Motion

entity to move over
character
word
line
go to line beginning (or end)
sentence
paragraph
page
sexp
function
go to buffer beginning (or end)
scroll to next screen
scroll to previous screen
scroll left
scroll right
scroll current line to center of screen

Killing and Deleting

entity to kill
character (delete, not kill)
word
line (to end of)
sentence
sexp
kill region
copy region to kill ring
kill through next occurrence of char
yank back last thing killed
replace last yank with previous kill

Marking

set mark here
exchange point and mark
set mark any words away
mark paragraph
mark page
mark sexp
mark function
mark entire buffer

Query Replace

interactively replace a text string
using regular expressions
Valid responses in query-replace mode are
replace this one, go on to next
replace this one, don't move
skip to next without replacing
replace all remaining matches
back up to the previous match
exit query-replace
enter recursive edit (C-M-c to exit)

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 a
display buffer in other window
find file in other window
C-x f
find file read-only in other window
run Died in other window
M-x make-window
find tag in other window
C-x t

Formatting

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

Case Change

upercase word
C-u
lowercase word
C-l
capitalize word
C-c
upercase region
C-x M-q
lowercase region
C-x M-q
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 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

Incremental Search

search forward
C-s
search backward
C-b
regular expression search
M-x
reverse regular expression search
C-M-x
select previous search string
M-p
select next search string
M-n
exit incremental search
C-M-x
undo effect of last character
DEL
abort current search
C-g

Use C-s or C-b again to repeat the search in either direction. If Emacs is still searching, C-g cancels only the part not done.
### 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 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
- **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-!
- **run a shell command on the region**: C-x 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**: 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
- **explicitly expand abbrev**: C-x a i l
- **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
- **alternative ("or")**: | [... |
- **grouping**: ( ... )
- **with group**: [ ... ]
- **beginning of buffer**: 
- **end of buffer**: 
- **word break**: |
- **not beginning or end of word**: 
- **beginning of word**: 
- **end of word**: 
- **any word-syntax character**: 
- **any non-word-syntax character**: 
- **character with syntax c**: 
- **character with syntax not c**: 

#### 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 s P
- **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 menu item by number (1-5)**: n
- **follow cross reference (return with 1)**: 1
- **return to last node you saw**: d
- **return to directory node**: g

#### Keyboard Macros
- **start defining a keyboard macro**: C-x (k
- **end keyboard macro definition**: C-x )
- **execute last-defined keyboard macro**: C-x e
- **append to last keyboard macro**: C-x C-x
- **name last keyboard macro**: C-x name-lst-keyboard-macro
- **insert Lisp definition in buffer**: M-x insert-lisp-definition
- **read and eval Emacs Lisp file**: M-x load-file
- **load from standard system directory**: M-x load-library

#### Commands Dealing with Emacs Lisp
- **eval sexp before point**: C-x C-e
- **eval current defun**: C-x M-x
- **eval region**: M-x eval-region
- **eval entire buffer**: M-x eval-current-buffer
- **read and eval minibuffer**: M-E
- **re-execute last minibuffer command**: C-x C-x ESC ESC
- **read and eval Emacs Lisp file**: M-x load-file

#### Simple Customization
Here are some examples of binding global keys in Emacs Lisp. Note that you cannot say "M-#": you must say "\#".
- **(global-set-key "C-c g" 'goto-line)**
- **(global-set-key "C-k" 'kill-region)**
- **(global-set-key "M-#" '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)**
  "Redo position line point is on top of window."
  (interactive "p")
  (recenter (if (null line))
  o)
  "(prefix-numeric-value line)"

The argument to interactive is a string specifying how to get the arguments when the function is called interactively. Type C-h f interactive for more information.

---

**Copyright © 1993 Free Software Foundation, Inc.**

**For GNU Emacs version 20 on Unix systems:**

**Permission is granted to make and distribute copies of this card provided the copyright notice and this permission notice are preserved on all copies.**

For copies of the GNU Emacs manual, write to the Free Software Foundation, Inc., 51 Franklin Street, Cambridge, MA 02139.