Starting Emacs

To enter GNU Emacs 21, just type its name: emacs
To read in a file to edit, see Files, below.

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 the file you really want C-x C-v
write buffer to a specified file C-x C-w
version control checkin/checkout C-x C-q

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 C-x 1
scroll help window C-M-v
apropos: show commands matching a 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 its original contents M-x revert-buffer redraw garbled 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

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 done.
<table>
<thead>
<tr>
<th>entity to move over</th>
<th>backward</th>
<th>forward</th>
</tr>
</thead>
<tbody>
<tr>
<td>character</td>
<td>C-b</td>
<td>C-f</td>
</tr>
<tr>
<td>word</td>
<td>M-b</td>
<td>M-f</td>
</tr>
<tr>
<td>line</td>
<td>C-p</td>
<td>C-n</td>
</tr>
<tr>
<td>go to line beginning (or end)</td>
<td>C-a</td>
<td>C-e</td>
</tr>
<tr>
<td>sentence</td>
<td>M-a</td>
<td>M-e</td>
</tr>
<tr>
<td>paragraph</td>
<td>M-{</td>
<td>M-}</td>
</tr>
<tr>
<td>page</td>
<td>C-x [</td>
<td>C-x ]</td>
</tr>
<tr>
<td>sexp</td>
<td>C-M-b</td>
<td>C-M-f</td>
</tr>
<tr>
<td>function</td>
<td>C-M-a</td>
<td>C-M-e</td>
</tr>
<tr>
<td>go to buffer beginning (or end)</td>
<td>M-&lt;</td>
<td>M-&gt;</td>
</tr>
<tr>
<td>scroll to next screen</td>
<td>C-v</td>
<td></td>
</tr>
<tr>
<td>scroll to previous screen</td>
<td>M-v</td>
<td></td>
</tr>
<tr>
<td>scroll left</td>
<td>C-x &lt;</td>
<td></td>
</tr>
<tr>
<td>scroll right</td>
<td>C-x &gt;</td>
<td></td>
</tr>
<tr>
<td>scroll current line to center of screen</td>
<td>C-u</td>
<td>C-1</td>
</tr>
</tbody>
</table>

**Killing and Deleting**

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

**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 | RET
enter recursive edit (C-M-c to exit) | C-r
When two commands are shown, the second is the "other" transit:
delete all other windows  C-x 1
split window, above and below  C-x 2  C-x 5 2
delete this window  C-x 0  C-x 5 0
split window, side by side  C-x 3
scroll other window  C-M-v
switch cursor to another window  C-x 0  C-x 5 0
select buffer in other window  C-x 4 b  C-x 5 b
display buffer in other window  C-x 4 C-o  C-x 5 C-o
find file in other window  C-x 4 f  C-x 5 f
find file read-only in other window  C-x 4 r  C-x 5 r
run Dired in other window  C-x 4 d  C-x 5 d
find tag in other window  C-x 4 .  C-x 5 .
grow window taller  C-x ^
shrink window narrower  C-x {  grow window wider  C-x }

Formatting
indent current line (mode-dependent)  TAB
indent region (mode-dependent)  C-M-\nindent sexp (mode-dependent)  C-M-q
indent region rigidly any 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-\nput 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 .
set face  M-g

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

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 later minibuffer input or default  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. Type F10 to activate the menu bar using the mini-
buffer.
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-,;

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

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-/
any single character except a newline . (dot)  
zero or more repeats *  
one or more repeats +  
zero or one repeat ?  
quote regular expression special character c \c  
alternative ("or") \|  
grouping \{ ... \}  
same text as nth group \n  
at word break \b  
not at word break \B  

<table>
<thead>
<tr>
<th>entity</th>
<th>match start</th>
<th>match end</th>
</tr>
</thead>
<tbody>
<tr>
<td>line</td>
<td>~</td>
<td>$</td>
</tr>
<tr>
<td>word</td>
<td>&lt;</td>
<td>&gt;</td>
</tr>
<tr>
<td>buffer</td>
<td>\’</td>
<td>\’</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class of characters</th>
<th>match these</th>
<th>match others</th>
</tr>
</thead>
<tbody>
<tr>
<td>explicit set</td>
<td>[ ... ]</td>
<td>[^ ... ]</td>
</tr>
<tr>
<td>word-syntax character</td>
<td>\w</td>
<td>\W</td>
</tr>
<tr>
<td>character with syntax c</td>
<td>\sc</td>
<td>\Sc</td>
</tr>
</tbody>
</table>

**International Character Sets**

specify principal language \M-x set-language-environment  
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 C-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 (1-9) n  
follow cross reference (return with 1) f  
return to last node you saw l  
return to directory node d  
go to any node by name g  

Other:

run Info tutorial h  
quit Info q  
search nodes for regexp \M-s  

5
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 from standard system directory          M-x load-library  

Simple Customization

customize variables and faces          M-x customize  

Making global key bindings in Emacs Lisp (examples):
(gLOBAL-set-key "\C-cg" 'goto-line)
(gLOBAL-set-key "\M-#" 'query-replace-regexp)

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."
   (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 for more details.

Copyright © 1997 Free Software Foundation, Inc.
v2.2 for GNU Emacs version 21, 1997
designed by Stephen Gildea

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.,
59 Temple Place, Suite 330, Boston, MA 02111-1307 USA