Emacs Reference

Fred Morcos

Description and History

- Emacs is an extensible, customizable, self-documenting, real-time display text editor.
- Originally, a set of editing macros for the TECO editor.
- GNU Emacs was developed by R. Stallman and G. Steele.
- Technically, GNU Emacs is an `elisp` interpreter with text editing extensions.
- GNU Emacs contains thousands of commands and allows the user to combine them into `elisp` procedures (called macros) to automate work.
- Commands are, themselves, `elisp` procedures and the GNU Emacs configuration *init* file is an `elisp` program.

Technical Description

Extensible

The user can define, undefine and redefine commands as well as re-use available commands into macros.

Customizable

The user can change properties of emacs elements including the key-bindings and the display.

Self-documenting

All defined commands and macros get automatic preliminary documentation (bound keys and parameters).

Why Emacs?

- Easily programmable
- Kill ring
- Rectangular editing
- Registers
- Command repetition
- Macros and Lisp procedures
- Modes
- Init file
- Remote files

  ...

- Shell, Terminal Emulator, IRC, Email, News, File Manager, Process Manager, Project Manager, Tetris, Doctor, ...

Terminology & Structure

Buffer	a container for data (text, completions, ...)
Point	the current position in the buffer (cursor)
Window	a visual container for a buffer
Frame	a visual container for one or more windows
Kill	the equivalent of cutting
Yank	the equivalent of copying
Kill Ring	a circular clipboard (very handy)
Mark	the coordinates of a selection
Region	the text inside a mark
Modeline	a status line
Minibuffer	a small buffer for commands and arguments
Mode	the current type of data being edited

Modeline

The Modeline is used to report different pieces of information.

```
-cs:ch-fr  buf      pos line   (major minor)------
```

cs	character set (or coding system)
:	newline mode
ch	file modification status
- or @	file is local or remote
fr	frame name on text terminals
buf	buffer/file name
pos	point/cursor position
line	current line and column number
modes	currently loaded major and minor modes

Keybind Terms & Minibuffer

Emacs has some different naming of command/control keys (coming from the old days).

M	**M**eta key, Alt or ESC
C	**C**ontrol key
S	**S**hift key
M-a	press *a* while holding the meta key
C-M-a	press *a* while holding the control and meta keys
C-a b	press *a* while holding the control key then press *b*
C-a C-b	press *a* and *b* both while holding the control key
C-a M-b	press *a* with control then *b* with meta

The Minibuffer is used to input commands, arguments and setting different modes (which some of are bound to keys or menu actions).

M-x	execute a command by name

Basic Usage

Loading

$ emacs	opens at the scratch buffer
$ emacs [*file*1] [*file*2] ...	opens in split-window mode
C-x C-f <*filename*>	loads a new buffer from filename

Saving

C-x C-s	saves the current buffer
C-x C-w <*filename*>	saves the current buffer as filename

Other

C-x C-c	quits emacs
C-g C-g	cancels key-sequence

Windows

Windows are used to display buffer contents. Windows can be split and resized inside an Emacs frame. The minibuffer has its own window.

C-x 2	open a window vertically
C-x 3	open a window horizontally
C-x 0	close the current window
C-x 1	close all other windows
ESC ESC ESC	close all other windows
C-x o	move to another window
C-x ˆ	increase window size vertically
C-x }	increase window size horizontally
C-x {	decrease window size horizontally
C-x −	shrink window to buffer size
C-x +	make all windows equal in size

Navigation & Movement

Word

C-←	moves one word backwards
C-→	moves one word forward

Line

C-a	goes to beginning of the line (Home)
C-e	goes to end of the line (End)
M-a	goes to beginning of the sentence
M-e	goes to end of the sentence
M-g g	goes to given line by number

Navigation, Movement & co. [contd.]

Paragraph

C-↑	moves up one paragraph
C-↓	moves down one paragraph

Buffer

C-x [	moves to the beginning of the buffer
C-x]	moves to the end of the buffer
C-l	moves the buffer to position point
C-x b	switches to another buffer by name
C-x ←→	switches to previous or next buffer
C-x k	kills the current buffer

Editing

Character

C-d	deletes next character
C-t	transposes current character with previous one
C-q TAB	inserts a TAB verbatim

Word

M-d	kills the next word
M-- M-d	kills to the previous word
M-/	expands current word
M-/ SPC M-/	expands current word and grabs next one
M-C-/	completes current word
M-t	transposes current word with next one
M-- M-t	transposes current word with previous one
M-c	capitalizes next word
M-- M-c	capitalizes previous word

Editing [contd.]

Word [contd.]

M-l	lowercase next word
M-- M-l	lowercase previous word
M-u	uppercase next word
M-- M-u	uppercase previous word

Line

M-o M-s	center a line
C-S-BACKSPC	kills current line
C-k	kills from point to EOL
M-- C-k	kills to the previous line
C-a C-k	goes to BOL then kills to EOL
TAB	indents line depending on current mode
C-o	breaks line before/after point
C-x C-t	transpose current line with previous one
M-0 C-x C-t	transpose current line with one at mark
`delete-matching-lines`	delete lines matching a regexp

Prefix Arguments & Command Repetition

Passing numerical arguments to commands can alter their behavior (e.g., repetition or inversion). Passing an argument can be done before **M-x** or key bindings.

M-n com	run *com* with prefix arg *n*
M-- com	run *com* with negative prefix arg
C-u n com	run *com* with prefix arg *n*
C-u com	run *com* with prefix arg 4
C-u C-u com	run *com* with prefix arg 8
C-u	also terminates the prefix argument
C-x [z]+	repeats the previous z^{th} command with args
C-x ESC ESC	repeats last command that uses the minibuffer

Examples:

C-u 5 0 C-k	will kill 50 lines
C-u 5 C-u 0	will insert 5 zeros

Search & Replace

Incremental search

C-s <*keyword*>	search forward
C-r <*keyword*>	search backward

Non-incremental search

C-s RET <*keyword*>	search forward
C-r RET <*keyword*>	search backward

Regular Expression Search

C-M-s <*expression*>	search regexp forward
C-M-r <*expression*>	search regexp backward

Notes:

- **ESC** will cancel and go to original point. **RET** will end at the current point.
- Searches are case-insensitive unless an uppercase letter is found in the search string.
- While searching, **C-w** will increment the search term with the current word.

Search & Replace [contd.]

Unconditional Replace

`replace-string`	replaces a string with another
`replace-regexp`	replaces a regexp match with a string

Query Replace (Conditional Replace)

M-%	conditional replace of a string
`query-replace-regexp`	conditional replace of a regexp

Notes:

- Replaces are case-insensitive unless an uppercase letter is found in the match string.
- Replaces work from position to end of buffer unless a mark is active.
- After a replace, the position will be at the end of the last match. **C-u C-SPC** to go back to the position before the replace started.
- During a query replace, **SPC** will apply a replace and **DEL** will skip.

Undo & Redo

In Emacs, there is no special redo function. Instead, there is only an undo function and redo can be achieved by undo-ing an undo.

C-/	undo a single change
C-g	break the chain of undos

Notes:

- If a mark is active, undo will only affect the marked region.

Regions

Emacs uses the mark and point to denote a region (i.e., a selection).

C-SPC ←↑↓→	creates a region (marks area)
C-x C-x	swaps mark and point in a region
C-u C-x C-x	swaps mark and point without region
C-x h	marks the entire document
M-h	marks the paragraph around
TAB	indents the current region
M-@	incrementally mark next word

Kill, Yank and Paste Regions

C-w	kills the currently selected region
M-w	copies the currently selected region
C-y (M-y)*	yanks and cycles from the kill ring

Notes:

- *Yank* is the copy command in `vi` and the paste command in `emacs`.

Rectangles

Rectangles are marked regions between the columns of the point and mark. Applying operations on rectangles is some sort of vertical editing.

C-x r k	**k**ill the rectangle
C-x r d	**d**elete the rectangle
C-x r y	**y**ank the last killed rectangle
C-x r o	push text to fill rectangle with spaces (**o**pen)
C-x r c	replace rectangle text with spaces (**c**lear)
C-x r t	replace each line in rectangle with **t**ext

Registers

Registers are places where you can store anything: text, position, rectangle, configuration, filename, ... Registers are named: *a*, *A* and *3* are three different registers.

`view-register`	view the contents of a register
C-x r SPC r	record current point position in register *r*
C-x r j r	jump to position in register *r*
C-x r s r	save region to register *r*
C-u C-x r s r	kill region to register *r*
C-x r i r	insert text from register *r*
`append-to-register`	append region to register
`prepend-to-register`	prepend region to register
C-x r r r	save rectangle into register *r*

Bookmarks

Bookmarks are like registers but persistent. They can also be named.

C-x r m foo	bookmark file and point to *foo*
C-x r b foo	jump to bookmark called *foo*
C-x r l	list all bookmarks
`bookmark-save`	save all bookmarks
`bookmark-delete`	delete a bookmark

Macros

Macros are user defined commands using the Emacs command language. A user can "record" and "replay" a macro, useful for simple repeatable tasks. More complex tasks (e.g., conditions, loops) must be implemented in `elisp`. Macros are recorded in a macro ring.

▶ **F3**	start macro definition or insert counter
▶ **C-x (**	start macro definition only
▶ **F4**	end macro definition **or** call macro
▶ **C-x e**	end macro definition **and** call macro
▶ **C-x)**	end macro definition only
▶ **C-u C-u F3**	append commands to last macro
▶ **C-u F3**	re-run last macro then append commands to it
▶ **C-x C-k r**	run macro on region
▶ **C-x C-k C-n**	rotate to select the next macro in the ring
▶ **C-x C-k C-p**	rotate to select the previous macro in the ring

Macros [contd.]

Every macro definition can have a counter to insert into the buffer.

F3	inside a macro definition, inserts the counter
C-x C-k C-i	outside a macro definition, inserts the counter
C-x C-k C-c	set the macro counter value
C-x C-k C-f	set the macro counter format

Macros can be named and saved.

C-x C-k n	name the most recently defined macro
C-x C-k b	keybind the most recently defined macro
`insert-kbd-macro`	insert macro into buffer as `elisp` code

Notes:

- It is best to use the reserved key bindings **C-x C-k [a-zA-Z0-9]** as to not cause problems with other bindings. **C-x C-k b 4** will define **C-x C-k 4** as a binding.

Alignment

Emacs contains alignment commands. Sometimes those depend on the current mode (i.e., the language).

`align`	aligns depending on mode
`align-regexp`	aligns using a regular expression
C-u `align-regexp`	aligns using a regular expression helper
`align-current`	aligns current paragraph depending on mode

Sorting

When sorting, prefixing with **C-u** sorts in descending order.

`sort-lines`	sort region by lines
`sort-paragraphs`	sort region by paragraphs
C-u **n** `sort-fields`	sort lines in region by $\underline{n}^{th}$ field
C-u -**n** `sort-fields`	sort lines in region by $\underline{n}^{th}$ field from right
`sort-numeric-fields`	interpret field as number and not text
`sort-columns`	sort by column specified by marked region
`reverse-region`	reverses current region

Tables

Emacs has support for creating and editing text-based tables by keeping track of their properties (e.g., position, size) in the buffer. When a buffer is saved to file, those properties are lost.

`table-insert`	interactively insert a table into buffer
`table-recognize`	detect properties of all tables in the buffer
`table-unrecognize`	remove special table properties
`table-recognize-region`	detect properties of tables in region
`table-unrecognize-region`	forget properties of tables in region
`table-recognize-table`	detect properties of table at point
`table-unrecognize-table`	remove properties of table at point

Tables [contd.]

Cell resizing:

► **C-u <u>n</u> C->**	widen cell at point by <u>*n*</u> characters
► **C-u <u>n</u> C-<**	narrow cell at point by <u>*n*</u> characters
► **C-u <u>n</u> C-}**	heighten cell at point by <u>*n*</u> lines
► **C-u <u>n</u> C-{**	shorten cell at point by <u>*n*</u> lines

Cell movement:

► **TAB**	move to cell on right
► **S-TAB**	move to cell on left

Cell merging/splitting:

► **C-c C-c ***	interactively merge two cells
► **C-|**	split cell horizontally
► **C- -**	split cell vertically

Tables [contd.]

Rows & Columns:

C-u n `table-insert-row`	insert *n* rows
C-u n `table-delete-row`	delete *n* rows
C-u n `table-insert-column`	insert *n* columns
C-u n `table-delete-column`	delete *n* columns

Other:

C-:	interactively justify cell, column or row text
C-!	toggle table fixed width mode
`table-generate-source`	table code in Latex, HTML or Cals

Goodies

▶ **C-x TAB**	force an indentation on a region
▶ **M-q**	indents and breaks paragraph into multiple lines
▶ **M-;**	comments a region or adds comment to line
▶ **M-(**	inserts a new lisp function (parenthesis)
▶ **M-)**	checks balanced parenthesis and opens line
▶ **C-x i**	inserts file contents at point position
▶ **M-!**	run shell command
▶ **C-u M-!**	run shell command and insert output
▶ **M-|**	run shell command on marked region
▶ **C-u M-|**	run shell command on and replace marked region

Goodies [contd.]

`insert-buffer`	inserts buffer contents at point position
`copy-to-buffer`	copy region content to a buffer
`kill-some-buffers`	interactively kill buffers
`electric-buffer-list`	interactive buffer list
`toggle-truncate-lines`	do not split lines over visual lines
`visual-line-mode`	split lines by words
`viper-mode`	vi compatibility

Setting Key Bindings

Emacs has a global keymap which maps between keys and commands. Each major mode can define, redefine or undefine its own key bindings, creating a local keymap (e.g., c-mode). Each minor mode can do so too (e.g., flymake). Each portion of text can also do so (e.g., tables).

Description of some prefix keys:

`C-x`	command prefix key
`M-`	command prefix key
`C-c`	mode specific prefix key
`C-h`	help prefix key

Commands to bind keys:

`global-set-key` **k** **com**	bind key *k* to *com* globally
`local-set-key` **k** **com**	bind key *k* to *com* locally (major mode)

Notes:

- Menu and mouse key bindings can also be set.

Storing Key Bindings

In the Emacs *init* file, you can either set global key bindings or “hook” local key bindings to mode hooks (i.e., callbacks, slots).

`global-set-key`	adds a binding to the global map
`local-set-key`	adds a binding to the local (major mode) map
`kbd`	converts a string to a key sequence

Examples:

Global key binding:

- `(global-set-key (kbd "C-c d") 'duplicate-line)`
- `(global-set-key (kbd "C-c d") (kbd "C-a C-@ C-e M-w RET C-y"))`
- `(global-set-key (kbd "C-c d") "\C-a\C- \C-n\M-w\C-y")`

Local key binding:

```
(add-hook 'LaTeX-mode-hook
    (lambda () (local-set-key (kbd "C-c n")
                              'forward-paragraph)))
```

The Emacs Help System

Emacs has a handy help system for quick lookup of various features, topics, functions and key bindings.

C-h ?	show the help system shortcuts
C-h <u>key</u>	run help system with shortcut <u>*key*</u>

Some useful help keys:

C-h a	show apropos page about a keyword
C-h k <u>keybind</u>	show help for function bound to <u>*keybind*</u>
C-h b	show all key bindings
C-h f	show documentation for function
C-h m	show documentation for current modes
C-h v	show documentation for variable
C-h w	show what keys a command is bound to
C-h t	start the emacs tutorial
<u>prefix</u> C-h	show all key binds starting with <u>*prefix*</u>

Modes

Modes are used to define the types of data being edited in buffers, or what Emacs calls "the language".

Every buffer has exactly one major mode which defines its language (C, Java, English, IRC, ...) and provides basic elements like syntax highlighting and (re)defines functions and their key bindings for relevant actions (e.g., comments).

Each buffer can have zero or more minor modes enabled which provide non-specific functionality such as spell checking or line wrapping (i.e., mode independent).

`ediff`

`ediff` (emacs-diff) is a mode for Emacs where you can view and merge difference between two or three buffers.

`ediff`	select files to view their differences
`ediff-buffers`	select buffers to view their differences

In `ediff-mode`:

|	switch between horizontal/vertical view
?	view `ediff` help
p	view previous difference
n	view next difference
a	set buffer b's area to what's in a's
b	set buffer a's area to what's in b's
q	quit `ediff` session

Spell Checking

M-$	check spelling at word or active region
`ispell`	check active region or entire buffer
i	accept word and insert into personal dictionary
?	show `ispell` help
RET	end current `ispell` session
ESC TAB	complete current word from dictionary
`flyspell-mode`	enable “on the fly” spell checking
`flyspell-prog-mode`	spell check comments and strings

References

- Emacs Wiki: `http://emacswiki.org/`
- Emacs Docs: `https://www.gnu.org/software/emacs/manual/html_node/emacs/index.html`
- Emacs Help