

# Emacs Command Reference Sheet

## Leaving Emacs

suspend Emacs (or iconify it under X) C-z  
exit Emacs permanently C-x C-c

## Files

read a file into Emacs C-x C-f  
save a file back to disk C-x C-s  
save all files C-x s  
insert contents of another file into this buffer C-x i  
replace this file with another C-x C-v  
write buffer to a specified file C-x C-w

## Getting Help

remove Help window C-h  
scroll Help window ESC C-l  
apropos: show commands matching string C-h a  
show the function a key runs C-h c  
describe a function C-h f  
get mode-specific information C-h m

## Error Recovery

abort partially typed or executing command C-g  
recover a file lost by a system crash M-x recover-file  
undo an unwanted change C-x u or C-  
restore a buffer to original contents M-x revert-buffer  
redraw garbaged screen C-l

## Incremental Search

search forward C-s  
search backward C-r  
regular expression search C-M-s  
reverse regular expression search C-M-r  
select previous search string M-p  
select next later search string M-n  
exit incremental search RET  
undo effect of last character DEL  
abort current search C-g  
C-s or C-r again to repeat the search C-g cancels

## Motion

entity to move over backward forward  
character C-b C-f  
word M-b M-f  
line C-p C-n  
go to line beginning (or end) C-a C-e  
sentence M-a M-e  
paragraph M- M-"  
page C-x [ C-x ]  
sexp C-M-b C-M-f  
function C-M-a C-M-e  
go to buffer beginning (or end) M-< M->  
scroll to next screen C-v  
scroll to previous screen M-v  
scroll left C-x <  
scroll right C-x >  
scroll current line to center of screen C-u C-l

## Killing and Deleting

entity to kill backward forward  
character (delete, not kill) DEL C-d  
word M-DEL M-d  
line (to end of) M-0 C-k C-k  
sentence C-x DEL M-k  
sexp M-- C-M-k C-M-k  
kill region C-w  
copy region to kill ring M-w  
kill through next occurrence of char M-z char  
yank back last thing killed C-y  
replace last yank with previous kill M-y

## Marking

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

## Query Replace

interactively replace a text string M-%  
using regular expressions M-x query-replace-regexp  
*Valid responses in query-replace mode are:*

replace this one, go on to next SPC  
replace this one, don't move ,  
skip to next without replacing DEL  
replace all remaining matches !  
back up to the previous match ^  
exit query-replace ESC  
enter recursive edit (C-M-c to exit) C-r

## Shells

execute a shell command 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

## Multiple Windows

delete all other windows C-x 1  
delete this window C-x 0  
split window in two vertically C-x 2  
split window in two horizontally C-x 3  
scroll other window C-M-v  
switch cursor to another window C-x o  
shrink window shorter M-x shrink-window  
grow window taller C-x ^  
shrink window narrower C-x -  
grow window wider C-x "  
select buffer in other window C-x 4 b  
display buffer in other window C-x 4 C-o  
find file in other window C-x 4 f  
find file read-only in other window C-x 4 r  
run Dired in other window C-x 4 d  
find tag in other window C-x 4 .

## Formatting

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

## Case Change

uppercase word M-u  
lowercase word M-l  
capitalize word M-c  
uppercase region C-x C-u  
lowercase region C-x C-l  
capitalize region M-x capitalize-region

## The Minibuffer

*The following keys are defined in the minibuffer:*

complete as much as possible TAB  
complete up to one word SPC  
complete and execute RET  
show possible completions ?  
fetch previous minibuffer input M-p  
fetch next later minibuffer input M-n  
regexp search backward through history M-r  
regexp search forward through history M-s  
abort command C-g

Type C-x ESC ESC to edit and repeat the last command that used the minibuffer. The following keys are then defined:

previous minibuffer command M-p  
next minibuffer command M-n

## 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

## 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  
regexp 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-

## 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 M-x clear-rectangle  
prefix each line with a string M-x string-rectangle

## 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-/

## Regular Expressions

any single character except a newline . (dot)  
zero or more repeats \*  
one or more repeats +  
zero or one repeat ?  
any character in the set [ : : ]  
any character not in the set [ ^ : : ]  
beginning of line ^  
end of line \$  
quote a special character c \"c"  
alternative ("or") "  
grouping "( : : )"  
nth group "n"  
beginning of buffer "  
end of buffer "'  
word break "b"  
not beginning or end of word "B"  
beginning of word "<  
end of word ">  
any word-syntax character "w"  
any non-word-syntax character "W"  
character with syntax c "sc"  
character with syntax not c "Sc"

## 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

## Info

enter the Info documentation reader C-h i  
*Moving within a node:*  
scroll forward SPC  
scroll reverse DEL  
beginning of node . (dot)

## Moving between nodes:

next node n  
previous node p  
move up u  
select menu item by name m  
select nth menu item by number (1-5) f  
follow cross reference (return with l) n  
return to last node you saw l  
return to directory node d  
go to any node by name g

## Other:

run Info tutorial h  
list Info commands ?  
quit Info q  
search nodes for regexp s

## 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  
eval entire buffer M-x eval-current-buffer  
read and eval minibuffer M-ESC  
re-execute last minibuffer command C-x ESC ESC  
read and eval Emacs Lisp file M-x load-file  
load from standard system directory M-x load-library

## Simple Customization

Here are some examples of binding global keys in Emacs Lisp. Note that you cannot say ""M-#"; you must say ""e#".

(global-set-key "C-cg" goto-line)  
(global-set-key "C-xC-k" kill-region)  
(global-set-key "e#" 'query-replace-regexp)

An example of setting a variable in Emacs Lisp:  
(setq backup-by-copying-when-linked t)

## Writing Commands

(defun command-name (args)  
"documentation"  
(interactive "template") body)

## An example:

(defun this-line-to-top-of-window (line)  
"Reposition line point is on to top of window.  
With ARG, put point on line ARG.

Negative counts from bottom."  
(interactive "P")

(recenter (if (null line)  
0  
(prefix-numeric-value line))))