

GNU Emacs Reference Card

(for version 27)

Starting Emacs

To enter GNU Emacs 27, just type its name: `emacs`

Leaving Emacs

| | |
|---------------------------------------|----------------------|
| suspend Emacs (or iconify it under X) | <code>C-z</code> |
| exit Emacs permanently | <code>C-x C-c</code> |

Files

| | |
|---|----------------------|
| read a file into Emacs | <code>C-x C-f</code> |
| save a file back to disk | <code>C-x C-s</code> |
| save all files | <code>C-x s</code> |
| insert contents of another file into this buffer | <code>C-x i</code> |
| replace this file with the file you really want | <code>C-x C-v</code> |
| write buffer to a specified file | <code>C-x C-w</code> |
| toggle read-only status of buffer | <code>C-x C-q</code> |

Getting Help

The help system is simple. Type `C-h` (or `F1`) and follow the directions. If you are a first-time user, type `C-h t` for a **tutorial**.

| | |
|--|--------------------|
| remove help window | <code>C-x 1</code> |
| scroll help window | <code>C-M-v</code> |
| apropos: show commands matching a string | <code>C-h a</code> |
| describe the function a key runs | <code>C-h k</code> |
| describe a function | <code>C-h f</code> |
| get mode-specific information | <code>C-h m</code> |

Error Recovery

| | |
|---|---|
| abort partially typed or executing command | <code>C-g</code> |
| recover files lost by a system crash | <code>M-x recover-session</code> |
| undo an unwanted change | <code>C-x u</code> , <code>C-_</code> or <code>C-/</code> |
| restore a buffer to its original contents | <code>M-x revert-buffer</code> |
| redraw garbaged screen | <code>C-l</code> |

Incremental Search

| | |
|-----------------------------------|--------------------|
| search forward | <code>C-s</code> |
| search backward | <code>C-r</code> |
| regular expression search | <code>C-M-s</code> |
| reverse regular expression search | <code>C-M-r</code> |
| select previous search string | <code>M-p</code> |
| select next later search string | <code>M-n</code> |
| exit incremental search | <code>RET</code> |
| undo effect of last character | <code>DEL</code> |
| abort current search | <code>C-g</code> |

Use `C-s` or `C-r` again to repeat the search in either direction. If Emacs is still searching, `C-g` cancels only the part not matched.

Motion

| entity to move over | backward | forward |
|--|---------------------|-----------------------|
| character | <code>C-b</code> | <code>C-f</code> |
| word | <code>M-b</code> | <code>M-f</code> |
| line | <code>C-p</code> | <code>C-n</code> |
| go to line beginning (or end) | <code>C-a</code> | <code>C-e</code> |
| sentence | <code>M-a</code> | <code>M-e</code> |
| paragraph | <code>M-{</code> | <code>M-}</code> |
| page | <code>C-x [</code> | <code>C-x]</code> |
| sexp | <code>C-M-b</code> | <code>C-M-f</code> |
| function | <code>C-M-a</code> | <code>C-M-e</code> |
| go to buffer beginning (or end) | <code>M-<</code> | <code>M-></code> |
| scroll to next screen | | <code>C-v</code> |
| scroll to previous screen | | <code>M-v</code> |
| scroll left | | <code>C-x <</code> |
| scroll right | | <code>C-x ></code> |
| scroll current line to center, top, bottom | | <code>C-l</code> |
| goto line | | <code>M-g g</code> |
| goto char | | <code>M-g c</code> |
| back to indentation | | <code>M-m</code> |

Killing and Deleting

| entity to kill | backward | forward |
|---|------------------------|-----------------------|
| character (delete, not kill) | <code>DEL</code> | <code>C-d</code> |
| word | <code>M-DEL</code> | <code>M-d</code> |
| line (to end of) | <code>M-O C-k</code> | <code>C-k</code> |
| sentence | <code>C-x DEL</code> | <code>M-k</code> |
| sexp | <code>M-- C-M-k</code> | <code>C-M-k</code> |
| kill region | | <code>C-w</code> |
| copy region to kill ring | | <code>M-w</code> |
| kill through next occurrence of <i>char</i> | | <code>M-z char</code> |
| yank back last thing killed | | <code>C-y</code> |
| replace last yank with previous kill | | <code>M-y</code> |

Marking

| | |
|---------------------------------------|--|
| set mark here | <code>C-@</code> or <code>C-SPC</code> |
| exchange point and mark | <code>C-x C-x</code> |
| set mark <i>arg</i> words away | <code>M-@</code> |
| mark paragraph | <code>M-h</code> |
| mark page | <code>C-x C-p</code> |
| mark sexp | <code>C-M-@</code> |
| mark function | <code>C-M-h</code> |
| mark entire buffer | <code>C-x h</code> |

Query Replace

| | |
|--|---------------------------------------|
| interactively replace a text string | <code>M-%</code> |
| using regular expressions | <code>M-x query-replace-regexp</code> |
| Valid responses in query-replace mode are | |
| replace this one, go on to next | <code>SPC</code> or <code>y</code> |
| replace this one, don't move | <code>,</code> |
| skip to next without replacing | <code>DEL</code> or <code>n</code> |
| replace all remaining matches | <code>!</code> |
| back up to the previous match | <code>^</code> |
| exit query-replace | <code>RET</code> |
| enter recursive edit (<code>C-M-c</code> to exit) | <code>C-r</code> |

Multiple Windows

When two commands are shown, the second is a similar command for a frame instead of a window.

| | | |
|-------------------------------------|------------------------|------------------------|
| delete all other windows | <code>C-x 1</code> | <code>C-x 5 1</code> |
| split window, above and below | <code>C-x 2</code> | <code>C-x 5 2</code> |
| delete this window | <code>C-x 0</code> | <code>C-x 5 0</code> |
| split window, side by side | | <code>C-x 3</code> |
| scroll other window | | <code>C-M-v</code> |
| switch cursor to another window | <code>C-x o</code> | <code>C-x 5 o</code> |
| select buffer in other window | <code>C-x 4 b</code> | <code>C-x 5 b</code> |
| display buffer in other window | <code>C-x 4 C-o</code> | <code>C-x 5 C-o</code> |
| find file in other window | <code>C-x 4 f</code> | <code>C-x 5 f</code> |
| find file read-only in other window | <code>C-x 4 r</code> | <code>C-x 5 r</code> |
| run Dired in other window | <code>C-x 4 d</code> | <code>C-x 5 d</code> |
| find tag in other window | <code>C-x 4 .</code> | <code>C-x 5 .</code> |
| grow window taller | | <code>C-x ^</code> |
| shrink window narrower | | <code>C-x {</code> |
| grow window wider | | <code>C-x }</code> |

Formatting

| | |
|--|----------------------|
| indent current line (mode-dependent) | <code>TAB</code> |
| indent region (mode-dependent) | <code>C-M-\</code> |
| indent sexp (mode-dependent) | <code>C-M-q</code> |
| indent region rigidly <i>arg</i> columns | <code>C-x TAB</code> |
| indent for comment | <code>M-;</code> |
| insert newline after point | <code>C-o</code> |
| move rest of line vertically down | <code>C-M-o</code> |
| delete blank lines around point | <code>C-x C-o</code> |
| join line with previous (with <i>arg</i> , next) | <code>M-^</code> |
| delete all white space around point | <code>M-\</code> |
| put exactly one space at point | <code>M-SPC</code> |
| fill paragraph | <code>M-q</code> |
| set fill column to <i>arg</i> | <code>C-x f</code> |
| set prefix each line starts with | <code>C-x .</code> |
| set face | <code>M-o</code> |

Case Change

| | |
|------------------|----------------------|
| uppercase word | <code>M-u</code> |
| lowercase word | <code>M-l</code> |
| capitalize word | <code>M-c</code> |
| uppercase region | <code>C-x C-u</code> |
| lowercase region | <code>C-x C-l</code> |

The Minibuffer

The following keys are defined in the minibuffer.

| | |
|---|------------------|
| complete as much as possible | <code>TAB</code> |
| complete up to one word | <code>SPC</code> |
| complete and execute | <code>RET</code> |
| show possible completions | <code>?</code> |
| fetch previous minibuffer input | <code>M-p</code> |
| fetch later minibuffer input or default | <code>M-n</code> |
| regex search backward through history | <code>M-r</code> |
| regex search forward through history | <code>M-s</code> |
| abort command | <code>C-g</code> |

Type `C-x ESC ESC` to edit and repeat the last command that used the minibuffer. Type `F10` to activate menu bar items on text terminals.

GNU Emacs Reference Card

Buffers

| | |
|-----------------------|---------|
| select another buffer | C-x b |
| list all buffers | C-x C-b |
| kill a buffer | C-x k |

Transposing

| | |
|-----------------------------|---------|
| transpose characters | C-t |
| transpose words | M-t |
| transpose lines | C-x C-t |
| transpose sexps | C-M-t |

Spelling Check

| | |
|---------------------------------------|-------------------|
| check spelling of current word | M-\$ |
| check spelling of all words in region | M-x ispell-region |
| check spelling of entire buffer | M-x ispell-buffer |
| toggle on-the-fly spell checking | M-x flyspell-mode |

Tags

| | |
|--|------------------------|
| find a tag (a definition) | M-. |
| find next occurrence of tag | C-u M-. |
| specify a new tags file | M-x visit-tags-table |
| regex search on all files in tags table | M-x tags-search |
| run query-replace on all the files | M-x tags-query-replace |
| continue last tags search or query-replace | M-, |

Shells

| | |
|--|-----------|
| execute a shell command | M-! |
| execute a shell command asynchronously | M-& |
| run a shell command on the region | M- |
| filter region through a shell command | C-u M- |
| start a shell in window *shell* | M-x shell |

Rectangles

| | |
|-------------------------------------|---------|
| copy rectangle to register | C-x r r |
| kill rectangle | C-x r k |
| yank rectangle | C-x r y |
| open rectangle, shifting text right | C-x r o |
| blank out rectangle | C-x r c |
| prefix each line with a string | C-x r t |

Abbrevs

| | |
|--|-----------|
| add global abbrev | C-x a g |
| add mode-local abbrev | C-x a l |
| add global expansion for this abbrev | C-x a i g |
| add mode-local expansion for this abbrev | C-x a i l |
| explicitly expand abbrev | C-x a e |
| expand previous word dynamically | M-/ |

Miscellaneous

| | |
|-------------------|-----------------|
| numeric argument | C-u <i>num</i> |
| negative argument | M-- |
| quoted insert | C-q <i>char</i> |

Regular Expressions

| | | |
|---|--------------|-------|
| any single character except a newline | . | (dot) |
| zero or more repeats | * | |
| one or more repeats | + | |
| zero or one repeat | ? | |
| quote special characters | \ | |
| quote regular expression special character <i>c</i> | \c | |
| alternative (“or”) | | |
| grouping | (...) | |
| shy grouping | (: ? ...) | |
| explicit numbered grouping | (: NUM ...) | |
| same text as <i>n</i> th group | \n | |
| at word break | \b | |
| not at word break | \B | |

| | | |
|----------------------------------|--------------------|---------------------|
| entity | match start | match end |
| line | ^ | \$ |
| word | \< | \> |
| symbol | _< | _> |
| buffer | \‘ | \’ |
| class of characters | match these | match others |
| explicit set | [...] | [^ ...] |
| word-syntax character | \w | \W |
| character with syntax <i>c</i> | \sc | \Sc |
| character with category <i>c</i> | \cc | \Cc |

International Character Sets

| | |
|------------------------------------|--------------------------|
| specify principal language | C-x RET l |
| show all input methods | M-x list-input-methods |
| enable or disable input method | C-\ |
| set coding system for next command | C-x RET c |
| show all coding systems | M-x list-coding-systems |
| choose preferred coding system | M-x prefer-coding-system |

Info

| | |
|---|-------|
| enter the Info documentation reader | C-h i |
| find specified function or variable in Info | C-h S |

Moving within a node:

| | |
|-------------------|-----|
| scroll forward | SPC |
| scroll reverse | DEL |
| beginning of node | b |

Moving between nodes:

| | |
|--|----------|
| next node | n |
| previous node | P |
| move up | u |
| select menu item by name | m |
| select <i>n</i> th menu item by number (1–9) | <i>n</i> |
| follow cross reference (return with 1) | f |
| return to last node you saw | l |
| return to directory node | d |
| go to top node of Info file | t |
| go to any node by name | g |

Other:

| | |
|----------------------------------|---|
| run Info tutorial | h |
| look up a subject in the indices | i |
| search nodes for regexp | s |
| quit Info | q |

Registers

| | |
|--------------------------------------|-----------|
| save region in register | C-x r s |
| insert register contents into buffer | C-x r i |
| save value of point in register | C-x r SPC |
| jump to point saved in register | C-x r j |

Keyboard Macros

| | |
|--|-------------------------|
| start defining a keyboard macro | C-x (|
| end keyboard macro definition | C-x) |
| execute last-defined keyboard macro | C-x e |
| append to last keyboard macro | C-u C-x (|
| name last keyboard macro | M-x name-last-kbd-macro |
| insert Lisp definition in buffer | M-x insert-kbd-macro |

Commands Dealing with Emacs Lisp

| | |
|---|------------------|
| eval sexp before point | C-x C-e |
| eval current defun | C-M-x |
| eval region | M-x eval-region |
| read and eval minibuffer | M-: |
| load a Lisp library from load-path | M-x load-library |

Simple Customization

| | |
|---|---------------|
| customize variables and faces | M-x customize |
| Making global key bindings in Emacs Lisp (example): | |
| (global-set-key (kbd "C-c g") 'search-forward) | |
| (global-set-key (kbd "M-#") 'query-replace-regexp) | |

Writing Commands

| |
|---|
| (defun <i>command-name</i> (<i>args</i>) " <i>documentation</i> " (interactive " <i>template</i> ") <i>body</i>) |
| An example: |
| (defun this-line-to-top-of-window (line) "Reposition current line to top of window. With prefix argument LINE, put point on LINE." (interactive "P") (recenter (if (null line) 0 (prefix-numeric-value line)))) |

The **interactive** spec says how to read arguments interactively. Type C-h f **interactive** RET for more details.