplied makefiles are provided in two forms:

- ***gcc.mak** - GNU make, using gcc.
- ***cc.mak** - Native make, consistent with the compiler and platform.

The makefiles are supplied with the following targets:

- **all** - Default build.
- **clean** - Removes intermediate files.
- **spotless** - Removes intermediate files and any backup files.
- **me** - Builds the default MicroEmacs executable.
- **med** - Builds a debugging version.
- **mec** - Builds a console only version.
- **mew** - Builds a windows only version.
- **mecw** - Builds a version which supports both console and windows.
- **ne** - Builds NanoEmacs executable.

## WINDOWS ENVIRONMENT

The Windows environment is a little more difficult than UNIX because it does not have a good native command line environment. There are two Microsoft Developer options for building. If you have not purchased MS-DEV then use the FREE Borland compiler which runs on the command line.

### MinGW Build

A collection of freely available and freely distributable Windows specific header files and import libraries combined with GNU toolsets that allow one to produce native Windows programs that do not rely on any 3rd-party C runtime DLLs. Available from http://www.mingw.org, **mingw** allows a Microsoft Windows version MicroEmacs to bu built and maintained.

### MS-Dev GUI Build

MicroEmacs is supplied with a MS-Dev 6.x project for building. This has only been set up for debugging and has limited build options, it will also produce a large executable image. The MS-Windows images shipped by JASSPA are built using the command line build environment which is detailed in a later section.

### MS-Dev Command Line Build

The command line build with MS-Dev is a little messy to set up, in the following sections we discuss MS-Dev 6.x under Windows 2000. There will be differences with different operating systems versions, different install paths and different versions of the MS-Dev toolkit. Use the following information as an outline ONLY, JASSPA will not take responsibility for the configuration of the command line environment, if in doubt then use the GUI builder.

To run **nmake**(1) from a DOS box then the MS-Dev compiler tools must be available in the command path. The MS-Dev kit V6.x is supplied with the necessary file to set up the path which is called `VCVARS32.BAT` and is typically located in directory:

```
C:\Program Files\Microsoft Visual Studio\VC98\Bin
```

This should be run first and then the command line make files may be run using nmake i.e. to build ME:

```
nmake -f win32v6.mak me
```

To build NE:

```
nmake -f win32v6.mak ne
```

To clean up after:

```
nmake -f spotless
```

If you are going to perform a lot of command line building then it is easier to set up the environment and not have to run `VCVARS32.BAT` every time you compile.

You can add the variables to your environment, you will need to expand the `VCVARS32.BAT` script and then add the results. The following example is for MS-Dev 6.0 under W2K, the settings will differ depending on what O/S variant you are running and where you have installed software. The following ONLY serves as an example, you need to tailor for your installation.

- To set up the variables then **Start** -> **Settings** -> **Control Panel** which displays the Control Panel items.
- Select **System** -> **Advanced** -> **Environment variables**
- In **User Variables for** <*User*> then **New...** or **Edit...** the `PATH` variable. Add the command line tools execution paths to the environment variable, this will be something like the following (but without newlines):

  ```
  Variable Name:  PATH
  Variable Value: %PATH%;
  C:\PROGRA 1\MICROS 3\Common\msdev98\BIN;
  C:\PROGRA 1\MICROS 3\VC98\BIN;
  C:\PROGRA 1\MICROS 3\Common\TOOLS\WINNT;
  C:\PROGRA 1\MICROS 3\Common\TOOLS;
  ```

- Similarly add the `INCLUDE` variable:

  ```
  Variable Name:  INCLUDE
  Variable Value: C:\PROGRA 1\MICROS 3\VC98\ATL\INCLUDE;
  C:\PROGRA 1\MICROS 3\VC98\INCLUDE;
  C:\PROGRA 1\MICROS 3\VC98\MFC\INCLUDE;
  %INCLUDE%
  ```

- Similarly add the `LIB` variable:

  ```
  Variable Name:  LIB
  Variable Value: C:\PROGRA 1\MICROS 3\VC98\LIB;
  C:\PROGRA 1\MICROS 3\VC98\MFC\LIB;
  %LIB%
  ```

- Accept the entries and close down the dialog. Start a new DOS command line and the variables should be set up ready for the build.

**NOTES**

Other UNIX ports should be fairly easy from the base set of ported platforms. If any new platform ports are performed by individuals then please submit the makefiles and any source changes back to JASSPA - see Contact Information.

Ref: Build(2) File: m2nar007.2 Date: 2026/02/16

## CLIENT-SERVER

This sections describes how MicroEmacs 2026 may be interfaced to external components through the **Client-Server** interface.

The **Client-Server** interface of MicroEmacs provides a capability for other applications to inject commands into a running version of the editor, which are interpreted and executed. The interface is only available on multi-tasking operating systems such as UNIX and Microsoft Windows; it is not available on MS-DOS systems.

Within the following discussions, the **Server** is a running version of the MicroEmacs editor; the **client** is the application (or shell script) that communicates a new command to the *server*.

The **Client-Server** interface may provide a bidirectional interface such that a *client* may submit a command to the *server* and may also retrieve a response to that command.

## DESCRIPTION

The **Client-Server** interface operates by making an external interface available which is continually monitored by the *server*. The external interface may be provided by a file, named pipe or socket (depending upon the platform) with a well know location in the file system. Typically two files are provided, an input file into which the *client* writes commands (*$TEMP*/**me***$MENAME*.**cmd**); and an output file where responses to those commands my be read (*$TEMP*/**me***$MENAME*.**rsp**).

Within MicroEmacs, the client server interface appears as a hidden **ipipe-shell-command**(2) buffer, with the name `*server*`. Commands are received through this buffer and responses are written back to the buffer.

### Client Commands

*Clients* may write directly to the *command* through the use of explicit embedded code, or may use a **me**(1) invocation with the **-m** option. Commands to the client interface take the form "**C:**<*client*>**:**<*command*>".

<*client*>

<*client*> is an identification string that may be used to identify the client, this information may be used when the command is handled to interpret the command if some special client specific action is required.

<*command*>

The <*command*> is an editor command (or macro) of the given name with any arguments. Standard command escape sequences must be adhered to. i.e. to write "`Hello World`" on the message line then a client may issue the command:

```
me -m "C:<client>:ml-write \"Hello world\"
```

A new line character may be included in the string using `\\n`, if the message is not terminated with a new line then on is automatically added. The *client-server* interface is typically used to load a file, this may be performed as follows:

```
me -m "C:<client>:find-file \"/path/foo.bar\""
```

The absolute path is specified in this type of transaction as the current working directory of the active MicroEmacs session is unknown. The **-m** option de-iconize's the existing editor session and bring it to the foreground.

### Client Responses

Responses from *client* commands are written to the response file, responses take a similar form to *client* commands except they are prefixed by an **R**, i.e. "**R:**<*client*>**:**<*data*>".

As multiple *clients* may be utilizing the *client-server* mechanism then the <*client*> sting passed in the command is typically returned in the response to allow the *client* to identify it's own response (rather than any other *clients*. It is the *clients* responsibility that this string is unique in order that it may be differentiated.

The returned <*data*> format is undefined and would be generated by a macro command used to handle the *client* command; sufficient to say that the data should exist on a single line.

### Server Side

On the *server* side, the **Client-Server** interface is managed like an **ipipe-shell-command**(2) using the hidden buffer `*server*` (as previously mentioned).

The *Client-Server* interface is enabled from the **user-setup**(3) interface, the user setting of the interface is confirmed by checking bit `0x20000` of the **$system**(5) variable.

The client server interface is typically initialized within the `me.emf` initialization file, whereby the *ipipe* input handler is bound to the client pipe buffer and the buffer is hidden, so it is not available when the buffers are swapped. (Note that the client buffer may be explicitly interrogated using find-buffer `*server*`). The client handler is installed as follows:

```
; Setup the Client Server
!if &band $system 0x20000
    define-macro-file meserver server-input
    find-buffer "*server*"
    set-variable :last-line 2
    set-variable :client-list ":"
    set-variable $buffer-ipipe server-input
    beginning-of-buffer
    goto-alpha-mark  "I"
    -1 find-buffer "*server*"
!endif
```

This binds a MicroEmacs macro called *server-input* to handle the client commands as they arrive on the input, an alpha-mark is used to record the processed position at the end of the buffer. The pipe handler itself decodes the client request and executes it. The default handler supplied with MicroEmacs is defined within the macro file `meserver.emf`

Responses to the client are inserted into the response file by writing directly into the ipipe buffer (`*server*`) using the **ipipe-write**(2) command. It is the calling macros responsibility to ensure that the response string adheres to the format outlined above in the previous sections.

### NOTES

It is not possible to kill the `*server*` buffer, and **ipipe-kill**(2) is ignored within the context of the buffer.

### FILES

`meserver.emf` - Default Client-Server ipipe handler.
*$TEMP*/**me***$MENAME*.**cmd** - Command file.
*$TEMP*/**me***$MENAME*.**rsp** - Response file.

## BUGS

The first MicroEmacs session that executes becomes the editor server, additional editor sessions that are executed do not become server processes. In the event that the *server* editor is terminated, any other sessions do not take over the role of server. Subsequently issuing a client command may fail, or invoke a new editor session which adopts the role of server.

## SEE ALSO

**me**(1), **ipipe-shell-command**(2)

Ref: Client-Server(2) File: m2nar005.2 Date: 2025/08/28

## COMPANY PROFILES

This section describes how a company profile should be incorporated into MicroEmacs. A company profile defines a set of extensions to MicroEmacs which encapsulate settings which are used on a company wide basis. This type of configuration is typically used with a networked (shared) installation. The company profile would typically include:

- Name of the company.
- Standard header files including company copyright statements.
- Standard file layouts
- Company defined language extensions.

### Location Of The Company Information

It is suggested that all of the company extensions applied to MicroEmacs are performed in a separate directory location which shadows the MicroEmacs standard macro file directory. This enables the original files to be sourced if a user does not want to include the company files. This method also allows MicroEmacs to be updated in the future, whilst retaining the company files.

The company files are generally placed in the `company` sub-directory described in the installation notes.

### Content Of The Company Information

**Company macro file**

The company file is typically called by the company name (i.e. `jasspa.emf`) create a new company file. The file includes your company name and hook functions for any new file types that have been defined for the company, an example company file for **Jasspa** might be defined as:

```
; -!- emf -!-
; Copyright (C) 2025 JASSPA (www.jasspa.com).
;
; This is part of JASSPA's MicroEmacs, see the LICENSE file for licensing and
; copying information.
;
; Synopsis:    JASSPA company setup file
; Authors:     Steven Phillips
;
; Define the name, url & license of the company.
set-variable %company-name "Jasspa"
set-variable %company-url "www.jasspa.com"
set-variable %company-license "This is part of JASSPA's MicroEmacs, ..."
; Add Jasspa specific file hooks
; Make-up foo major-mode and file hook
1 major-mode "foo"
; Override the make with localised build command
set-variable %compile-com "build"
```

The file contains company specific major-modes and the name of the company.

**Other Company Files**

Files defined on behalf of the company are included in the company directory. These would include:

- Template header files **etf**(8).
- Major-mode file hook definitions (**hk***XXX***.emf**) for company specific files, see **major-mode**(2).
- Extensions to the standard hook definitions (**my***XXX***.emf**) for company specific language extensions to the standard hook files. See File Hooks and File Language Templates.

## SEE ALSO

**$MEINSTALLPATH**(5), **$MENAME**(5), **$MEPATH**(5), File Hooks, **major-mode**(2), File Language Templates, Installation, **user-setup**(3), User Profiles.

Ref: CompanyProfiles(2) File: m2nar002.2 Date: 2026/01/17

## INTERFACING

This sections describes how MicroEmacs 2026 may be interfaced to external components.

### Shells

A shell window may be opened within the context of the editor using the command **ishell**(3), whereby an interactive command shell is presented within a buffer.

In the Microsoft Windows environment a **cygwin** UNIX style BASH shell may be realised with the **cygwin**(3) command.

### Debugger

Within the UNIX environment the GNU **gdb**(1) or native UNIX **dbx**(1) debuggers may be invoked from the editor using **gdb**(3) or **dbx**(3). respectively This invokes the debugger and follows the debugging process in the editor window, automatically opening the source files as the debugger calls for them.

### File Searching

File searching is performed using **grep**(1) using the **grep**(3) command. For Windows then the GNU grep utility is recommended, for MS-DOS then the DJGPP version of GNU grep is recommended.

### File Differencing

Differencing files, or directories is performed using the **diff**(1) utility using the **diff**(3) command. For all platforms the GNU diff utility is recommended as this provides a comprehensive differencing that is not typically available with native UNIX diff utilities.

### Tag Files

A **tag** capability exists (see **find-tag**(2)) such that source functions and alike may be located quickly using a **tags** file. The standard **ctags**(1) format is used by MicroEmacs. The **tags** file itself may be generated by MicroEmacs from the menu (*Tools->XX Tools->Create Tags File*). Alternatively a **tags** file may be generated by the **ctags**(1) utility. This is typically standard on UNIX platforms. For Windows and DOS platforms then the **Exuberant Ctags** is recommended, this is available from:

```
https://ctags.sourceforge.net/
```

A MicroEmacs compatible tags file may be generated using the command line "`ctags -N --format=1 .`" cataloging the current directory. To generate **tags** for a directory tree then use "`ctags -NR --format=1 .`". Refer to the **Exuberant Ctags** documentation for a more detailed description of the utility.

### Compilation

Compilation is performed using the **compile**(3) command. This invokes a command shell, typically using **make**(1) to initiate a build sequence.

### Client-Server

The Client-Server interface allows other client applications to inject commands into an already existing MicroEmacs session (the server), thereby controlling the editor remotely. This is typically used to inject new files into the editor to be presented to the user.

The *Client-Server* interface is available in both the UNIX and Microsoft Windows environments.

### Command Line Filer

MicroEmacs may be invoked as a command filter in it's own right, macro scripts have been developed to perform a **dos2unix**(1) conversion operation, generate tags files etc. See Command Line Filters.

### SEE ALSO

**ctags**(1), **compile**(3), **cygwin**(3), **dbx**(3), **diff**(3), **find-tag**(2) **gdb**(3), **grep**(3), **ishell**(3), Client-Server, Command Line Filters.

Ref: Interfacing(2) File: m2nar004.2 Date: 2025/08/28

## FILE LANGUAGE TEMPLATES

MicroEmacs 2026 provides a large range of macros and templates to deal with the most commonly occurring types of ASCII file that may be edited. However, there is a requirement for users to extend this capability to include more obscure file types, in addition to bespoke files found internally within organizations, or devised by the user.

For each file type, MicroEmacs may be tailored to recognize the file and modify it's hilighting, key binding configuration, osd display and indentation to accommodate the file. In addition, new shorthand macros may be introduced to help deal with the contents of the file.

This section outlines the steps to be taken to integrate a new file language template into MicroEmacs. **buffer-init**(3) automates many of the operations defined below and it is suggested that this is a preferable method to add a new language template into MicroEmacs.

### The scope of the File Type

The first step is to decide the scope of the file, this will determine where the file hook should be defined. The options are:

**A standard file type not supported**
: If this is a standard file type not supported by MicroEmacs then it should be added to `me.emf`, in addition contact us and we will add it to the standard release. Any macro files associated with this file type should be available globally and are added to the MicroEmacs *macro* directory.

**Local To your organization**
: If it is a file type local to your organization then it should be added to your *company*.emf file. Any macro files associated with the file type should be added to your local company MicroEmacs directory.

**Local to an individual**
: If this is a file type that is only used by a limited number of individuals then it should be added to the *user*.emf file. Any files associated with the file type are added to your local user MicroEmacs directory.

### Recognizing the File Type

The next step to adding a new file type is to get MicroEmacs to recognize the file as the new type. Recognition is performed by the File Hooks which perform recognition on the file extension and/or the file content. The name of the file type must be determined, this is typically the name of the file prepended by `hk`. e.g. a file with extension *foo* uses the file `hkfoo.emf` for it's language specific definitions.

Using the **add-file-hook**(2) invocation the file recognition is bound to the file hook macro whenever the file type is loaded. The file hook is added to the appropriate global, company or user start up file as determined in step 1. The file hooks for file *foo* might be defined as follows, depending upon the recognition method:

**Recognizing the extension**
: To recognize the file extension, then a space separated list of extensions may be defined, including the dot '`.`' (or other) extension separator.

```
add-file-hook ".foo"        fhook-foo
```

**Recognizing a magic editor string in the file**

If the file type adopts multiple extensions (or does not use a file extension) then an editor specific string may be inserted into the file to enable the editor to recognize it, typically of the form −!− *type* −!−, if the string is GNU Emacs compatible then the −∗− convention may be used. The binding is defined as:

```
-1 add-file-hook "-!-[ \t]*foo.*-!-"            fhook-foo
```

**Recognizing a magic string in the file**

UNIX files use a "#!<path>" notation for executable ASCII files. If the file is this type of file (or uses any other type of common string in the as the first characters of a file) then the binding may be defined as follows, in this case we have assumed *foo* is the UNIX executable variety i.e. #!/usr/local/bin/foo:

```
1 add-file-hook "^#!/.*foo" fhook-foo
```

Any, or all of the above recognition methods may be employed to invoke the language specific macro. Note that the methods are evaluated in a LIFO order, hence it is possible to over-ride an existing method.

### Defining the Macro File

Once the hook has been defined, the language specific file must be created. Create the language specific file with the same name as defined in the hooks, removing the **fhook-** prefix and replacing it with **hk**, i.e. fhook-foo invokes the language specific file hkfoo.emf. Create, the file and add the file hook macro. for example hk*foo*.emf contents may be defined as:

```
define-macro fhook-foo
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
ml-write "[MicroEmacs foo file hook loaded]"
```

The file hook may be tested by exiting and re-loading MicroEmacs, or simply by executing the file containing the add-file-hook function. Once the file bindings are installed a *foo* file may be loaded and the hook message should be displayed.

The best way to create a new file hook is to find an existing file type with similar support to the new file hook you wish to create and re-use the hook code to create the new one.

### Modifying an Existing file hook

The standard file hooks supplied with MicroEmacs should not be modified, typically a user will want to extend the repertoire of hi-lighting tokens to encompass locally defined programming libraries or syntactical extensions, in addition to extending support macros that are associated with the file type. In this case, an extension to the hook function is required. The hook file **my***XXX***.emf**, allows extensions to be made to the **hk***XXX***.emf**, without editing the original file. This may be considered to be an *include* file and is executed, if it exists, after the **hk** file has been executed. i.e. if the hook file **hkfoo.emf** is already defined and extensions are added to **myfoo.emf**.

Note that the **my***XXX***.emf** files do not typically include any **fhook-XXX** functions, the original *fhook* functions would be used. However, if a different buffer environment is required from the one created be the hook, such as a different setting of **tab**(2m) mode, the hook function should be copied to **my***XXX***.emf** and altered appropriately.

Most extensions and changes to the buffer environment may be performed using **my***XXX***.emf**. If the extension macro file includes a macro definition **my-fhook-***XXX*, then **my-fhook***XXX* is automatically executed after **fhook***XXX* has been run.

### Adding Hilighting definitions

File specific hilighting is used to pick out key words and tokens used within the file type, it greatly improves readability; the hilighting is also used for printing. The hilighting is defined within the body of the file and is executed once when the hook file is loaded, this occurs when the hook function is executed. During development of the hilighting code, it is usually necessary to execute the hook buffer to view the effects of any changes to the hilighting.
The hilighting is defined using the command **hilight**(2) which requires a hilighting identifier, used to identify the hilighting scheme. This identifier is dynamically allocated when the hook file is loaded, again using *foo*, the identifier is allocated at the top of the file and is protected such that a value is assigned once only.

```
!if &sequal .hilight.foo "ERROR"
    set-variable .hilight.foo &pinc .hilight.next 1
!endif
```

The variable `.hilight.next` allocates unique hilighting numbers, typically a single hilighting number is consumed, incrementing the `.hilight.next` variable ready for the next allocation. The hilighting color scheme is defined in a macro variable **.hilight.***ext*, where *ext* is the name of the language scheme (i.e. *foo*).
Given a hilighting number, the hilighting scheme may be defined. Each of the tokens in the language is assigned a hilighting color, for our simple *foo* file type:

```
0 hilight .hilight.foo 1                $global-scheme
hilight .hilight.foo 2 "#"              .scheme.comment
hilight .hilight.foo 4 "\"" "\"" "\\"   .scheme.string
hilight .hilight.foo 0 "'.'"            .scheme.quote
hilight .hilight.foo 0 "'\\\\.'"        .scheme.quote ; '\?' quoted char

hilight .hilight.foo 1 "if"             .scheme.keyword
hilight .hilight.foo 1 "then"           .scheme.keyword
hilight .hilight.foo 1 "else"           .scheme.keyword
hilight .hilight.foo 1 "endif"          .scheme.keyword
```

When the hilighting tokens have been defined, the hilighting scheme is bound to the buffer. This is performed by assigning **$buffer-hilight**(5) with the hilighting scheme within the *fhook* macro body, e.g.

```
define-macro fhook-foo
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
```

Putting it all together `hkfoo.emf` now comprises:

```
!if &sequal .hilight.foo "ERROR"
    ; Allocate a hilighting scheme number
    set-variable .hilight.foo &pinc .hilight.next 1
!endif

; Define the hilighting scheme
0 hilight .hilight.foo 1                $global-scheme
hilight .hilight.foo 2 "#"              .scheme.comment
hilight .hilight.foo 4 "\"" "\"" "\\"   .scheme.string
hilight .hilight.foo 0 "'.'"            .scheme.quote
hilight .hilight.foo 0 "'\\\\.'"        .scheme.quote ; '\?' quoted char

hilight .hilight.foo 1 "if"             .scheme.keyword
hilight .hilight.foo 1 "then"           .scheme.keyword
hilight .hilight.foo 1 "else"           .scheme.keyword
hilight .hilight.foo 1 "endif"          .scheme.keyword
```

```
; File hook - called when new file is loaded.
define-macro fhook-foo
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro

; Notification that hook is loaded.
ml-write "[MicroEmacs foo file hook loaded]"
```

### Adding a Template

A template inserts initial text into a new file that is created. This mechanism is typically used to insert a standard header into the file on creation. The insertion text is defined within a template file, given the file extension **etf**(8), which is created in the corresponding global, company or user directory as determined in step 1. The template is named *ext*.etf, so for our example file *foo*, the template file is called `foo.etf`. We shall simply add a file header, our comment is `#` (as defined by the hilighting tokens). Our example *foo* template file `foo.etf` may be defined as follows:

```
#-!- foo -!- ################################
#
#  Created By    : $USER_NAME$
#  Created       : $ASCII_TIME$
#  Last Modified : <160495.1521>
#
#  Description
#
#  Notes
#
#  History
#
#  Copyright (c) $YEAR$ $COMPANY_NAME$.
##############################################
```

The template file must be explicitly loaded by the hook file, within the **fhook** function. A new file condition may be tested within the fhook macro by checking the numerical argument, an argument of 0 indicates that this is a new file. The template file is inserted with an invocation of **insert-template**(3). The **fhook** macro checks the argument and inserts the template file as follows:

```
; File hook - called when new file is loaded.
define-macro fhook-foo
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "foo"
    !endif
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
```

### Adding abbreviations

Abbreviations are short-cut expansions which may be defined for the language specific file. The abbreviations are defined in a **eaf**(8) file, *ext*.eaf, located in the appropriately defined MicroEmacs directory. The abbreviation file defines the key sequences which may be automatically inserted, under user intervention, using **expand-abbrev**(2). An abbreviation file for *foo*, `foo.eaf`, may be defined as:

```
if "if \p\rthen\rendif\P"
el "else\r\p\P"
```

The binding to the hook is defined in the *fhook* macro using **buffer-abbrev-file**(2). For the example language file *foo* the *fhook* macro becomes:

```
; File hook - called when new file is loaded.
define-macro fhook-foo
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "foo"
    !endif
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Set the abbreviation file
    buffer-abbrev-file "foo"
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
```

### Automatic Indentation

Automatic indentation may be applied to the file, such that the indentation is automatically performed when new lines are entered into the file. Indentation also benefits from automatic re-styling operations using **restyle-region**(3) and **restyle-buffer**(3).

The indentation style is declared by defining language tokens that constitute positions in the syntax where the indentation is changed. The indentation requires a unique identifier to identify the indentation style, the hilighting identifier is used. If hilighting is not defined, then the language template may still obtain an identifier as described in the hilighting section.

The indention is create with an argument of 0 to the **indent**(2) command, the subsequent tokens are defined using **indent** with no argument. For our simple *foo* syntax then the indentation might be defined as follows:

```
0 indent  .hilight.foo 2 10
indent .hilight.foo n "then" 4
indent .hilight.foo s "else" -4
indent .hilight.foo o "endif" -4
```

This provides an indentation of the form:

```
if condition
then
    XXXX
else
    if condition
    then
        XXXX
    endif
endif
```

The indentation is bound to the buffer in the *fhook* macro by defining **$buffer-indent**(5). For the example file *foo* then the *fhook* is defined as:

```
; File hook - called when new file is loaded.
define-macro fhook-foo
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "foo"
    !endif
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Assign the buffer indentation
    set-variable $buffer-indent .hilight.foo
    ; Set the abbreviation file
    buffer-abbrev-file "foo"
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
```

### Setting Buffer Modes

Buffer modes which are to be adopted (or discarded) by the language specific file are defined in the *fhook* macro. Typical modes that are applied are:

time — Enables time stamping on the file, modifying the time stamp field with the modification date and time.

indent — Automatic indentation, where the cursor is returned to the same column on entering a new line, rather than to the start of the line.

As an example, the *foo fhook* file becomes:

```
; File hook - called when new file is loaded.
define-macro fhook-foo
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "foo"
    !endif
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Assign the buffer indentation
    set-variable $buffer-indent .hilight.foo
    ; Set the abbreviation file
    buffer-abbrev-file "foo"
    ; Set up the buffer modes
    1 buffer-mode "time"
    1 buffer-mode "indent"
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
```

### Assigning New Bindings

New bindings and language specific macros may be added to the language specific file. New macros, to extend the repertoire of commands specifically developed for the language file are defined within the macro body using **define-macro**(2) these are automatically loaded when the hook file is loaded, which in turn is loaded when the file type is identified and loaded.
New bindings, which may be associated with new macros or existing commands, are assigned within the *fhook* macro. As an example, we shall extend the *foo* language file to include a commenting and uncommenting macros, locally binding the macros to the keys "C-c C-c" and "C-c C-d" respectively. The macro definitions are defined as follows:

```
; Macro to comment a line
define-macro foo-comment-line
    !while &gre &pdec @# 1 0
        beginning-of-line
        insert-string "#"
        beginning-of-line
        forward-line
    !done
!emacro

; Macro to remove a comment from a line
define-macro foo-uncomment-line
    !while &gre &pdec @# 1 0
        beginning-of-line
        -1 search-forward "#"
        backward-delete-char
        forward-line
    !done
!emacro
```

The key bindings for the macros are defined for the local buffer ONLY, as such are added using **buffer-bind-key**(2). The bindings are declared in the *fhook* macro as follows:

```
; File hook - called when new file is loaded.
define-macro fhook-foo
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "foo"
    !endif
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Assign the buffer indentation
    set-variable $buffer-indent .hilight.foo
    ; Set the abbreviation file
    buffer-abbrev-file "foo"
    ; Set up the buffer modes
    1 buffer-mode "time"
    1 buffer-mode "indent"
    ; Set up local bindings
    buffer-bind-key foo-comment-line "C-c C-c"
    buffer-bind-key foo-uncomment-line "C-c C-d"
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
```

### Allowing Other to Modify the Hook

Other users of the file hook may need to modify or extend the file hook, the most common form is the addition of user specific hilight tokens. MicroEmacs uses a simple mechanism of executing a user hook extension file if it exists. The extension file name must be of the form **my***XXX***.emf**, i.e. for our example it must be "`myfoo.emf`". This is performed at the end of the macro file so that anything within the file can be altered, it is executed as follows:

```
; load in user extensions if found
!force execute-file "myfoo"
```

Note the **!force**(4) directive is used as the file may not exist.

### Summing Up

The previous sections have presented the basic steps involved in setting up a new language file template. They cater for simple file types, for more complex examples then browse the **hk***xxx*.emf files.

The completed files that should have been generated by following the previous examples are now presented:

**file.foo**

```
# This is a comment.
if condition
then
    do something
else
    if condition
    then
        do something
    endif
endif
```

**hkfoo.emf**

```
!if &sequal .hilight.foo "ERROR"
    ; Allocate a hilighting scheme number
    set-variable .hilight.foo &pinc .hilight.next 1
!endif

; Define the hilighting scheme
0 hilight .hilight.foo 1                $global-scheme
hilight .hilight.foo 2 "#"              .scheme.comment
```

```
hilight .hilight.foo 4 "\"" "\"" "\\"   .scheme.string
hilight .hilight.foo 0 "'.'"            .scheme.quote
hilight .hilight.foo 0 "'\\\\.'"        .scheme.quote ; '\?' quoted char

hilight .hilight.foo 1 "if"             .scheme.keyword
hilight .hilight.foo 1 "then"           .scheme.keyword
hilight .hilight.foo 1 "else"           .scheme.keyword
hilight .hilight.foo 1 "endif"          .scheme.keyword

; File hook - called when new file is loaded.
define-macro fhook-foo
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro

; Define the indentation scheme
0 indent  .hilight.foo 2 10
indent .hilight.foo n "then" 4
indent .hilight.foo s "else" -4
indent .hilight.foo o "endif" -4

; Reset the hilighting printer format and define the color bindings.
0 hilight-print .hilight.foo
hilight-print .hilight.foo "i"  .scheme.comment
hilight-print .hilight.foo "b"  .scheme.keyword
hilight-print .hilight.foo "bi" .scheme.string .scheme.quote

; Macro to comment a line
define-macro foo-comment-line
    !while &gre &pdec @# 1 0
        beginning-of-line
        insert-string "#"
        beginning-of-line
        forward-line
    !done
!emacro

; Macro to remove a comment from a line
define-macro foo-uncomment-line
    !while &gre &pdec @# 1 0
        beginning-of-line
        -1 search-forward "#"
        backward-delete-char
        forward-line
    !done
!emacro

; File hook - called when new file is loaded.
define-macro fhook-foo
    ; if arg is 0 this is a new file so add template
    !if &not @#
        insert-template "foo"
    !endif
    ; Assign the hilighting
    set-variable $buffer-hilight .hilight.foo
    ; Assign the buffer indentation
    set-variable $buffer-indent .hilight.foo
    ; Set the abbreviation file
    buffer-abbrev-file "foo"
    ; Set up the buffer modes
    1 buffer-mode "time"
    1 buffer-mode "indent"
    ; Set up local bindings
    buffer-bind-key foo-comment-line "C-c C-c"
    buffer-bind-key foo-uncomment-line "C-c C-d"
    ; Temporary comment to make sure that it works.
    ml-write "Loaded a foo file"
!emacro
```

```
; Notification that hook is loaded.
ml-write "[MicroEmacs foo file hook loaded]"

; load in user extensions if found
!force execute-file "myfoo"
```

### foo.eaf

```
if "if \p\rthen\rendif\P"
el "else\r\p\P"
```

### foo.etf

```
#-!- foo -!- ################################
#
#  Created By    : $USER_NAME$
#  Created       : $ASCII_TIME$
#  Last Modified : <160495.1521>
#
#  Description
#
#  Notes
#
#  History
#
#  Copyright (c) $YEAR$ $COMPANY_NAME$.
############################################
```

## SEE ALSO

**add-file-hook**(2), **buffer-abbrev-file**(2), **buffer-init**(3), **insert-template**(3), **execute-buffer**(2), **expand-abbrev**(2), **global-abbrev-file**(2), **hilight**(2), **scheme-editor**(3), **indent**(2), **indent**(2m), **restyle-buffer**(3), **restyle-region**(3), **time**(2m), **$buffer-major-mode**(5), **$buffer-hilight**(5), **$buffer-indent**(5), **etf**(8), **eaf**(8),
File Hooks.

## LOCALE SUPPORT

Locale support within MicroEmacs handles the hardware and software configuration with respect to location, including:

Character Sets
Word Characters
Keyboard Support
Language
Spell Support

There are many other locale problems which are not addressed in this help page. Supporting different locale configurations often requires specific hardware (a locale specific keyboard) and knowledge of the language and customs of the region. This makes it a very difficult area for one localized development team to support, as such, JASSPA rely heavily on the user base to report locale issues.

### Note on Names and IDs

The language name is not sufficient to identify a locale (Mexican Spanish is different to Spanish Spanish) neither is the country name (two languages are commonly used in Belgium), so before we've really started the first problem of what to call the locale has no standard answer! Call it what you like but please try to call it something meaningful so others may understand and benefit from your work.

In addition, the *internal id* and *data file* names have a length limit of just four characters due to the "`8.3`" naming conversion of MS-DOS. The standard adopted by JASSPA MicroEmacs for the internal locale id is to combine the 2 letter ISO language name (ISO 639-1) with the 2 letter ISO country name (ISO 3166-1). Should the locale encompasses more than one country, then the most appropriate *country id* is selected.

### Character Sets

A character set is the mapping of an integer number to a display symbol (i.e. character). The ASCII standard defines a mapping of numbers to the standard English characters, this standard is well defined and accepted, as a result the character set rarely causes a problem for plain English.

Problems occur when a language uses characters located outside the ASCII standard, such as letters with accents, letters which are not Latin based (e.g. Greek alphabet) and particularly with languages that have a large number of 'letters' or logograms (e.g. Chinese) which require multiple bytes to encode. There are many different character sets to choose between and if the wrong character set is selected then the incorrect character translation is performed resulting in an incorrect command behaviour and/or character display.

For several reasons the core of MicroEmacs can only support a single byte character set so cannot support languages like Chinese.

### Internal Character Set

Before the adoption of international standards such as Unicode, each operating system vendor would typically create their own character set, or code page, that best met the needs of their customers. As a result there are many in use making it impossible to properly support them

all directly. The use of a standard internal character set allows for the creation of a single set of spelling rules and dictionary that can be used by all compatible character sets with a simple mapping.

MicroEmacs does not intrinsically require, nor is it biased towards any one character set or group of sets, however the main Windows code pages (i.e. CP-1252, CP-1253 etc) have been adopted as these typically have the most defined characters providing the best language support. Therefore all supported languages use the most appropriate Windows character set for their spelling rules and dictionaries.

### Displayed Character Set

The current display character set being used is not determined by the current operating system, but by the font being used in MicroEmacs (which may be limited be the OS). For example, UNIX X-Term systems typically use ISO-8859 based fonts but fonts using other code pages, such as CP-1252 which have numerous additional important characters such as a Euro symbol, can be found in `pcf` format, installed and used by MicroEmacs.

If the character display looks incorrect in MicroEmacs, such as text containing incorrect or weird symbols, use the **insert-symbol**(3) command the review the display character set actually being used and verify what it is, then use the platform page of **user-setup**(3) command to check and correct the font and character set M-S is configured to use.

If the problem persists (i.e. because the character set used to write the text is not supported on your current system) use the **change-buffer-charset**(3) command to convert the text to the current display character set.

If your character-set is not supported then first make sure that MicroEmacs will draw all of the characters to be used. By default MicroEmacs does not draw some characters directly as the symbol may not be defined. When a character is not defined then there will typically be a gap or space in the text at the unknown character, in some cases there may be no space at all which will make it very hard to use. The **insert-symbol**(3) command (**Edit->Insert Symbol**) is a good way of looking at which characters can be used with the current character set.

For a character to be rendered (when in main text) or poked (drawn by **screen-poke**(2) or **osd**(2)) is defined by the `d` and `p` flags of **set-char-mask**(2) command. The characters that are used when drawing MicroEmacs's window boarders or **osd** dialogs is set via the **$box-chars**(5) and **$window-chars**(5) variables.

MicroEmacs attempts to improve the availability of useful graphics characters on Windows and UNIX X-Term interfaces. The characters between 0 and 31 are typically control characters with no graphical representation (e.g. new-line, backspace, tab etc.) if bit 0x10000 of the **$system**(5) variable is set then MicroEmacs renders its own set of characters. These characters are typically used for drawing boxes and scroll-bars.

With so many character sets, each with their own character mappings, then the problem of spelling dictionary support is also tied to the locale. MicroEmacs uses the ISO standard character sets (ISO 8859) internally for word and spelling support and therefore a mapping between the ISO standard and the user character set is required. This mapping is defined by using the '`M`' flag of the **set-char-mask**(2) command.

The user may declare the current character set in the platform page of **user-setup**(3). All the settings required for supporting each character set may be found in the `charset.emf` macro file, so if your character set is not supported, this is the file to edit.

### Unicode support

Unicode support within MicroEmacs is very limited due to its single byte character set limitation, however flag **c** of command **set-char-mask**(2) can be used to map characters 0x80 to 0xff of the display character set to their Unicode equivalent. Correctly mapping these characters allows better handling of text transfer between MicroEmacs and the system clipboard (see bit **0x800000** of **$system**(5)) and allows conversion to and from UTF-8 using the **change-buffer-charset**(3) command.

The use of character `0x07` (Bell) is used to denote an unsupported character while **change-buffer-charset** has an option to preserve unsupported Unicode characters by encoding them to a 3 or 5 digit hexadecimal number prefixed with the character `0x01` or `0x02`. Care must be taken to avoid splitting or corrupting the resultant 4 or 6 byte strings, however once editing is complete the **change-buffer-charset** command can be used to convert the buffer back to UTF-8 before saving.

### Word characters

Word characters are those characters which are deemed to be part of a word, numbers are usually included. Many MicroEmacs commands use the 'Word' character set such as **forward-word**(2) and **upper-case-word**(2). The characters that form the word class should be determined by the display character set being used rather than by the language, this is because which character are letters within the display character set doesn't change and flagging all letters improves general usability. For example, English does not typically use accented letters but it does inherit words that do, eg. *fianc'e*, so treating `'e` ('e' acute) as a letter always makes more sense.

The 'a' flag of command **set-char-mask**(2) is used to specify whether a character is part of a word, you must specify the uppercase letter and then the lowercase equivalent so the case conversion functions work correctly.

This may unfortunately be made a little more tricky by the requirement that this list must be specified in the most appropriate character set (see **Internal Character Set** section). When extending the word character set the characters have to be mapped to the current character set which may not support all the required characters. For example in the PC-437 DOS character set there is an e-grave (`` `e``) but no E-grave so the E-grave is mapped to the normal E. As a result, if trying to write French text the case changing commands will behave oddly, for example:

```
r`egle -> REGLE -> r`egl`e
```

The conversion of all 'E's to '`E' is an undesirable side effect of '`E' being mapped to E. This can be avoided by redefining the base letter again at the end of the word character list, for example:

```
set-char-mask "a" "`E`eEe"
```

### Keyboard Support

The keyboard to character mapping is defined in the Start-Up page of **user-setup**(3), where the keyboard may be selected from a list of known keyboards. If your keyboard is not present, or is not working correctly, then this section should allow you to fix the problem (please send JASSPA the fix).

Most operating systems seem to handle keyboard mappings with the exception of MS-Windows which requires a helping hand. The root of the problems with MS-Windows is it's own locale character mappings which change the visibility status of the keyboard messages which conflict with Emacs keystroke bindings. To support key-bindings like 'C-tab' or 'S-return' a low level keyboard interface is required, but this can lead to strange problems with the more obscure

keys, particularly with the 'Alt Gr' accented letter keys. For example on American keyboards pressing 'C-#' results in two 'C-#' key events being generated, this peculiarity only occurs with this one key. On a British keyboard the same key generates a 'C-#' followed by a 'C-\'. This problem can be diagnosed using the **$recent-keys**(5) variable. Simply type an obvious character, e.g. 'A' then the offending key followed by another obvious key ('B'), then look for this key sequence in the **$recent-keys** variable (use the **list-variables**(2) or **describe-variable**(2) command). So for the above British keyboard problem the recent-keys would be:

```
B C-\\ C-# A
```

($recent-keys lists the keys backwards). Once you have found the key sequence generated by the key, the problem may be fixed using the **translate-key**(2) to automatically convert the incorrect key sequence into the required key. For the problem above the following line is required:

```
translate-key "C-# C-\\" "C-#"
```

Note that once a key sequence has been translated everything, including **$recent-keys**, receive only the translated key. So if you a suspected a problem with the existing definition, change the keyboard type in **user-setup** to **Default** so no translations are performed, quit and restart MicroEmacs before attempting to re-diagnose the problem.

All the settings required for supporting each keyboard may be found in the `keyboard.emf` macro file, so if your keyboard is not supported, this is the file you need to edit.

### Language

The current language can be set in the Start-up page of **user-setup**(3), setting the language lets to the following two variables being set:

**.spell.language**
: This variable defines the 4 letter ID used for the language which is made up of the 2 letter ISO language code followed by the 2 letter ISO country code, see *Note on Names and IDs* above. The value of this variable determines which spell checking rules and dictionary are loaded.

**.change-font.ln-type**
: This variable is a string whose characters determine which character set groups can support the language, for example Francais is set to "24" indicated that both character sets of group **2** and **4** can support the language, these character sets include CP-1252, ISO-8859-1, CP-850 & CP-437 from group **2** and CP-1254 & ISO-8859-9 from group **4**. The first character set group given by this variable, 2 in this case, defines the main or preferred character set group, and the first character set within this group, see variable **.change-font.cs-type** defined in **charset.emf** defines the primary or internal character set used by that group.

  If your language is not supported you will need to add it to the list and define these two variables found in the `language.emf` macro file.

### Spell Support

The current language is set using the Language setting on General page of **user-setup**(3), if your required language is not listed you must first create the basic language support by following the guide lines in the **Word Character** section above. If you Language is listed, select it and enable it by either pressing **Current** or saving and restarting MicroEmacs. in a suitable test buffer run the spelling checker, one of three things will happen:

The `Spelling Checker` dialog opens and spelling is checked

The spelling checker is supported by the current language and can be used (the rules and dictionaries have been downloaded and installed).

Dialog opens with the following error message:

```
Rules and dictionaries for language "XXXX"
   are not available, please download.
```

The spelling checker is supported by the current language but the required rules and dictionaries have not been downloaded. You should be able to download them from the JASSPA website, see Contact Information. Once downloaded they must be placed in the MicroEmacs search path, i.e. where the other macro files (like `me.emf`) are located.

Dialog opens with the following error message:

```
Language "XXXX" not supported!
```

The spelling checker is not supported by the current language, see the following **Adding Spell Support** section.

### Adding Spell Support

To support a language MicroEmacs's spelling checker requires a base word dictionary and a set of rules which define what words can be derived from each base word in the dictionary. The concept and format of the word list and rules are compatible with the **Free Software Foundation** GNU **ispell**(1), **myspell**(1) and **hunspell**(1) packages. Not all features of hunspell are supported by MicroEmacs, most notably composite rules and right to left languages.

The best starting point is to obtain rules and dictionary files (<lang>.aff & <lang>.dic) for one of these packages, the web can usually yield these. Once these have been obtained the rules file (or affix file) must be converted to a MicroEmacs macro file calling the **add-spell-rule**(2) command to define the rules. The rule file should be named "`lsr`<*lang-id*>`.emf`" where "<*lang-id*>" is the spelling language id, determined by the **.spell.language** variable set in the `language.emf` macro file.

The `spellutl.emf` macro file contains the command **ispell-convert** which will attempt to convert the aff and dic files and macros **spell-check-list**, **spell-check-guess** and **ispell-test** to aid conversion accuracy verification. See existing spelling rule files (`lsr*.emf`) for examples and help on command **add-spell-rule**(2).

**Note**: the character set used by the rules should be the most appropriate for the language (see **Internal Character Set**section), this can make the process more difficult, but adding support for the base character sets involved to `charset.emf` will enable the command **change-buffer-charset**(3) to function correctly and avoid some of the more subtle issues that can arise from incompatible character sets. If you are having difficulty with this please e-mail JASSPA Support.

Once the conversion process has completed it should have generated a "`lsr`<*lang-id*>`.emf`" rules file and a "`lsm`<*lang-id*>`.edf`" dictionary file. With the rapid increase in memory capacity there is little point these days in splitting dictionaries into two files so the use of an extended dictionary file for obscure words has been dropped.

Once the generated word and dictionary files have been place in the MicroEmacs search path, the spelling checker should find and use them. Please submit your generated support to MicroEmacs for others to benefit.

## SEE ALSO

**set-char-mask**(2), **translate-key**(2), **add-spell-rule**(2), **change-buffer-charset**(3), **user-setup**(3), **$box-chars**(5), **$recent-keys**(5). **$search-path**(5), **$user-path**(5), **$window-chars**(5).

Ref: localeSupport(2) File: m2nar008.2 Date: 2024/05/21

## REGULAR EXPRESSIONS

Regular Expressions are used in the search (and replace) operations. The following notes are applicable when **magic**(2m) mode is enabled.

### Overview

A "*regular expression*" (or "*regex*", or "*pattern*") is a text string that describes some (mathematical) set of strings. A regex **R** "*matches*" a string **S** if **S** is in the set of strings described by **R**.

Some regular expressions match only one string, i.e., the set they describe has only one member. For example, the regular expression '`foo`' matches the string '`foo`' and no others. Other regular expressions match more than one string, i.e., the set they describe has more than one member. For example, the regular expression '`f*`' matches the set of strings made up of any number (including zero) of '`f`'s. As you can see, some characters in regular expressions match themselves (such as '`f`') and some don't (such as '`*`'); the ones that do not match themselves instead let you specify patterns that describe many different strings.

### Syntax of Regular Expressions

Regular expressions have a syntax in which a few characters are special constructs and the rest are "*ordinary*". An ordinary character is a simple regular expression which matches that same character and nothing else. The special characters are '`$`', '`^`', '`.`', '`*`', '`+`', '`?`', '`[`', '`]`' and '`\`'. Any other character appearing in a regular expression is ordinary, unless a '`\`' precedes it.

For example, '`f`' is not a special character, so it is ordinary, and therefore '`f`' is a regular expression that matches the string '`f`' and no other string. (It does **not** match the string '`ff`'.) Likewise, '`o`' is a regular expression that matches only '`o`'. (When case distinctions are being ignored, these regexs also match '`F`' and '`O`', but we consider this a generalization of "*the same string*", rather than an exception.)

Any two regular expressions `A` and `B` can be concatenated. The result is a regular expression which matches a string if `A` matches some amount of the beginning of that string and `B` matches the rest of the string.

As a simple example, we can concatenate the regular expressions '`f`' and '`o`' to get the regular expression '`fo`', which matches only the string '`fo`'. Still trivial. To do something nontrivial, you need to use one of the special characters. Here is a list of them.

**.** (Period)
: is a special character that matches any single character except a newline. Using concatenation, we can make regular expressions like '`a.b`', which matches any three-character string that begins with '`a`' and ends with '`b`'.

***** (asterisk)
: is not a construct by itself; it is a postfix qualifier that means to match the preceding regular expression repetitively as many times as possible. Thus, '`o*`' matches any number of '`o`'s (including no '`o`'s).

  '`*`' always applies to the **smallest** possible preceding expression. Thus, '`fo*`' has a repeating '`o`', not a repeating '`fo`'. It matches '`f`', '`fo`', '`foo`', and so on.

  The '`*`' qualifier is greedy, which means the matcher processes the construct by matching, immediately, as many repetitions as can be found. Then it continues with

the rest of the pattern. If that fails, backtracking occurs, discarding some of the matches of the '*'-modified construct in case that makes it possible to match the rest of the pattern. For example, in matching 'ca*ar' against the string 'caaar', the 'a*' first tries to match all three 'a's; but the rest of the pattern is 'ar' and there is only 'r' left to match, so this try fails. The next alternative is for 'a*' to match only two 'a's. With this choice, the rest of the regex matches successfully.

**+** (plus) is a postfix qualifier, similar to '*' except that it must match the preceding expression at least once. So, for example, 'ca+r' matches the strings 'car' and 'caaaar' but not the string 'cr', whereas 'ca*r' matches all three strings. This qualifier is also greedy, see above.

'**?**' (question mark)
is a postfix qualifier, similar to '*' except that it can match the preceding expression either once or not at all. For example, 'ca?r' matches 'car' or 'cr'; nothing else. This qualifier is also greedy, see above.

***?** (asterisk-question mark)
is a postfix qualifier, similar to '*' except that it is lazy, i.e. not greedy. When processing a lazy qualifier, the matcher first matches the fewest allowed repetitions (zero in this case) before continuing with the rest of the pattern. It is only when a match fails and backtracking occurs that more repetitions are matched if possible. So, for example, 'c.*?r' will return a match of 'car' for 'carrrrd' rather than 'carrrr' as 'c.*r' would do. Both would return 'carrrrd' if a 'd' was added to the regex.

**+?** (plus-question mark)
is a lazy postfix qualifier to match one or more occurrences of the preceding expression.

**??** (question mark-question mark)
is a lazy postfix qualifier to match zero or one occurrence of the preceding expression.

**[ ... ]** is a "character set", which begins with '[' and is terminated by ']'. In the simplest case, the characters between the two brackets are what this set can match.

Thus, '[ad]' matches either one 'a' or one 'd', and '[ad]*' matches any string composed of just 'a's and 'd's (including the empty string), from which it follows that 'c[ad]*r' matches 'cr', 'car', 'cdr', 'caddaar', etc.

You can also include character ranges in a character set, by writing the starting and ending characters with a '-' between them. Thus, '[a-z]' matches any lower-case ASCII letter. Ranges may be intermixed freely with individual characters, as in '[a-z$%.]', which matches any lower-case ASCII letter or '$', '%' or period.

The word and digit special characters (i.e. \w, \l, etc.) are recognized within the character set.

To include a ']' in a character set, you must make it the first character. For example, '[]a]' matches ']' or 'a'. To include a '-', write '-' as the first or last character of the set, or put it after a range. Thus, '[]-]' matches both ']' and '-'.

To include '^' in a set, put it anywhere but at the beginning of the set.

When you use a range in case-insensitive search, you should write both ends of the range in upper case, or both in lower case, or both should be non-letters. The behavior of a mixed-case range such as '`A-z`' is somewhat ill-defined, and it may change in future Emacs versions.

[^ ... ] '`[^`' begins a "*complemented character set*", which matches any character except the ones specified. Thus, '`[^a-z0-9A-Z]`' matches all characters *__except__* letters and digits.

'`^`' is not special in a character set unless it is the first character. The character following the '`^`' is treated as if it were first (in other words, '`-`' and '`]`' are not special there).

A complemented character set can match a newline, unless newline is mentioned as one of the characters not to match. This is in contrast to the handling of regexs in programs such as **grep**(1).

^ (caret) is a special character that matches the empty string, but only at the beginning of a line in the text being matched. Otherwise it fails to match anything. Thus, '`^foo`' matches a '`foo`' that occurs at the beginning of a line.

**$** (dollar)
is similar to '`^`' but matches only at the end of a line. Thus, '`x+$`' matches a string of one '`x`' or more at the end of a line.

\ (backslash)
has two functions: it quotes the special characters (including '`\`'), and it introduces additional special constructs.

Because '`\`' quotes special characters, '`\$`' is a regular expression that matches only '`$`', and '`\[`' is a regular expression that matches only '`[`', and so on.

**Note:** for historical compatibility, special characters are treated as ordinary ones if they are in contexts where their special meanings make no sense. For example, '`*foo`' treats '`*`' as ordinary since there is no preceding expression on which the '`*`' can act. It is poor practice to depend on this behavior; it is better to quote the special character anyway, regardless of where it appears.

For the most part, '`\`' followed by any character matches only that character. However, there are several exceptions: two-character sequences starting with '`\`' that have special meanings. The second character in the sequence is always an ordinary character when used on its own. Here is a table of '`\`' constructs.

\— (bar)
specifies an alternative. Two regular expressions `A` and `B` with '`\|`' in between form an expression that matches some text if either `A` matches it or `B` matches it. It works by trying to match `A`, and if that fails, by trying to match `B`.

Thus, '`foo\|bar`' matches either 'foo' or 'bar' but no other string.

'`\|`' applies to the largest possible surrounding expressions. Only a surrounding '`\( ... \)`' grouping can limit the grouping power of '`\|`'.

Full backtracking capability exists to handle multiple uses of '`\|`'.

\( ... \) is a grouping construct that serves three purposes:

- To enclose a set of '\|' alternatives for other operations. Thus, '\(foo\|bar\)x' matches either 'foox' or 'barx'.
- To enclose a complicated expression for the postfix operators '*', '+' and '?' to operate on. Thus, 'ba\(na\)*' matches 'bananana', etc., with any (zero or more) number of 'na' strings.
- To record a matched substring for future reference. This last application is not a consequence of the idea of a parenthetical grouping; it is a separate feature that is assigned as a second meaning to the same '\( ... \)' construct. In practice there is no conflict between the two meanings.

'**D**' matches the same text that matched the Dth occurrence of a '\( ... \)' construct.

After the end of a '\( ... \)' construct, the matcher remembers the beginning and end of the text matched by that construct. Then, later on in the regular expression, you can use '\' followed by the digit D to mean "match the same text matched the Dth time by the '\( ... \)' construct."

The strings matching the first nine '\( ... \)' constructs appearing in a regular expression are assigned numbers 1 through 9 in the order that the open-parentheses appear in the regular expression. So you can use '\1' through '\9' to refer to the text matched by the corresponding '\( ... \)' constructs.

For example, '\(.*\)\1' matches any newline-free string that is composed of two identical halves. The '\(.*\)' matches the first half, which may be anything, but the '\1' that follows must match the same exact text.

If a particular '\( ... \)' construct matches more than once (which can easily happen if it is followed by '*'), only the last match is recorded.

\` matches the empty string, but only at the beginning of the buffer or string being matched against.

**NOTE:** This currently only matches the start of the current line - it does not match the start of the buffer.

\' matches the empty string, but only at the end of the buffer or string being matched against.

**NOTE:** This currently only matches the end of the current line - it does not match the end of the buffer.

\= matches the empty string, but only at point.

\< matches the empty string, but only at the beginning of a word. '\<' matches at the beginning of the buffer only if a word-constituent character follows.

\> matches the empty string, but only at the end of a word. '\>' matches at the end of the buffer only if the contents end with a word-constituent character.

\a matches any alpha characters.

\A matches any non-alpha characters.

`\b` matches the empty string, but only at the beginning or end of a word. Thus, '`\bfoo\b`' matches any occurrence of '`foo`' as a separate word. '`\bballs?\b`' matches '`ball`' or '`balls`' as a separate word. '`\b`' matches at the beginning or end of the buffer regardless of what text appears next to it.

`\B` matches the empty string, but *not* at the beginning or end of a word.

`\d` matches any digit character.

`\D` matches any non-digit character.

`\h` matches any hexadecimal digit character.

`\H` matches any non-hexadecimal character.

`\l` matches any lowercase word-constituent character. The syntax table determines which characters these are.

`\L` matches any character that is not a lowercase word-constituent.

`\m` matches any alphanumeric characters

`\M` matches any non alphanumeric characters.

`\s` matches any white-space character, including new-line.

`\S` matches any character that is not white-space.

`\u` matches any uppercase word-constituent character. The syntax table determines which characters these are.

`\U` matches any character that is not a uppercase word-constituent.

`\w` matches any word-constituent character. The syntax table determines which characters these are.

`\W` matches any character that is not a word-constituent.

`\{N,M\}`
is a greedy postfix qualifier to matches an integer number of the previous item, where `N` and `M` are integer constants interpreted as follows:

- `\{N\}` The preceeding item is matched exactly `N` times.
- `\{N,\}` The preceeding item is matched `N` or more times.
- `\{N,M\}` The preceeding item is matched at least `N` times, but no more than `M` times.
- `\{,M\}` The preceeding item is optional and is matched at most `M` times.

`\{N,M\}?`
is the lazy form or the previous postfix qualifier, see above for the meaning of `N` and `M`.

The constructs that pertain to words and syntax are controlled by the setting of the syntax table.

### Syntax of Replacement Expressions

A regular expression replacement, **query-replace-string**(2) command (with **magic**(2m) mode enabled), replaces exact matches for a single string or pattern. The replacement pattern may be a constant; it may also refer to all or part of what is matched by the regular expression search string. The following replacement constructs are supported:

**\&** In the replacement pattern, **\&** stands for the entire match being replaced (as does \0).

**\D** In the replacement pattern, where **D** is a digit 1-9, stands for whatever matched the Dth parenthesized grouping (\( .. \)) in search pattern. To include a '\' in the text to replace with, you must enter '\\'. For example,

```
M-x query-replace-string<RET> c[ad]+r <RET> \&-safe <RET>
```

replaces (for example) "cadr" with "cadr-safe" and "cddr" with "cddr-safe".

```
M-x query-replace-string<RET> \(c[ad]+r\)-safe <RET> \1 <RET>
```

performs the inverse transformation.

**\0** is a special case, this represents the whole of the search pattern, it is equivalent to **\&**.

**\c***D* In the replacement pattern capitalizes the string, where *D* is a digit 0-9 or &, stands for whatever matched the Dth parenthesized grouping (\(..\)) in search pattern.

**\l***D* In the replacement pattern down-cases the string, where **D** is a digit 0-9 of &, stands for whatever matched the Dth parenthesized grouping (\(..\)) in search pattern.

**\u***D* In the replacement pattern up-cases the string, where **D** is a digit 0-9 or &, stands for whatever matched the Dth parenthesized grouping (\(..\)) in search pattern.

**\{***D*,L,R,N**\}**
In the replacement pattern the **\{...\}** qualifier can be used to enforce a minimum length for a group string, where *D* is the group number, l is the left pad character, R the right pad, and N the minimum length. If the group string to be inserted is already N characters or longer then no padding characters are inserted, otherwise an equal number of L and R characters are inserted to pad the inserted string to **N** character, effectively centring the group. If L is omitted the inserted group will be effectively left aligned, if **R** is omitted instead the group with be right aligned. Note the ',' separator character can be changed to any character that is not L or R.

**\{c***D*,L,R,N**\}**, **\{l***D*,L,R,N**\}**, **\{u***D*,L,R,N**\}**
As with **\{***D*,L,R,N**\}** except the case of the group string is altered accordingly.

*****macro-name [ *arg1 arg2 ...* ]
The string to be inserted is dynamically generated be the given *macro-name*, this must be at the start and the only content of the replacement pattern. The given macro can use the given arguments and the **@s1**(4) group variables to obtain the match string, it must set the **$result**(5) variable to the fully expanded replacement string to be inserted. Note that the macro should not change the current window, buffer, cursor position or edit the current buffer in any way, the effect of doing so is undefined but a crash is likely.

### Searching and Case

Searching may be either case sensitive or case insensitive, and is controlled by the **exact**(2m) mode. When *exact* mode is enabled (default) searches are case sensitive; disabled then case is ignored. The **exact**(2m) mode is set on a per-buffer basis.

## NOTES

Greedy qualifiers will return the longest string that they can matche, however the longest pattern is sometimes not what is actually required. For instance, consider searching for an HTML bracket set. The simplest search is:

```
M-x search-forward "<.*>"
```

Unfortunately, this pattern is not specific enough, given an HTML line:

```
<a href="www.jasspa.com">Jasspa Site</a>
```

Then the pattern matched is actually the whole line as the `.*` matches everything to the last `>`, this is the longest string. To rectify the pattern then we can either use a lazy qualifier, for example:

```
M-x search-forward "<.*?>"
```

Alternatively the pattern can be made more specific, for example the correct search pattern to use in this instance is:

```
M-x search-forward "<[^>]*>"
```

In this case we match any character excluding the closing character, this guarantees that we always find the shortest string match. A search of our HTML line locates two separate instances of the regular expression `<a href="www.jasspa.com">` and `</a>`.

When matching a group with multiple alternatives the matcher will match/use the first one that successfully matches, this means you should order the alternative forms in preference, typically longest first. For example, if trying to match '`ok`' or '`okay`' the pattern should be '`\(okay\|ok\)`' and not '`\(ok\|okay\)`' as this will only ever match the 'ok' in an 'okay'.

## HISTORY

As of March 2005 then the existing Emacs search string syntax of "`\s?`" and "`\S?`" has been discarded in favor of Perl's simpler syntax of "`\s`" and "`\S`" for a whitespace character class. Added new character classes "`\d`" = digits, "`\D`" != digits, "`\h`" = hexdigits, "`\H`" != hexdigits, "`\l`" = lowercase, "`\L`" != lowercase, "`\u`" = uppercase and "`\U`" != uppercase. The regular expression replacement string characters include "`\c#`" "`\l#`" and "`\u#`" to change the case of groups.

In January 2026 the \{...\} and **macro* replacement construct where added.

## SEE ALSO

**search-forward**(2), **search-backward**(2), **buffer-mode**(2), **exact**(2m), **hunt-backward**(2), **hunt-forward**(2), **isearch-forward**(2), **magic**(2m), **replace-string**(2).

Ref: RegularExpressions(2) File: m2nar006.2 Date: 2026/02/05

## USER PROFILES

This section describes how a user profile should be incorporated into MicroEmacs 2026. A user profile defines a set of extensions to MicroEmacs which encapsulates settings which are used by an individual user.

The user profile allows:

- Saving of the last session (history), allowing the next invocation of MicroEmacs to restore your previous session. The **-c** option to **me**(1) restores a previous sessions buffers to the same state prior to exiting the editor.
- Personalized spelling dictionaries.
- Redefinition of MicroEmacs operation, allowing the editor to be tailored to an individual's requirements. Including the re-binding of keys, modification of the screen colors. Definition of personal macros etc.

### Identification

In order to identify a user MicroEmacs uses information in the system to determine the name of the user, and in turn the configuration to use. On all systems the value of the environment variable **$MENAME**(5) takes priority over any other means of user identification. If this variable is not defined then the host system typically provides a mechanism to determine the current user. MS-DOS and Microsoft Windows systems present problems where a login prompt is not used.

Each of the supported platforms are now described.

**UNIX** The login name is determined from the `/etc/passwd` file and defines **$user-name**(5). The environment variable **$MENAME**(5) may be defined alter the default user configuration file name.

**DOS** MS-DOS typically has no concept of the user name. The user name may be defined in the `autoexec.bat` file, choose a name of 8 characters or less, i.e. to fix the user name to `fred` then add the following line:

```
SET MENAME=fred
```

Remember to re-boot the system before the new command takes effect. (see the next step, there is another change to `autoexec.bat`).

**Microsoft Windows**
Microsoft windows environments may, or may not, have login enabled. If login is enabled then **$user-name**(5) is set to the login user name. The login name may be overridden by setting the environment variable **$MENAME**(5).

### Shared Platforms

Platforms may share the same set of configuration files. Consider a system which may boot under MS-DOS, Windows and Linux. Provided that the macro files are located on a file system that may be mounted by all of the other operating systems and the **$MEUSERPATH**(5) and **$MEPATH**(5) are set appropriately to point to the users private MicroEmacs and system directories, then a single set of macro files may be shared across all platforms.

### Personal MicroEmacs Directory

The private user profile is stored in a separate directory. The directory that MicroEmacs uses is typically automatically created at start up, but may be created manually by the user. If the directory is to be placed in a special location then the **$MEUSERPATH**(5) environment variable should be defined and set to point at this directory location.

**UNIX** In the UNIX environment then the local directory is typically created in the users home directory obtained from the `/etc/passwd` file and is called `/.config/jasspa`. MicroEmacs will automatically look for this directory and set the **$user-path**(5) variable to this value. The location may be overridden by setting the environment variable **$MEUSERPATH**(5) to reference a different directory location.

**DOS** For MS-DOS environments, there is typically no user directory, it is suggested that the user directory is created in the MicroEmacs directory, use the `$MENAME` defined in the previous step i.e.

```
mkdir c:\jasspa\fred
```

Change or add the environment variable **$MEUSERPATH**(5) in the **autoexec.bat** to include the new directory i.e.

```
SET MEUSERPATH=c:\jasspa\fred
```

**Windows**

In Microsoft Windows environment using Explorer 4.0 or later then a user application data directory exists (`$APPDATA`) and MicroEmacs will create a directory `jasspa` in this directory location to hold private data. If there is no such data area then all user configuration is stored in the executable directory location (or a sub-directory called **$user-name**(5)).

The user data area **$user-path**(5) may be modified by creating a new directory and setting the environment variable **$MEUSERPATH**(5) to point to this new directory location.

### Creating Your Profile

On starting MicroEmacs for the first time then the user is guided through the configuration file setup to create any local configuration files. Once set up then MicroEmacs may be configured using the **user-setup**(3) dialog:

```
Tools -> User Setup
```

### The User Profile

Files created in the user directory include:

- Setup registry and previous session history $user-name.erf (see **erf**(8)). This stores the **user-setup** settings and the context from previous edit sessions.
- Users start-up file $user-name.emf (see **emf**(8)). This is typically referred to as "*user.emf*" in the documentation and means the users private MicroEmacs macro file. The user may make local changes to MicroEmacs in this file, this may include definition of new key bindings, defining new major-mode functions etc. You should override the standard MicroEmacs settings from your start-up file rather than modifying the standard MicroEmacs files.

- Personal spelling dictionary **lsp***lang*.edf (see **edf**(8)). Multiple spelling dictionaries may be created, one for each language. Any words that are added to the spelling dictionary are stored in these files.

In addition to the above, if new private macro files are constructed then they should be added to this directory (if they are not global to the company).

**EXAMPLE**

The following are examples of some individuals start-up files:

```
; Jon's special settings
;
; Last Modified <190698.2226>
;
; Macro to delete the whitespace, or if an a word all of the
; word until the next word is reached.
define-macro super-delete
    set-variable #l0 0
    !while &not &sin @wc " \t\n"
        forward-char
        set-variable #l0 &add #l0 1
    !done
    !repeat
        !force forward-char
        !if $status
            set-variable #l0 &add #l0 1
        !endif
    !until &or &seq @wc "" &not &sin @wc " \t\n"
    #l0 backward-delete-char
    !return
!emacro
; Make a previous-buffer command.
define-macro previous-buffer
    &neg @# next-buffer
!emacro
; spotless; Perform a clean and remove any multi-blank lines.
define-macro spotless
    -1 clean
!emacro
; comment-adjust; Used for comments in electric-c mode (and the other
; electic modes. Moves to the comment fill position, saves having to mess
; around with comments at the end of the line.
0 define-macro comment-adjust
    ; delete all spaces up until the next character
    !while &sin @wc " \t"
        forward-delete-char
    !done
    ; Fill the line to the current $c-margin. We use this as
    ; this is the only variable that tells us where the margin
    ; should be.
    !if &gre $window-acol 0
        backward-char
        !if &sin @wc " \t"
            forward-delete-char
            !jump -4
        !else
            forward-char
        !endif
    !endif
    ; Now fill to the $c-margin
    &sub $c-margin $window-acol insert-string " "
!emacro
; Macro to force buffer to compile buffer for C-x '
define-macro compile-error-buffer
    !force delete-buffer *compile*
    change-buffer-name "*compile*"
!emacro
```

```
;
; Set up the bindings.
;
global-bind-key super-delete            "C-delete"
global-bind-key beginning-of-line       "home"
global-bind-key end-of-line             "end"
global-bind-key undo                    "f4"
!if &seq %emulate "ERROR"
    global-bind-key comment-adjust      "esc tab"
    global-bind-key comment-adjust      "C-insert"
    ; Like a korn shell please.
    ml-bind-key tab "esc esc"
!endif
;
; Setup for windows and UNIX.
;
; Define my hilighting colour for Windows and UNIX.
!if &equ &band $system 0x001 0
    !if &not &seq $platform "windows"
        ; Small bold font is better for me.
        change-font "-*-clean-medium-r-*-*-*-130-*-*-*-*-*-*-*"
    !endif
!endif
; Change the default diff command-line for GNU diff utility all platforms
set-variable %diff-com "diff --context --minimal --ignore-space-change --report-identical-files --recurs
set-variable %gdiff-com "diff --context --ignore-space-change -w"
; Setup for cygwin
!if &seq $platform "windows"
    set-variable %cygwin-path "c:/cygwin"
    set-variable %cygwin-hilight 1
!endif
; Set up the http flags.
set-variable %http-flags "cspw"
; Add Jasspa MicroEmacs File (JMF) support:
11 major-mode "jmf" "jm-file" "Jasspa ME File"

; Finished
ml-write "Configured to Jon's requirements"
```

## SEE ALSO

**user-setup**(3), **$search-path**(5), **$user-name**(5), **$user-path**(5), **$MEINSTALLPATH**(5), **$MENAME**(5), **$MEPATH**(5), **$MEUSERPATH**(5), **major-mode**(2), Company Profiles, File Hooks, File Language Templates, Installation.

Ref: User Profiles(2) File: m2nar003.2 Date: 2026/02/17

## MACRO DEBUGGING

In writing new macros then you need to debug them, there are various strategies as follows:

### Tracing

The simplest method is to add debugging information that is printed on the message line that traces the execution sequence as it progresses through the macro.
Use **ml-write**(2) to print out values as you go, use

```
1000 ml-write &spr "Got here, value [%s]" #l1
```

to see the values. The integer argument is a delay in milliseconds, the command above will print the message and then wait 1 second before continuing with execution. The wait allows you to see the message.

Once the macro is debugged then delete or comment out the trace messages.

### Debugger

The macro debugger is a built in debugger that allows the macro execution sequence to be traced and debugged. The debugger is operation is controlled by the variable **$debug**(5) by default the value is `0` which disables the debugger. The debugger displays the next command to execute in the message line and the user may single step through the macro and interrogate the variables.
Use the **$debug** variable for interactive debugging. The best way is to insert the following line somewhere BEFORE the problem occurs where the macro needs to be debugged:

```
set-variable $debug 1
```

As soon as this variable is set then it drops the user into debug mode. If we applied this to our tutorial example then we would define it as:

```
define-macro write-region
    set-variable $debug 1       ; Enable debugger
    copy-region
    set-variable #l0 $buffer-fname
    set-position "\x88"
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0
    !force set-variable #l1 @1
    !if $status
        !force !force @# write-buffer #l1 @mna
    !else
        !force !force @# write-buffer @mna
    !endif
    !if &not $status
        !force delete-buffer $buffer-bname
        goto-position "\x88"
        !abort
    !endif
    delete-buffer $buffer-bname
    goto-position "\x88"
!emacro
```

so as soon as we execute **write-region** then we drop into debug mode. The mode line appears as follows:

```
macro-name:position:number [line] ?
```

where **macro-name** is the name of the macro that is executing, **position** is the line from the start of the macro definition. **line** is the macro line that is being executed. i.e.

```
display-matching-fenc:22:0 [4 show region] ?
```

At the prompt the commands that are available are:

| | |
|---|---|
| `?` | Help, display the debugger commands. |
| `^G` | Abort the macro. Immediately terminates the macro sequence, the state of the buffers is undefined and leaves the windows in the state when the macro was aborted. |
| `^L` | Redraws the current screen. |
| `c` | Continues macro execution the value of $debug is set to 0. Execution continues without user interaction up until the next **$debug** change. |
| `s` | Single steps through the macro, if another macro is invoked then debugging with continue in the new sub-macro. |
| `v` | Display the value of a variable. The user is prompted for a variable value to show, once entered, the value of the variable is displayed. |
| <*any*> | Any other key will step onto the next instruction, unlike *step* then any macros that are invoked are not debugged. |

See the **$debug**(5) help page for more information.
It is important that you SAVE ALL YOUR WORK before starting to debug macros. It is possible to sometimes crash the editor up while developing macros, prior to the January 2005 build then the following example of infinite recursion would destroy the editor:

```
define-macro Test
    Test
!emacro
```

This macro nesting is now caught the nesting is limited to 20 calling macros.

For everyday macros that the typical user will write MicroEmacs will be robust and is not likely to result in any damage. If you start writing macros with for deeply nested macro hooks from the core of MicroEmacs (i.e. ipipe callbacks) then deleting buffers under the editor will cause stability problems when mistakes are made. For performance reasons it is not possible to check for all permutations of every command that could be issued that may possibly change the state of the editor and then recover. Provided that the calls are used within their calling remit then no ill effects will be observed.

### Running macros without the editor

Macro logic can be run and checked without a terminal, and without starting the editor, using the piped mode option **-p**. This is useful for regression tests and for timing, as the macro runs to completion and the process exits.
The macro file is given with @ and it **must** define a **start-up**(3) macro, as there is no user input to drive the editor; without one the editor reports "`Pipe script failed to exit`" and stops. Exit with **exit-emacs**(2), giving a numeric argument of 4 so that the user is not prompted. Use **ml-write**(2) with a numeric argument of -2 to write to the standard output.

```
define-macro start-up
    -2 ml-write &spr "2 + 2 = %d" &add 2 2
    4 exit-emacs 0
!emacro
```

Run with:

```
export MEPATH=<install>/macros:$PWD
mec -p "@./mytest" < /dev/null
```

The output uses carriage returns as line endings, so pipe it through "`tr '\r' '\n'`" when reading it from a shell script. Note that the **-P** option is piped mode with debugging, which waits for input and is not suitable for an unattended run.

See `test/emf/` and `test/hearts/` for worked examples, the latter including a test runner script.

## SEE ALSO

**$debug**(5),
Macro Glossary
MicroEmacs Macro Language at a Glance.
Writing MicroEmacs Macros

Ref: macroDebugging(4) File: m2nar011.2 Date: 2026/08/06

## MACRO LANGUAGE OVERVIEW

MicroEmacs macro language is a run-time interpreted language and uses a prefix notation form, that is operators appear before their arguments, sometimes known as Polish notation, i.e. '`+ 3 4`' rather than the more conventional infix notation '`3 + 4`').

Every line is self contained, you cannot make a single command span multiple line (by using something like a '\' character as in C or tcl).
Comments start with semi-colon '`;`', anything to the right of an unquoted semi-colon is ignored.

```
; I'm a comment and exist to the end of line.
set-variable #l1 1      ; I am a comment to the end of line.
```

There are three types of line

- Blank or comment line
- Process flow control lines using directives (i.e. '`!if #l1`')
- Executable command lines (i.e. `forward-line`)

The first letter of any word or token on a line is very important and informs MicroEmacs and the user what type of token it is. The basic token start characters are:

`;` Comment
`$` System or environment variable
`%` User variable
`:` Buffer variable
`.` Command variable
`#` Register variable
`@` Macro variable
`!` Directive
`&` Function
`"` A string, which finishes at the next unquoted `"`.
`0-9` A number, which is read as a string
`*` A goto label

If its not one of the above then it must be the first letter of a command name.

### Escape Character

**Backslash '\' is escape character in macro scripts, so every normal occurrence needs to be doubled.**

The MicroEmacs macro language uses a backslash '\' character as an escape character. In order to use a backslash character in macro arguments then all backslash characters should be themselves escaped with another backslash i.e. '\' becomes '\\' or '\\' becomes '\\\\'. If the escaping is omitted then it is likely that the command will receive an incorrect character sequence that will be misinterpreted and cause the command to fail or perform an incorrect action.

### Comments

Comments may appear anywhere on a line and everything to the right of the semi-colon to the end of the line is ignored:

```
; this is a comment
set-variable #l0 1    ; this part of the line is a comment.
```

### System or environment variables

MicroEmacs system variables start with '$' and are used to configure most aspects of the editor (such as tab widths, window drawing characters etc.). There are a fixed list of system variables, see MicroEmacs - Macro Language Glossary.

If a given variable name is not a recognized system variable (e.g. $PATH) then MicroEmacs treats it as an environment variable, using **set-variable**(2) to change an environment variable. This is useful when used in conjunction with one of MicroEmacs's shell commands. If the environment variable does not exist its value is returned as the string "ERROR".

### User variables

User variables commence with a percentage character '%' and have a global context within the editor, but unlike the *system variables* they do not span into the environment space (i.e. private to MicroEmacs).

The use of **User Variables** is discouraged because of their scope, historically MicroEmacs only supported system and user variables. If two different macros use the same variable then things go wrong which is principally why they are no longer used. Macros have become a very large part of JASSPA's MicroEmacs and therefore they must be reliable by the nature of their global scope they are considered unreliable.

A few commands still use them but this is more for historical reasons, for example xgrep's **%xgrep-com**(5) variable to set the default grep command would be better implemented as the command variable .xgrep.com, see below.

### Buffer Variables

Buffer variables are set in the context of a buffer, each buffer has its own independent list of buffer variables so this type of variable is used by macros which need to store buffer context information. As an example, a file hook may allow each buffer to have its own private **$fill-ignore**(5) value (the variable is global). It may stores each buffer's own fill ignore in the variable :fill-ignore. This is private to the buffer. Referencing the variable :fill-ignore retrieves the current buffer's :fill-ignore variable, if it not set its value is returned as "ERROR". Referencing :*scratch*:fill-ignore retrieves the "*scratch*" buffer's :fill-ignore value.

The operator **&exist**(4) may be used to determine if the buffer variable has been created and set.

### Command Variables

Command variables are set in the context of a command or macro and are similar to buffer variables. .var retrieves the the current macros command variable whereas .hilight.var will get command **hilight** command variable.

It is important to understand that these variables are stored with the command and the command must exist before the variable is used. As an example of the command variable scope, consider the following macro definitions.

```
define-macro Test1
    ml-write &spr "Test1 .foo is [%s]" .foo
!emacro
set-variable .Test1.foo "hello world"
define-macro Test2
    ml-write &spr "Test1 .foo is [%s]" .Test1.foo
!emacro
```

On running **Test1** and **Test2** both will print "`Test1 .foo is [hello world]`". The initialisation of `.Test1.foo` is performed outside of the macro and is done only once, once the variable has been initialized the line is discarded. This discard reduces the run-time memory overhead.

### Register Variables

Most variables are used to store temporary information, once a macro completes the variable is no longer required. Register variables are used for this purpose, their scope is only the current macro's execution as such they are the most efficient variables to use.

A new set of registers is created every time that a macro is executed, called `#l0` through to `#l9`. Where macros are nested then the parent registers are effectively stacked and may be access through `#p0` through to `#p9`, this is especially useful when writing recursive macros or macros which must return values as they may return a processed result in the parents register. A single set of registers always exists which is the global registers `#g0` through to `#g9`, for a first level macro then this forms its parent register variable.

### Macro variables

Macro variables commence with a at-sign `@` and are only available in the context of a macro and are used to retrieve the numeric argument, parameters etc. The variables include:

**@?** - Boolean determines if a numeric argument was supplied
**@#** - The value of the numeric argument

**@0** - The name of the macro
**@1** - The first argument of macro
**@2** - The second argument of macro
**@3** ... **@***n*

**@p** - The name of the calling (or parent) macro.

### Arguments and returning values

Arguments are not copied into the macro, they are the caller's text and are evaluated each time they are used. Reading `@1` evaluates the caller's first argument at that moment, in the caller's context, and reading it twice evaluates it twice.
This is what allows a macro to **return** a value. Assigning to an argument assigns to whatever the caller named, so by convention a macro takes its results in its last arguments:

```
0 define-macro suit-of
    ; @1 = card, @2 <- the suit of that card
    set-variable @2 &band @1 112
!emacro

suit-of 90 #l3
; #l3 now holds 80
```

The caller may name a register, a buffer variable, a command variable or one of its own arguments; whichever it names is what gets written.
Passing one of the macro's own arguments straight on to another macro is supported, and the value is written back to the frame the argument originally came from rather than the immediate caller. So in the following, `Test`'s `#l0` is the register that ends up holding 500:

```
0 define-macro l2
    set-variable @1 @2
!emacro
```

```
0 define-macro l1
    l2 @1 @2
!emacro
define-macro Test
    l1 #l0 500
    ml-write #l0
!emacro
```

### Directives

The Macro Directives are denoted by a **!** prefix and provide control over the command sequence. Directives are used to define macros and affect the macro execution sequence. i.e.

```
define-macro if-test
    set-variable #l0 1
    !if #l0
        ml-write "you will see this"
        !return
    !elif &not #l0
        ml-write "you will not see this"
        !abort
    !else
        ml-write "you will not see this"
        !abort
    !endif
!emacro
```

The common directives are:

**!abort** !abort terminates a macro and exits with a *false* status.

**!return** !return terminates a macro and exits with a *true* status.

**!while/!continue/!done**
!while defines a conditional loop that is tested before entry to the loop, the !continue statement within the loop allows the loop to be broken. **!while** loops may not be nested, but may be contained within a !repeat/!until construct.

**!repeat/!until**
!repeat defines a conditional loop that is tested on exit from the loop. **!repeat** loops may not be nested, but may be contained within a !while/!done construct.

**define-macro/!emacro**
define-macro defines a new macro which is terminated with a !emacro.

!if/!elif/!else/!endif
!if defines a conditional statement, where there is more than one condition then the !elif directive may continue the conditional test.

**!goto/!tgoto**
!goto and !tgoto provide an absolute unconditional and conditional jump to a ***label** in a file, respectively. A **!goto** label is defined as line commencing with a $\star$ followed by a name i.e. ***label**.

**!force** *command*
!force ignores the return status of a *command* while executing a macro, thereby ignoring any exceptions which would otherwise halt macro execution.

### Functions

All functions are denoted by a **&** prefix, they are perhaps miss named and should be called operators as they operate on variables and literal strings to produce a new result.

There are different type of functions:

- Numeric - interprets arguments as numbers i.e. add two numbers **&add**(4).
- String - interprets arguments as strings i.e. to concatenate two strings together **&cat**(4).
- Logical - boolean operators i.e. greater than **&great**(4).
- Key - key bindings i.e. key binding of command **&kbind**(4).
- Bitwise - bit-wise numerical operators i.e. bitwise AND **&band**(4).
- File - file interaction i.e. find a file **&find**(4).
- Lists - string item lists i.e. find item in list **&lfind**(4).
- Miscellaneous - miscellaneous functions to which query and return information to the caller i.e. registry query **&reg**(4).

The functions use a postfix notation as the *function* appears before the arguments.

```
set-variable #l0 2
set-variable #l1 1
!if &gre #l0 #l1
    ml-write "#l0 is greater than #l1"
!endif

ml-write &cat "#l0 + #l1 = " &add #l0 #l1
```

The interpretation of strings and numbers is discussed in the next section.

### Strings and Numbers

All variables are treated as strings within MicroEmacs so the integer value of 102 is the string `"102"`. Where functions require an integer argument (i.e. &add) the strings are converted to integers, operated on and converted back to a string result. Floating point notation is not understood.

Integers may be expressed in different bases:

`0xXXX` - Hexadecimal notation i.e. *0x12af*.
`0oooo` - Octal notation requires a leading *012673*.
`ddddd` - Decimal notation *12345*.

Strings do not necessarily need to be quoted, provided that the string is unambiguous and does not include any special characters then all input between white spaces is considered to be a string. Strings may be quoted '`"string"`', within a quoted string then a backslash '\'is used as an escape character. the escape sequences that are interpreted include:

`\\` - A backslash character.
`\t` - A literal TAB character.
`\n` - A literal NEWLINE character.
`\xx` - A hexadecimal character.

## SEE ALSO

Introduction to Variable Functions
Macro Glossary
Macro Debugging Crash Course
Writing MicroEmacs Macros

Ref: macroOverview(4) File: m2nar012.2 Date: 2026/08/16

## MACROS

**MicroEmacs** is based on a core set of built in commands and variables that provide a a basic set of functionality, almost all of the visible features of MicroEmacs are implemented or controlled by its macro language. **NanoEmacs** is quite simply this base set of commands of MicroEmacs with no macro support and provides a basic Emacs editor.

MicroEmacs is extended with an interpreted script file that allows new commands to be defined using the base level commands and variables which is referred to as a Macro. Once a macro command has been defined then it appears as a built in command to the user, and as with the existing commands, may have a key binding to invoke the new command.

In addition to the base commands, the macro language includes control statements and logical operators which allow loops and conditional command sequences to control the sequence of operations.

### Starting to write Macros

For new users to MicroEmacs writing macros can be quite daunting, some of the macros shipped with MicroEmacs are complex and have evolved over many years. Beginners should start with simple macros which has verifiable output and can be performed interactively. This makes the task of writing and testing easier, some of the *behind the scene* macros like the mouse driver and auto-spell checker are difficult to get working and requires a greater understanding of MicroEmacs and experience in macro debugging.

A good starting point is to try to write a macro to do something that you find yourself doing all the time, for example reread-file, write-region etc.

**Escape Sequences**

The backslash character '\' is an escape sequence in MicroEmacs macro files, therefore any backslash character that appears in a macro command must be also escaped. i.e. '\' will become '\\' when placed in a macro file.

**Editor Control**

Remember when you write a macro you are placing the editor in control of your macro, when testing a new macro then start a new MicroEmacs session or make sure that if things go wrong you will not loose any work. If you loose control of the editor then **abort** (`C-g`) will typically regain control again.

### Developing your First Macro

In the following example then we walk through the steps to create a macro that saves a region of text to a file (i.e. the text between **point** and **mark**). The resultant macro will be called **write-region**, note that this macro already exists in MicroEmacs.

The first thing to do is walk through the macro that you wish to write interactively in MicroEmacs. Whilst doing this you should decide which approach to take and the commands (or keys) used. Our **write-region** could either use narrows or copy the region to a temporary buffer, a decision has to be made which approach is best to use. Narrows would be more efficient but there are more complications (such as restoring edit mode, avoid changing the **undo** stack etc.). For the moment then the temporary buffer approach is probably the best solution.
Walking through the sequence of steps required to save a region of text then the following command sequence would be executed (assuming a marked up region has been selected):

```
esc w                       ; Copy the region.
```

```
C-x b "tmp-buffer"          ; Create a new temporary buffer.
C-y                         ; Yank copied region into temporary buffer.
C-x C-w "write_region_file" ; Save the buffer to a file.
C-x k RETURN                ; Kill the temporary buffer.
```

This is a fairly easy sequence of steps, the history of key presses can be viewed in the variable **$recent-keys**(5), the command **list-variables**(2) includes this variable in its output.
Macros are not written in key sequences, instead the commands associated with the key bindings are used. A key binding command name may be determined using **describe-key**(2) and **describe-bindings**(2). Some command bindings include a numeric argument, the numeric argument is important (both the value and absence/presence of the argument) and can change the behavior of the command. `C-up` is not '**backward-line**', it is '**5 backward-line**' to go back 5 lines, if you supply a numerical value to the command as well, e.g. '`esc 3 C-up`' you must multiply the user argument with the numerical argument of the binding:

```
3 up   -> 3   'backward-line' -> 3 * 1 backward-line ->  3 backward-line
3 C-up -> 3 '5 backward-line' -> 3 * 5 backward-line -> 15 backward-line
```

Translating the keys that we used into commands then our **write-region** process becomes:

```
copy-region
find-buffer "tmp-buffer"
yank
write-buffer "write_region_file"
delete-buffer RETURN
```

**TIP:** If you are in an MicroEmacs macro buffer (e.g. `foo.emf`) use the **emf-quote-command** command (bound to `A-q`) to make this process trivial. This converts the key binding into a command which is inserted into the buffer.

The obvious problem here is the last line, '`delete-buffer RETURN`'. When running interactively you can simply press RETURN to delete the current buffer however you cannot do this in a macro so you must explicitly specify the buffer name to delete:

```
delete-buffer  "write_region_file"
```

The the process of writing the buffer has the side effect of changing the buffer name to reflect the output file name, as **delete-buffer** follows the **write-buffer** then the name will have changed.
Correcting the delete buffer name, the sequence of commands is wrapped with a macro definition, which binds a name to the command sequence, and a terminator which terminates the macro.

```
define-macro write-region
    copy-region
    find-buffer "tmp-buffer"
    yank
    write-buffer "write_region_file"
    delete-buffer "write_region_file"
!emacro
```

This macro works but is not very effective as the output filename has been fixed, this creates a file "`write_region_file`" in the MicroEmacs start up directory - probably not what is required. The specific macro that has been created must be made interactive, this means it must get filenames from the user, cope with a failure and the user doing silly things. So what must we cope with ? The list of defects is surprisingly long considering we are doing so little:

1. The temp buffer used ("`tmp-buffer`") is not a temporary buffer, temporary buffers must always commence with a '`*`' and should be identifiable with the generating macro.**Note:** knowledge that a buffer is temporary allows MicroEmacs to discard content when shutting down without prompting the user to save.

2. What happens if the "`tmp-buffer`" already exists ?
3. What happens if the "`write_region_file`" buffer already exists ?
4. We need to get the filename to write from the user.
5. What happens if the user aborts during the macro ?
6. What happens if the write fails (disk full or file exists etc.) ?
7. What happens if the user has the current buffer displayed in more than one window ?
8. Does this fit in with other MicroEmacs commands amd macros ? Do we want to use this new command from another context ?

The list of issues is a re-occurring theme, once you have got the hang of this it does become easier. In short reliable macros must:

- Check inputs
- Check outputs
- Handle errors
- Where appropriate handle an unexpected failure.

The last one is really reserved for low level drivers (i.e. mouse, directory listings etc.) which must be bomb proof.

### Generalising Macros

In the previous section we have created a basic macro to save a region of text, in this section we look at that macro is transformed from a static sequence to commands to a fully interactive robust command. Reference should be made to MicroEmacs Macro Language at a Glance for a quick tour of the language syntax.

#### 1. The temp buffer used is not a temporary buffer...

Forming a temporary buffer is simply a naming issue, temporary buffer names are simply started with a "`*`", by convention for a simple buffer name then they are terminated with a "`*`" as well. The temporary buffer name should reflect the name of the original macro hence the buffer name should be something like `*write-region-tmp*`

```
define-macro write-region
    copy-region
    find-buffer "*write-region-tmp*"
    yank
    write-buffer "write_region_file"
    delete-buffer "write_region_file"
!emacro
```

#### 2. What happens if the 'tmp-buffer' already exists ?

Looking at the current implementation if the write fails the temporary buffer will be left lying around. This is very likely to happen and if it does then we will get the last region as well as the next region in the output file. The simple solution is to delete the temporary buffer first, our first attempt to correct this:

```
define-macro write-region
    copy-region
    delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    ....
!emacro
```

Unfortunately this fails if the buffer does not exist, so we must tell MicroEmacs to ignore the failure by **forcing** the execution:

```
define-macro write-region
    copy-region
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    write-buffer "write_region_file"
    delete-buffer "write_region_file"
!emacro
```

### 3. What happens if the 'write_region_file' buffer already exists ?

If the **write_region_file** buffer already exists the buffer being written is renamed to "**write_region_file<1>**" rather than "write_region_file" and so the **delete-buffer** will delete the wrong buffer. Therefore rather than delete an explicitly named buffer the environment variables are used to query the name of the current buffer. The variable **$buffer-bname**(5) returns the current buffer's buffer name and may be used instead of an explicit name.

```
define-macro write-region
    copy-region
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    write-buffer "write_region_file"
    delete-buffer $buffer-bname
!emacro
```

Macro variables may be used to get just about any information you could possibly want. For example **$buffer-fname**(5) could be used after the **buffer-write** to get the file name, **$window-line**(5) provides the current line number etc. See

> Help -> Variable Glossary

for a full list of system variables. There are also **Macro Variables** (@,#,...) which are also very useful, e.g. @wl gives the text on the current line as a string, see the Macro Language Glossary for a full list of these and other macro features such as functions and directives.

### 4. We need to get the destination filename from the user.

Getting input from the user is always dangerous, most macros run so quickly that the user will not have sufficient time to abort the process (C-g) and if they do there is not much that can be done. When collecting input from the user the macro must wait for some input, at this point the user can and should be able to abort the process which must be handled. If the user has not aborted the user still has this uncanny ability of doing the unexpected and the input has to be checked, better still get MicroEmacs to get the input and check it.

There are two approaches to the input problem:

- We get the filename ourselves and pass this to the **write-buffer** command.
- Make **write-buffer** get the input from the user itself as the command would when run from the command key binding.

We would need to use the first approach if we needed the entered value multiple times, in this case we are not so we can use the simpler second approach but for the sake of this document lets consider the first approach.
First stab:

```
set-variable $write-region-file-name @ml "Write-region file"
write-buffer $write-region-file-name
```

This does work but you really do not want to do this ! Consider what happens if the user aborts while entering the file name ? The user would be left in the "*write-buffer-tmp*" buffer and would have to clear up the mess themselves. This may be fixed by catching the abort from the input and deleting the buffer.

```
!force !force set-variable $write-region-file-name @ml "Write-region file"
!if &not $status
    !force delete-buffer $buffer-bname
    !abort
!endif
write-buffer $write-region-file-name
```

**NOTE:** The call to set-variable requires a double !force, a single **!force** allows a macro to continue after a failed command but not after a user interrupt (i.e. a C-g abort), to continue after this requires a double **!force**.

The second problem is with the input type, the @ml does not indicate that a file name is required. The type of input should be qualified to first check the input type and to provide other features such as file completion on file name entry. Specifying the input type correctly:

```
!force !force set-variable $write-region-file-name @ml04 "Write-region file"
!if &not $status
    !force delete-buffer $buffer-bname
    !abort
!endif
write-buffer $write-region-file-name
```

**TIP:** How did we find out about @ml04 ? The best way to find out how to do something is find another macro (or command) that does something very similar and copy it, in this case the diff macro has to perform a similar operation. If you cannot find anything then look in the command documents, the documents generally provide this sort of information (for example see **@ml**(4)). If all else fails then use the News Group, somebody will be able to provide an answer to your question.

In progressing through the example we have used the basic system variables ($var) as they are simple. Other variables exist: user (%var), buffer (:var) or command (.var). All of the previously mentioned variables use valuable system resource and consideration should be given as to whether the value is required once the macro has completed execution. In our case the file name is temporary and is not required outside of the scope of the macro, rather than waste memory then a register variable is preferred as they have zero cost and execute faster than the other variables. Re-writing the macro with register (#lx) variables:

```
!force !force set-variable #l1 @ml04 "Write-region file"
!if &not $status
    !force delete-buffer $buffer-bname
    !abort
!endif
write-buffer #l1
```

**TIP:** Avoid using the global variables (#g1 etc.) as these could be altered by other macros. Sub-macros (i.e. macros called by other macros) can pass values back to their calling macro via the parent's register variables (#p1 etc.). When using parent register variables then ensure that the sub-macro does not destroy some context of the parent macro through altering a register variable that has been used for another purpose.

At the start of this section we said that there were two techniques by which we could get user input, we selected the first (which was less desirable than the second). The approach that we have taken of passing the filename to **write-buffer** appears to be correct, however the **write-buffer** command probably expects a reasonable filename – what happens if the user enters a directory name ? There are other potential esoteric problems with our current approach, all of which may be overcome, but not here.

The second technique, which is by far the best solution, is to simply get the command **write-buffer**(2) to get its input directly from the user. The command does this by default when bound to a key and run from the command line however when run from a macro the behavior is different as there is no command line. We can force this behavior by asking the command to be executed as if it were run from the command line:

```
write-buffer @mn
```

When **write-buffer** asks the MicroEmacs macro processor for a file name the @mn argument tells it to go directly to the user even though ME is currently running a macro. This will only get a single argument from the user, if the file already exits **write-buffer** asks for confirmation that over-writing the existing file is okay, but this argument has not been given so the command fails. To fix this we could do:

```
write-buffer @mn "y"
```

However this has now created even more issues, what happens if this file is already loaded into MicroEmacs, should we allow the user to decide ? We could allow the command line to handle all of the cases:

```
write-buffer @mn @mn @mn @mn ....
```

Which is rapidly becoming very messy for each individual argument, instead we use @mna to ask the command to handle all input associated with the command:

```
write-buffer @mna
```

Note that we should have used this for the first method to protect **write-buffer**:

```
write-buffer #l1 @mna
```

In this case the file name is provided by the argument `#l1` but the remaining arguments are handled internally by the command itself as if invoked from the command line.

Moreover, there is one subtle problem here that can cause a lot of problems especially for other macros trying to use this macro (uses tend to spot the problem themselves and correct it causing only minor frustration). The macro creates a new temporary buffer which has no file name, this means that saving to "`./file`" will write the region to "`file`" in the startup directory rather than in the current buffer's file path. For example, if you start MicroEmacs from "`/bin`", load file "`/tmp/foo.txt`" and `write-region` "`./bar.txt`" you will create the file "`/bin/bar.txt`" not "`/tmp/bar.txt`".

To solve this directory issue the file name of the temporary buffer has to be set to the same filename as the current buffer we are copying from. This is safe as this is a temporary buffer (starts with a '*') so MicroEmacs will not attempt to auto save or allow the buffer to be saved using write-buffer and is performed as follows:

```
define-macro write-region
    copy-region
    set-variable #l0 $buffer-fname       ; ADDED to fix directory location
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0       ; ADDED to fix directory location
    write-buffer @mna
    delete-buffer $buffer-bname
!emacro
```

After all of that, the solution is in fact very simple, however the underlying complexity of the user expectations as to the behavior of the command are complicated.

### 5. What happens if the user aborts during the macro ?

This issue has already been touched on in the previous sections, by using the **@mna** only the **write-buffer** is likely to be aborted while the user is prompted for a file name etc. and if the user manages to abort elsewhere there really is not much that can be done about it without severely impacting the complexity of the macro. Practically it is only necessary to catch the exception when the user is being prompted on the command line with **write-buffer**, as with our existing user input a double **!force** is required to catch an abort at a prompt.

```
define-macro write-region
    copy-region
    set-variable #l0 $buffer-fname
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0
    !force !force write-buffer @mna
    !if &not $status
        !force delete-buffer $buffer-bname
        !abort
    !endif
    delete-buffer $buffer-bname
!emacro
```

The **$status**(5) variable allows us to catch the exit state of the forced command and test for an abort condition following the forced statement. Note that the **$status** value is only valid immediately after the last executed command, executing another command changes the value of **$status**. If the status is required after execution of another command then it may be saved in a variable and used later.

### 6. what happens if the write fails (disk full or file exists etc) ?

As the **write-buffer** command should handle write file failures sensibly, and we have enabled it to get input from the user, then the command will automatically handle the failure conditions. Within our macro then it is simply a case of detecting and handling the failure condition – we have already done this by handling the user input so no additions are required to the macro.

### 7. What happens if the current buffer displayed in more than one window ?

If the user has the current buffer displayed only once then the action of deleting the temporary buffer returns the current window to the previous buffer, as required. If however the buffer is displayed more than once then when the temporary buffer is deleted MicroEmacs will select a different non-displayed buffer to become the current window (unless this is the only buffer) and so the user will not be returned back to their original buffer.
This may be fixed by storing the initial buffer name and explicitly returning to it after the **delete-buffer**:

```
define-macro write-region
    copy-region
    set-variable #l0 $buffer-fname
    set-variable #l1 $buffer-bname
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0
    !force !force write-buffer @mna
    !if &not $status
        !force delete-buffer $buffer-bname
        find-buffer #l1
        !abort
    !endif
    delete-buffer $buffer-bname
```

```
    find-buffer #l1
!emacro
```

However, when there are two windows into the same buffer the command may inherit the other window's buffer location, i.e. if the current window is on line 100 and the other window display this buffer is at line 200 the above version will leave the user at line 200 and a different region. The commands **set-position**(2) and **goto-position**(2) allow the current position to be saved and later restored, the commands both take a single character which identifies the saved position, this allows multiple positions to be saved. Re-writing our macro to save and restore the buffer position we get:

```
define-macro write-region
    copy-region
    set-variable #l0 $buffer-fname
    set-position "P"
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0
    !force !force write-buffer @mna
    !if &not $status
        !force delete-buffer $buffer-bname
        goto-position "P"
        !abort
    !endif
    delete-buffer $buffer-bname
    goto-position "P"
!emacro
```

### 8. Does this fit in with other MicroEmacs commands and macros ?

Finally we should consider how to make the macro more like a built in command so that other users and macros can take advantage of this new feature. Firstly let us consider supporting the numerical argument which is passed to all commands and macros. Considering **write-buffer** the command permits an invocation from the command line of:

```
esc 0 C-x C-w
```

or from a macro:

```
0 write-buffer
```

The 0 argument disables validity checks and therefore forces the write. It would be reasonable to expect that a new command that writes a region to a file behaves in exactly the same way, where the commands have consistent behavior it makes it far easier for the user to guess how like commands operate. Unfortunately this behavior is absent from our current macro command. To support the numerical argument then the @? and @# variables are used to test for and retrieve the numerical argument.

```
define-macro write-region
    copy-region
    set-variable #l0 $buffer-fname
    set-position "P"
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0
    !if @?
        !force !force @# write-buffer @mna
    !else
        !force !force write-buffer @mna
    !endif
    !if &not $status
        !force delete-buffer $buffer-bname
        goto-position "P"
        !abort
    !endif
```

```
    delete-buffer $buffer-bname
    goto-position "P"
!emacro
```

**Note:** that the **!else** and **!endif** directive lines do not effect the setting of **$status** so this is safe, only the execution of commands effect this variable.

We can optimize the macro a little by noting that the default behavior of "**write-buffer**" is the same as "**1 write-buffer**" (this is not always the case), re-writing:

```
define-macro write-region
    copy-region
    set-variable #l0 $buffer-fname
    set-position "P"
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0
    !force !force @# write-buffer @mna
    !if &not $status
        !force delete-buffer $buffer-bname
        goto-position "P"
        !abort
    !endif
    delete-buffer $buffer-bname
    goto-position "P"
!emacro
```

After all of this work there are still some compatibility problems to be resolved for our macro to be a true command replacement. Considering our use of **write-buffer**, we have effectively have:

```
write-buffer "myfile"
```

The command gets the filename from the macro line and allow us to invoke the command in our own macros. So what about our new **write-region** macro ? It is not unreasonable to expect that some macro developer in the future will want to write the current region out to a file and if we write this one correctly they will be able to use this command directly. At the moment a command invocation of

```
write-region "myfile"
```

in another macro will not have the desired effect. The "`myfile`" argument is ignored and the user is always prompted for the file name. This can be solved by providing an optional parameter to be passed to the command using the /Hl @1 @1 4 variable as follows:

```
define-macro write-region
    copy-region
    set-variable #l0 $buffer-fname
    set-position "P"
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0
    !force set-variable #l1 @1
    !if $status
        !force !force @# write-buffer #l1 @mna
    !else
        !force !force @# write-buffer @mna
    !endif
    !if &not $status
        !force delete-buffer $buffer-bname
        goto-position "P"
        !abort
    !endif
    delete-buffer $buffer-bname
    goto-position "P"
!emacro
```

Setting the register variable `#l1` to the first argument passed to the command will fail if no argument was given, so we use a **!force** to stop the macro from quitting and change the **$status** of the **set-variable** command. If the assignment fails then we get the filename from the user as before, if it succeeds then the calling macro has provided the file name.
The **write-region** macro is now complete and provides both a command line and macro interface. It is important to note that there is a subtle difference between macros and built in commands even though their capabilities are the same. To execute **write-buffer** in a macro and get the user to supply a file name you must do:

```
write-buffer @mna
```

However for a macro, it is not itself a command and you cannot ask a macro to behave like a command. For a macro you must call it as:

```
write-region
```

The use of '`@mna`' is only valid for commands, this perhaps is a quirk of the language which should really be fixed in the future.

### Concluding

We started this tutorial with a unusable 5 line macro and now have a 20 line version which should be indistinguishable from a built in MicroEmacs command. The same command is available as part of the macro release and this document was written as part of the process of writing that macro. The only change to be made in making it part of the macro release is to change the **set-position** label from "`P`", which a user could use, to something like "`\x88`" which is much safer to use.
The final macro release version becomes:

```
define-macro write-region
    copy-region
    set-variable #l0 $buffer-fname
    set-position "\x88"
    !force delete-buffer "*write-region-tmp*"
    find-buffer "*write-region-tmp*"
    yank
    set-variable $buffer-fname #l0
    !force set-variable #l1 @1
    !if $status
        !force !force @# write-buffer #l1 @mna
    !else
        !force !force @# write-buffer @mna
    !endif
    !if &not $status
        !force delete-buffer $buffer-bname
        goto-position "\x88"
        !abort
    !endif
    delete-buffer $buffer-bname
    goto-position "\x88"
!emacro
```

It is perhaps surprising the number of steps it has taken to write something that appears to be simple, however in writing macros there are many re-occurring themes and once familiar with the concepts then then you will typically get them right first time. Creating a macro like this takes approximately half an hour.

We have to question whether it is worth writing these extensions. If you find yourself doing the same thing all the time then definitely yes, it will save time and you will have less aches and pains in your hand. Spending the extra time to write a good macro saves time in the long run, it also provides a better base on which more sophisticated macros may be based. MicroEmacs

has actually evolved like this. Hopefully this tutorial will encourage rather than frighten off budding macro writers.

## SEE ALSO

Macro Glossary
Macro Debugging - A Crash Course
MicroEmacs Macro Language at a Glance

Ref: macroWriting(4) File: m2nar010.2 Date: 2026/02/23

**FREQUENTLY ASKED QUESTIONS - Contact Information**

This document contains frequently asked questions submitted to JASSPA. Use the E-Mail reflector and associated logs, described in the Contact Information section, alternatively questions may be submitted to:

```
Email:support@jasspa.com
```

We cannot promise to resolve all questions, but will endeavor to answer most. We would also appreciate comments on how to improve the readability of the documentation or suggestions for improvements where you think the documentation is deficient.

[FAQ Index].

Ref: FAQs(0f) File: faq.2 Date: 2026/03/05

**QUESTION (00)**

New functionality; what is useful to me as an old MicroEmacs user ??

**ANSWER**

There are a lot of new features in this distribution. Assuming that you just want to use the editor (and have cottoned onto the fact that there are now scroll bars etc.) then the most frequent commands that we use are:

- **grep**(3) - May need to set up in *<user>.emf*.
- **diff**(3) - May need to set up in *<user>.emf*.
- **compile**(3) - May need to set up in *<user>.emf*.
- **clean**(3) - cleans a buffer, removing spaces etc.
- **restyle-buffer**(3) - Reformats 'C' + known languages.
- **spell-buffer**(3) - For documentation work, spells the buffer.
- **C-s** - **isearch-forward**(2) incremental search.
- **C-x u** or `C-_` - **undo**(2) undoes edits.
- **F10** - **file-browser**(3) allows the file system to be browsed

Other useful macros include:

- **tabs-to-spaces**(3) - Good for sorting out the mess made by Microsoft Developer Studio.
- **sort-lines**(2) - Two versions of this, allows marked lines to be sorted alphabetically.

Be wary of:

- **esc-o** - **fill-paragraph**(2). The default mode is an automatic mode which attempts to guess at the format required. It works most of the time. Also works in 'C'.

Most of the other new features are in the background, such as the macro processor, indentation control, color hilighting, indentation control, auto-saving etc.

[FAQ Index].

Ref: FAQ(00) - New functionality; what is useful to me as an old MicroEmacs user ?? File: faq.2 Date: 2026/03/05

**QUESTION (01)**

Languages; Are any foreign languages supported other than English ??

**ANSWER**

Unfortunately as we started with V3.8 as a base many years ago, we missed the distribution with foreign language extensions. We have not incorporated them into the release.

The May 1999 release improves the language support by supporting the ISO-Latin character sets.

We do have spelling dictionaries for French, Spanish, Portuguese and German. Other languages may be supported by transforming native **ispell**(1) dictionaries.

If there is enough interest in this release from people with foreign languages then we may consider including support for foreign language(s). However we would be very much reliant on external help for local testing and translation. We would be open to suggestions.

[FAQ Index].

Ref: FAQ(01) - Languages; Are any foreign languages supported other than English ?? File: faq.2 Date: 2026/03/05

**QUESTION (02)**

C++ is not default, C is - how do I change this ??

**ANSWER**

If your main programming language is C++, then you will require the `.def` and `.h` files to be loaded in C++ mode by default, rather than 'C'. To modify this hen the extension list and priority of the `cpp` major-mode needs adjusting.
Within your *<user>.emf*, over-ride the default extension list and priority by adding the line:

```
4 major-mode "cpp" ".cc .cpp .cxx .def .h .hh .hpp .hxx .rc"
```

This redefines the C++ major-mode extension list to include "`.h`" and "`.def`" and makes it highest priority.

[FAQ Index].

Ref: FAQ(02) - C++ is not default, C is - how do I change this ?? File: faq.2 Date: 2026/03/05

**QUESTION (03)**

GNU Emacs; are there any GNU Emacs bindings. ?

**ANSWER**

No not at the moment. The GNU Emacs bindings would be added as a compatibility file (meemacs.emf) in much the same way that the me3.8 bindings are added, see meme3_8.emf.

From the user-setup(3), the user would then ask for "gnu" compatibility.

We would welcome submissions for a gnu compatibility file, gnu.emf, to add to the release.

The **Meta** key (typically `Alt`) may be bound to key strokes, as opposed to the menu short-cut from the **user-setup**(3) as follows:

> **Tools** -> **User Setup** -> **General** -> **Alt Action: Main Menu Hot-keys** = N
> **Tools** -> **User Setup** -> **General** -> **Alt Action: Esc Prefix** = Y

[FAQ Index].

Ref: FAQ(03) - GNU Emacs; are there any GNU Emacs bindings. ? File: faq.2 Date: 2026/03/05

**QUESTION (04)**

Icons are not displayed correctly in Microsoft Windows environments !!

**ANSWER**

After installing on Microsoft platforms, the Icons in the Explorer window may not be showing correctly. To remedy the situation then the following steps may be taken.

**Windows ’95**

Try re-starting the system first. If the icons are still incorrect then re-start in Safe mode and delete the file:

```
c:\windows\ShellIconCache
```

Restart and the Icons should be correct.

**Window ’98**

Try re-starting the system first. If the Icons are still incorrect then re-start in Safe mode, this should re-generate the Icon cache. Restart windows.

**NT** Restart the system.

[FAQ Index].

Ref: FAQ(04) - Icons are not displayed correctly in Microsoft Windows environments !! File: faq.2 Date: 2026/03/05

**QUESTION (05)**

ipipes not working on Microsoft Windows network drives ?

**ANSWER**

We are aware of a problem with the ipipe commands with ’95 and ’98 (not sure about NT) when the current drive is a Novel network drive.

Although we have not been ably to fully characterize the problem, we know that:

- Old Novel Clients prior to 2.2 - Does not work
- Novel Intranetware Client 2.2 - Does not work.
- Novel Network Client 2.5 - Does work.
- Novel Client 3.01 - Does work.

Any other information in this area would be appreciated to fully characterize the problem.

To get around the problem then disable ipipes using **$system**(5). From within your <*user*>.*emf* knock off bit 0x800 from **$system**(5). This will enable regular pipes, which will work, albeit not in the background.

[FAQ Index].

Ref: FAQ(05) - ipipes not working on Microsoft Windows network drives ? File: faq.2 Date: 2026/03/05

**QUESTION (06)**

Language not supported - will it be ??

**ANSWER**

We only support the (programming) languages that we have come into contact with. If you are using a language that we are not supporting then you will need to write a new `hk<language>.emf` file. See Language Templates on how to map a new programming language. The list of currently supported file types is defined in Supported File Types.

Jasspa would appreciate any new templates that people define for standard file types so that we can add them to the distribution.

For Microsoft Windows, any associated ”me” icons types would also be appreciated.

[FAQ Index].

Ref: FAQ(06) - Language not supported - will it be ?? File: faq.2 Date: 2026/03/05

## QUESTION (07)

Language file is incomplete

## ANSWER

For a number of the (programming) language templates we have only provided a sub-set of the commands, this is typically because we only use a sub-set ourselves.

For a number of templates, there is no indent support (see **indent**(2) and Supported File Types).

Note that when extending the template then only standard words should be added. Words which are local extensions should be added to a `myXXX.emf`.

Jasspa would appreciate completed template definitions.

## SEE ALSO

FAQ06

[FAQ Index].

Ref: FAQ(07) - Language file is incomplete File: faq.2 Date: 2026/03/05

**QUESTION (08)**

Input locked up and not accepting keys; how do I unlock ?

**ANSWER**

This sometimes happens if a macro has been aborted badly. Typically a few ”Ctrl-G”s (see **abort-command**(2)) will terminate the macro and return control back to the caller.

[FAQ Index].

Ref: FAQ(08) - Input locked up and not accepting keys; how do I unlock ? File: faq.2 Date: 2026/03/05

**QUESTION (09)**

MicroEmacs Bindings; How do I get the original MicroEmacs bindings ?

**ANSWER**

From **user-setup**(3) set the Emulation to "`MicroEmacs v3.8`". On re-starting (or Current) the macro file `meme3_8.emf` is executed and the bindings loaded. This file should restore your familiar execution set.

[FAQ Index].

Ref: FAQ(09) - MicroEmacs Bindings; How do I get the original MicroEmacs bindings ? File: faq.2 Date: 2026/03/05

**QUESTION (10)**

Microsoft Windows Locks up after killing an ipipe.

**ANSWER**

This is a known problem for '95/'98 (not NT), on killing an ipipe. Sometimes the "Winoldapp" locks up, if this is the case use "Alt-Ctrl-Del" to bring up the "**Close Program**" dialogue, kill off the "`WinOldApp`" if it is not responding.

MicroEmacs will then come back. We are looking for ways around this problem at the moment. From the programming perspective Windows is just not as nice as UNIX - which just works !!

[FAQ Index].

Ref: FAQ(10) - Microsoft Windows Locks up after killing an ipipe. File: faq.2 Date: 2026/03/05

**QUESTION (11)**

Mouse support under Microsoft windows is strange !!

**ANSWER**

The mouse operation under Microsoft windows (and DOS) is biased towards a 3-button mouse operation (Logitech is ideal !!), operating in a similar way to UNIX. i.e. <select> operation gets text <Middle button> yanks text back.

This stems from the fact that we all came from UNIX backgrounds. We have had a number of comments about this already and do plan to address this issue.

**Note:** Those of you that have already had a little dip into the operation of the mouse will have probably worked out that the whole of the visible mouse/screen interaction is driven through macros, so this functionality is actually a macro change.

[FAQ Index].

Ref: FAQ(11) - Mouse support under Microsoft windows is strange !! File: faq.2 Date: 2026/03/05

**QUESTION (12)**

Scroll bars too narrow !!

**ANSWER**

You can change the width of the scroll bars to double width from **user-setup**(3) "Wide Scroll Bars". Alternatively, you may do this yourself from `<user>.emf` by:

```
set-variable $scroll-bar &bor $scroll-bar 1
```

See **$scroll-bar**(5).

Remember if you have enabled wide scroll bars, under windows, or X-Windows, you may want to change your start-up screen width to `82` characters rather than `80` - see **change-frame-width**(2).

[FAQ Index].

**QUESTION (13)**

Tab key; Why does the tab key not operate in some windows ??

**ANSWER**

In buffers with indentation information the tab key re-computes the indentation of the line. This behavior may be changed from the user-setup.

Refer to documentation for **$system**(5) and **user-setup**(3).

[FAQ Index].

Ref: FAQ(13) - Tab key; Why does the tab key not operate in some windows ?? File: faq.2 Date: 2026/03/05

**QUESTION (14)**

Termcap; On a color terminal why is there no color ??

**ANSWER**

MicroEmacs has to be enabled to show color by default. From **user-setup**(3) enable ”Termcap Color”. This will give you basic colors.

You may also try enabling ”With Bold” - this may increase the range of colors.

[FAQ Index].

Ref: FAQ(14) - Termcap; On a color terminal why is there no color ?? File: faq.2 Date: 2026/03/05

## QUESTION (15)

Termcap; Some of the keys do not work - how can I bind them ?

## ANSWER

In your user setup <user>.emf add the new keys. You have to be careful as to the environment and probably need to do something like the following:

```
; First check we are not an Xterm
!if &not $use-x
    ; Quick check on the terminal type. We probably need to
    ; distinguish between terminal types for different bindings
    !if &seq $TERM "myterm"
        translate-key "<from>" "<to>"
        ...
        translate-key "<from>" "<to>"
    !endif
!endif
```

See **translate-key**(2) for details of translating termcap keys. See **describe-key**(2) to help identify the key.

[FAQ Index].

**QUESTION (16)**

Timestamp; Format incorrect, how can I change to MMDDYY.hhmm ?

**ANSWER**

From within your <*user.emf*> set the time stamp default format i.e.

```
set-variable $timestamp "<%M%D%Y.%h%m>"
```

See **$timestamp**(5).

[FAQ Index].

Ref: FAQ(16) - Timestamp; Format incorrect, how can I change to MMDDYY.hhmm ? File: faq.2 Date: 2026/03/05

### QUESTION (17)

Windows; Component characters rendered incorrectly, how do I fix ?

### ANSWER

If some of the components of the windows are rendered incorrectly, typically caused by local variations of character sets, then new window component characters may be defined. See **$window-chars**(5) for details on how to define new character replacements.

[FAQ Index].

Ref: FAQ(17) - Windows; Component characters rendered incorrectly, how do I fix ? File: faq.2 Date: 2026/03/05

**QUESTION (18)**

Windows Autosave and Backup files; are these potentially a problem ?

**ANSWER**

For windows ’95 up until OEM service release 2, the OS could not distinguish the difference between the files:

```
.xxx
.xxx~
```

on a read, we have managed to find a work around for this, however we would advise that the 3 letter extension is adhered to for these releases. For releases of ’95 OEM service release 2 and greater, ’98 and NT we have not found a problem with any of the auto save and backup naming.

Obviously, the backup naming will depend on the native file system. For instance if your system administrator has not enabled long file names on your Novel server.

[FAQ Index].

Ref: FAQ(18) - Windows Autosave and Backup files; are these potentially a problem ? File: faq.2 Date: 2026/03/05

**QUESTION (19)**

Printing; Why in Windows does the output come out in a buffer ??

**ANSWER**

Use the **File** -> **Printer Setup** dialog and change the destination to the "**Direct to printer**".
[FAQ Index].

Ref: FAQ(19) - Printing; Why in Windows does the output come out in a buffer ?? File: faq.2 Date: 2026/03/05

**QUESTION (20)**

Printing; On Windows which font should I use ??

**ANSWER**

We suggest that "**Courier New**" is used as the print font. This scales well and supports the full character set. Problems have been reported with networked postscript printers when used in conjunction with fixed fonts.

[FAQ Index].

Ref: FAQ(20) - Printing; On Windows which font should I use ?? File: faq.2 Date: 2026/03/05

**QUESTION (21)**

Printing; My printer is not supported ?

**ANSWER**

We are in the process of providing native postscript generation - UNIX users can stream their output through ”a2ps” and then into their standard printer queues.

Windows, the support is already built in.

For DOS then you need to get your printer manual out and sort out how to map the printer codes onto fonts. The printer codes are added to ”`printer.erf`”. We have already provided support for the HP DeskJet printer (PCL), look at this printer definition for some help as to the type of information that you need to set up. It’s all a bit fiddley, but you do not get much choice if you want more than plain ASCII out.

[FAQ Index].

Ref: FAQ(21) - Printing; My printer is not supported ? File: faq.2 Date: 2026/03/05

**QUESTION (22)**

Alt key maps to the Menu, how do I change ?

A-f opens the main File menu instead of executing forward-word (esc f). How do I make the Alt key act like the Meta key all the time?

**ANSWER**

The **Meta** key (typically `Alt`) may be bound to key strokes, as opposed to the menu short-cut from the **user-setup**(3) as follows:

> **Tools** -> **User Setup** -> **General** -> **Alt Action: Main Menu Hot-keys** = N
> **Tools** -> **User Setup** -> **General** -> **Alt Action: Esc Prefix** = Y
>
> [FAQ Index].

Ref: FAQ(22) - Alt key maps to the Menu, how do I change ? File: faq.2 Date: 2026/03/05

**QUESTION (24)**

Windows - Where is app850.fon ?

**ANSWER**

"app850.fon" is the font file used for the DOS window under '95/'95/NT. You should find it in your c: hidden. If you search from the Explorer->Tools->Find "app850.fon" it should be found in the fonts directory. There is nothing to be done - the file exists and is in the correct location.

If you do not have this file then, you might have "appXXX.fon", or some other fixed font. You can locate the file that you want as follows:

Start->Settings->Control Panel->Fonts

Display the font list as 'details'. Within that list you should find a "MS-DOS CPXXX" entry. It will be a red font (if you are in monochrome then it will have a 'A' in the box rather than a 'Tt'). This is a fixed font and will be a good alternative to "app850.fon", you can also try the "Fixedsys" font file which has some weird name.

To be honest I do not know what Microsoft are currently shipping. Most of the Windows platforms that I have used have been upgrades or been abused by so many people you never know what is original !!

We would be interested in any details of other fixed fonts, which support the full OEM character set that are better alternatives to the DOS ones.

[FAQ Index].

Ref: FAQ(24) - Windows - Where is app850.fon ? File: faq.2 Date: 2026/03/05

## QUESTION (25)

Time; mode line is showing the date in DD/MM/YY format how do I change ?

## ANSWER

From within your user setup, over-ride the default mode line setting with the modifications you require. i.e. to change the date format to MM/DD/YY use:

```
set-variable $mode-line "%s%r%u%k %b %l - %h:%m %M/%D/%Y (%e) - (%f) "
```

## SEE ALSO

**$mode-line**(5).

[FAQ Index].

Ref: FAQ(25) - Time; mode line is showing the date in DD/MM/YY format how do I change ? File: faq.2 Date: 2026/03/05

**QUESTION (26)**

C or C++ indentation and effects; how can I turn off ?

**ANSWER**

The C mode indentation is supposed to make editing 'C' easier, by forcing the user to follow a preset editing convection. The layout is pretty standard, following a 4 space indent, writing either K&R or standard 'Pascal' type layout, with braces aligning vertically.

The problem most new users have is the inability of the tab key to function, or more simply do not want to be 'forced' to write in a particular style (GNU writers will probably not like this either - conversely they will be using GNU emacs !!). However the constrained layout can be configured to create most styles and does help in a project situation, whereby most of the authored code roughly adheres to the same sort of layout conventions. For C++ users then edit "`hkcpp.emf`" rather than "`hkc.emf`".

**To turn off all automatic 'C' layout**

To disable ALL automatic layout in C and C++ files then from the menu

```
help -> Buffer Setup -> Setup Auto Indent
```

Or from the command line **esc x buffer-setup** and disable through the dialogue.

Restart the editor.

**To re-enable the <tab> key**

To retain the 'C' layout aid, but re-enable the tab key operation then disable the **Tab To Indent** option in **user-setup**(3) i.e.

**Tools** -> **User Setup** -> **General** -> **Tab To Indent**

This enables the use of the TAB key in all column positions with the exception of column 0. A <tab> in Column 0 will still enable the automatic line re-formatting.

If you want real <Tabs> then disable the **tab** mode using **user-setup**(3) i.e.

**Tools** -> **User Setup** -> **General** -> **Tab**

**To change the 'C' Indentation Layout**

The 'C' layout indentation is controlled from the **indent**(2) command (see `hkc.emf`), the indentation level is controlled by the variables **$buffer-indent-width**(5) and **$buffer-indent**(5) these may be changed in the **Buffer Setup** dialog.

[FAQ Index].

Ref: FAQ(26) - C or C++ indentation and effects; how can I turn off ? File: faq.2 Date: 2026/03/05

## QUESTION (27)

fill-paragraph function does not fill ??
I can't seem to get the fill-paragraph function to fill the following paragraph:

```
This is a very
poorly formed paragraph
which refuses to fill
properly!
```

## ANSWER

The default justification mode is Auto which tries to work out the mode required for each paragraph. Its fairly smart at maintaining a documents indentation, e.g. consider the example right hand justified:

```
                This is a very
          poorly formed paragraph
            which refuses to fill
                        properly!
```

It will maintain this indentation. The problem comes when the detected form is not the required form as in the example. The detected paragraph justification to be used is "none" because the lines are short. There are 2 ways to solve this problem:

- You can change the **$fill-mode**(5) to left or both (in fhook-doc mode use C-c l or C-c b) and then use fill-paragraph as normal.
- Manually concatenate the first few lines into one to create a longer first line and then use the fill-paragraph a normal, i.e. change the paragraph to:

  ```
  This is a very poorly formed paragraph which refuses to fill properly !
  ```

  and then fill. This works because the longer line will lead to a different assessment of what's required.

[FAQ Index].

Ref: FAQ(27) - fill-paragraph function does not fill ?? File: faq.2 Date: 2026/03/05

**QUESTION (28)**

Key modifier which acts as the ESC key; what is it ?

What is the modifier key which acts as the ESC key ? Having to type ESCAPE and then f to move one word forward is very boring.

With Gnu Emacs (on Unix systems), there is a ”meta” modifier key which is a shortcut for pressing ESCAPE followed by the command key. The ”meta” key should be the ”Alt” key.

**ANSWER**

The ”meta” key is the ”Alt” key. But ’F’ is the Main menu hot-key for the ’File’ sub-menu so by default ’A-f’ will open the File sub-menu. This can be disabled by clearing bit 0x2000 in the $system variable. This option can now be set using user-setup (Alt -> Main Menu).

[FAQ Index].

Ref: FAQ(28) - Key modifier which acts as the ESC key; what is it ? File: faq.2 Date: 2026/03/05

**QUESTION (29)**

find-file start location; where is it ?

**ANSWER**

The **find-file**(2) start location is defined as follows:

- `*scratch*` is current buffer; the current working directory.
- `file` is current buffer; the directory location containing *file*.

Running under Microsoft Windows or UNIX, using an icon launch, then it may become frustrating that the start location is always `C:\Program Files\JASSPA\MicroEmacs` (Microsoft windows) or `/usr/local/bin` (UNIX) this is simply resolved by starting the executable with the **-c** option, as defined by **me**(1). The **-c** option starts the editor with the last editing session, this is typically where a user will want to commence an editing session.

If the **-c** approach is not acceptable, then it is worth defining the environment variable `$HOME` within the start up script, or in the users environment. Using **find-file** with tilde ( ) implies that the directory start path is `$HOME`.

[FAQ Index].

Ref: FAQ(29) - find-file start location; where is it ? File: faq.2 Date: 2026/03/05

**QUESTION (30)**

Re-using a MicroEmacs session; how to ??

**ANSWER**

A MicroEmacs editing session may be re-used, such that the current editor is prompted to load a new file externally. This is typically invoked from a short-cut launch from a file manager i.e. **Explorer**(1), **Tkdesk**(1) etc.

In order to facilitate the re-use of the session, then **me**(1) is invoked with the **-o** option, this locates the active editor session and passes the file load request. If an existing session does not exist then a new session is started.

In order for this mechanism to operate, then the Client-Server Interface must be enabled from the **user-setup**(3) i.e.

> [**Tools** -> **User Setup** -> **Platform** -> **Client Server** = `Y`]
>
> [FAQ Index].

Ref: FAQ(30) - Re-using a MicroEmacs session; how to ?? File: faq.2 Date: 2026/03/05

**QUESTION (31)**

Drag and Drop; is it supported ??

**ANSWER**

MicroEmacs supports Microsoft and UNIX XDND (Gnome/KDE) *drag-and-drop* interaction. Multiple files and directories may be dragged from Microsoft Explorer, Nautilus (or other application) and dropped into a buffer window. The destination buffer window is the window in which the dropped file(s) are displayed.

**Note** if the user is currently on the command line, then the command line operation is aborted in order to facilitate the *dropped* files.

[FAQ Index].

Ref: FAQ(31) - Drag and Drop; is it supported ?? File: faq.2 Date: 2026/03/05

**QUESTION (32)**

Cut and Paste to/from other applications; is it supported ??

**ANSWER**

MicroEmacs supports *cut and paste* operations on all platforms.

To copy a region from MicroEmacs to another application

Select a region (with the mouse or keys) - there is no need to invoke a copy operation. All selected text is immediately available to other applications.

Move to the new application and paste, as dictated by the platform.

To copy a region from another application to MicroEmacs

Select the region in the application into the clipboard, as dictated by the platform.

Move to MicroEmacs, position the cursor and **yank**(2) (`C-y` or typically the middle mouse button) the clipboard text.

[FAQ Index].

Ref: FAQ(32) - Cut and Paste to/from other applications; is it supported ?? File: faq.2 Date: 2026/03/05

**QUESTION (33)**

Fonts; how can I change the font ??

**ANSWER**

The currently selected font may be modified from the **user-setup**(3).

**Tools** -> **User Setup** -> **Platform**

The font selection depends upon the platform, in all cases a fixed font should be selected, otherwise rendering anomalies will result.

If you are running on Microsoft platforms ensure that the **OEM/ANSI** flag matches the settings of the **Display Font Set** entry.

[FAQ Index].

Ref: FAQ(33) - Fonts; how can I change the font ?? File: faq.2 Date: 2026/03/05

**QUESTION (34)**

Colors; how can I change screen colors ??

**ANSWER**

The screen colors are selected from the **user-setup**(3).

**Tools** -> **User Setup** -> **Platform** -> **Color Scheme**

The default setting is *White on Black*, the *Black on Cream* is the most popular setting.
[FAQ Index].

Ref: FAQ(34) - Colors; how can I change screen colors ?? File: faq.2 Date: 2026/03/05

**QUESTION (35)**

File Types; how do I interchange between UNIX, Windows and DOS files ??

**ANSWER**

MicroEmacs facilitates the editing of the standard file types on all platforms. All files retain their line ending type through edits. i.e. if a DOS file is edited on a UNIX system, the file is still written as a DOS file. When new files are created, they are created with the standard attributes of the host O/S.

The line ending of the file may be modified from the menu

**file** -> **attributes**

This brings up a dialog that allows the file type and attributes to be modified.

Note that the only ending that is NOT preserved are files whose lines end in <CR>'s only. The line format is correctly interpreted on reading, but is not retained on the write.

[FAQ Index].

Ref: FAQ(35) - File Types; how do I interchange between UNIX, Windows and DOS files ?? File: faq.2 Date: 2026/03/05

**QUESTION (36)**

Non-English Languages; What font should I select ??

**ANSWER**

MicroEmacs has only been tested with Western Lanuguages only. Within the Microsoft Windows environment an ANSI type font should be selected, assuming of course that the characters required are in the ISO-Latin character set. UNIX typically supports ISO-Latin character sets.

[FAQ Index].

Ref: FAQ(36) - Non-English Languages; What font should I select ?? File: faq.2 Date: 2026/03/05

**QUESTION (37)**

MicroEmacs ’99; How do I up-grade from MicroEmacs’98 ??

**ANSWER**

Backup your current version!

Follow the MicroEmacs’99 installation procedure to install and get MicroEmacs’99 running.

Due to the great improvement to **user-setup**(3) it is advised that the user creates a new setup using **user-setup** and then migrates required macro code changes from the old release into the new.

[FAQ Index].

Ref: FAQ(37) - MicroEmacs ’99; How do I up-grade from MicroEmacs’98 ?? File: faq.2 Date: 2026/03/05

## QUESTION (38)

Some keys on my foreign keyboard do not work properly, how do I get them working ??

## ANSWER

The most common problem are with foreign keyboards where the <AltGr> key is used to generate some characters in a similar fashion to the <Shift> key. For example, on a Belgian keyboard the '9' key produces a '{' character when the <AltGr> key is also pressed.

The quickest and best solution is to use the **Keyboard** setup on the Start-Up 0f of **user-setup**(3). This however may not provide the solution as not all keyboards are currently supported. If you are using an unsupported keyboard please send configuration information back to JASSPA for inclusion in the next release. The keyboard configuration information is stored in the macro file `keyboard.emf`.

If **user-setup** does not currently support your keyboard, or you wish to remap some keys, then the command **translate-key**(2) should be used. **translate-key** remaps generated key stroke(s) into another key at a low level so the mapping is supported in all areas. If a macro and key binding were used instead, while they would work in the main text windows, they would not work in the message line. See help on **translate-key** for more information.

Note that some <AltGr> keys can produce 2 keys, for example on a Belgian keyboard '<AltGr>-9' produces the key 'A-C-9' first, immediately followed by 'A-C-{'. This is an unfortunate side effect of windows, it is better to have two keys rather than none. But this does add confusion to the problem! Again, see **translate-key** for more information.

[FAQ Index].

Ref: FAQ(38) - Some keys on my foreign keyboard do not work properly, how do I get them working ?? File: faq.2 Date: 2026/03/05

## QUESTION (39)

Tabs; How to change the tab width ??

## ANSWER

**$tab-width**(5) controls the size of the displayed tab character (number of spaces) and the simulated tab character size, where the user entered tab character is replaced by a number of space characters. The latter is only used when **tab**(2m) mode is enabled (it is typically enabled by default).
To change the tab character width then the **set-variable**(2) command is used:

```
esc x set-variable
```

You will then be prompted for the remaining arguments. <TAB> is the completion so:

```
esc x set-v<TAB>
$tab-width<TAB>
2
```

If this is the setting that you always want to use then it is easier if you put this in your <user.emf> as:

```
set-variable $tab-width 2
```

then whenever you start a new session you will always have the $tab-width defined as you want it.
If you are reading in a file with TAB's embedded then you can convert all of the <TAB>'s to spaces using **tabs-to-spaces**(3):

```
esc x tabs-to-spaces
```

[FAQ Index].

Ref: FAQ(39) - Tabs; How to change the tab width ?? File: faq.2 Date: 2026/03/05

**QUESTION (40)**

Windows/DOS; Where do I get grep/diff etc. ??

**ANSWER**

For windows and DOS users the UNIX tools may be obtained from:

```
ftp://ftp.cdrom.com/pub/garbo/garbo_pc/unix/uxutl23a.zip (238 KB)
ftp://ftp.cdrom.com/pub/garbo/garbo_pc/unix/uxutl23b.zip (227 KB)
ftp://ftp.cdrom.com/pub/garbo/garbo_pc/unix/uxutl23c.zip (221 KB)
ftp://ftp.cdrom.com/pub/garbo/garbo_pc/unix/uxutl23d.zip (160 KB)
```

comments for this at:

```
http://www.geocities.com/SiliconValley/Lakes/1401/softlib1.htm
```

One awk-port; the Gnuish project has 16 bit and 32 bit versions of **gawk**(1) in:

```
ftp://mirrors.aol.com/pub/simtelnet/gnu/gnuish/gawk303x.zip (1997, 495K)
```

Acknowledgment: **DG** - 99/07/02

[FAQ Index].

Ref: FAQ(40) - Windows/DOS; Where do I get grep/diff etc. ?? File: faq.2 Date: 2026/03/05

**QUESTION (41)**

Home/End Keys; How do I change the default bindings ??

**ANSWER**

Some users prefer the HOME and END keys to map to the beginning and end of the line, rather than beginning/end of the buffer, respectively. Within the *<user>.emf* the following global bindings may be applied to re-assign the key mappings:

```
global-bind-key "beginning-of-buffer" "C-home"
global-bind-key "end-of-buffer"       "C-end"
global-bind-key "end-of-line"         "end"
global-bind-key "beginning-of-line"   "home"
```

Acknowledgment: **DG** - 99/07/02

[FAQ Index].

Ref: FAQ(41) - Home/End Keys; How do I change the default bindings ?? File: faq.2 Date: 2026/03/05

## QUESTION (42)

tags; How do I generate a MicroEmacs compatible tags file ??

## ANSWER

A **tags** file is used by the **find-tag**(2) command. This is used to hypertext to the *tagged* definition or variable. The standard **ctags**(1) format is used by MicroEmacs. The **tags** file itself may be generated by MicroEmacs from the menu (*Tools->XX Tools->Create Tags File*). Alternatively a **tags** file may be generated by the **ctags**(1) utility. This is typically standard on UNIX platforms. For Windows and DOS platforms then the **Exuberant Ctags** is recommended, this is available from:

```
https://ctags.sourceforge.net/
```

A MicroEmacs compatible tags file may be generated using the command line "`ctags -N --format=1 .`" cataloging the current directory. To generate **tags** for a directory tree then use "`ctags -NR --format=1 .`". Refer to the **Exuberant Ctags** documentation for a more detailed description of the utility.

The user variable **%tag-option**(5) may be used to enable **find-tag**(2) to locate a recursivelly generated **tags** file from a parent directory.

[FAQ Index].

Ref: FAQ(42) - tags; How do I generate a MicroEmacs compatible tags file ?? File: faq.2 Date: 2026/03/05

**QUESTION (43)**

ne; how can I enable line wrapping in NE ??

**ANSWER**

The line wrapping feature is controlled by the wrap buffer-mode in ME but this feature is omitted in NE. To get the size of NE down many features have to be disabled including the wrap mode and fill-paragraph.

To enable then you have to build your own version of NE with this enabled by changing the 2nd MEOPT_WORDPRO settings in "emain.h" to 1. Alternatively rename your ne.emf to me.emf and use 'me' instead of ne.

Once it is built and you want to edit a text file then you need to enable WRAP and INDENT mode in the buffer for it to behave correctly. WRAP does the wrap to the next line and INDENT maintains the left indentation. fill-paragraph is present.

[FAQ Index].

Ref: FAQ(43) - ne; how can I enable line wrapping in NE ?? File: faq.2 Date: 2026/03/05