# UNIX Editors Summary

**An ex/vi reference**

# Contents

# 1 Ex Quick Reference

## Entering/leaving Ex

| Command | Action |
|---|---|
| `% ex $NAME` | edit `$NAME`, start at end |
| `% ex +$N $NAME` | . . . at line n |
| `% ex -t tag` | start at tag |
| `% ex -r` | list saved files |
| `% ex -r $NAME` | recover file `$NAME` |
| `% ex $NAME ...` | edit first; rest via `:n` |
| `% ex -R $NAME` | read only mode |
| `:x` | exit, saving changes |
| `:q!` | exit, discarding changes |

## Ex states

| State | Entry | Input | Exit |
|---|---|---|---|
| Command | normal / initial | `:` prompts | kill char cancels partial commands |
| Insert | `a`, `i`, and `c` | arbitrary text | empty line with only a . |
| Open / visual | `open` or `vi` | same as above | `Q` or `\^\\` |

## Specifying terminal type

| Command | Version |
|---|---|
| `% setenv TERM $TYPE` | `csh` and all version 6 |
| `$ TERM=$TYPE; export TERM` | `sh` in version 7 |

See also `tset(1)`

## Ex commands

| Command | Abbreviation | Command | Abbreviation | Command | Abbreviation |
|---|---|---|---|---|---|
| `abbrev` | `ab` | `next` | `n` | `unabbrev` | `una` |
| `append` | `a` | `number` | `nu` | `undo` | `u` |
| `args` | `ar` | `open` | `o` | `unmap` | `unm` |
| `change` | `c` | `preserve` | `pre` | `version` | `ve` |
| `copy` | `co` | `print` | `p` | `visual` | `vi` |
| `delete` | `d` | `put` | `pu` | `write` | `w` |
| `edit` | `e` | `quit` | `q` | `xit` | `x` |
| `file` | `f` | `read` | `re` | `yank` | `ya` |
| `global` | `g` | `recover` | `rec` | `window` | `z` |
| `insert` | `i` | `rewind` | `rew` | `escape` | `!` |
| `join` | `j` | `set` | `se` | `lshift` | |
| `list` | `l` | `shell` | `sh` | `print next` | `CR` |
| `map` | | `source` | `so` | `resubst` | `&` |
| `mark` | `ma` | `stop` | `st` | `rshift` | `>` |
| `move` | `m` | `substitute` | `s` | `scroll` | `^D` |

## Ex command addresses

| Command | Address | Command | Address |
|---|---|---|---|
| `n` | line `n` | `/$PATTERN` | next with `$PATTERN` |
| `.` | current | `?$PATTERN` | previous with `$PATTERN` |
| `$` | last | `x-$N` | `n` before `x` |
| `+` | next | `x,y` | `x` through `y` |
| `-` | previous | `'x` | marked with `x` |
| `+$N` | `n` forward | `''` | previous context |
| `%` | `1,$` | | |

## Initializing options

| Command | Action |
|---|---|
| `EXINIT` | place `set`s here in environment variable. |
| `set x` | enable option |
| `set nox` | disable option |
| `set x=$VAL` | give `x` value `$VAL` |
| `set` | show changed options |
| `set all` | show all options |
| `set x?` | show value of option `x` |

## Useful options

| Option | Abbreviation | Description |
|---|---|---|
| autoindent | ai | supply indent |
| autowrite | aw | write before chaning files |
| ignorecase | ic | in scanning |
| lisp | | () and {} are s-expressions |
| list | | print ^I for tab, $ at end |
| magic | | ., [, and * special in patterns |
| number | nu | number lines |
| paragraphs | para | macro $NAMEs which start ... |
| redraw | | simulate smart terminal |
| scroll | | command mode lines |
| sections | sect | macro $NAMEs ... |
| shiftwidth | sw | for >, and input ^D |
| showmatch | sw | to ) and } as typed |
| slowopen | slow | choke updates during insert |
| window | | visual mode lines |
| wrapscan | ws | around end of buffer? |
| wrapmargin | wm | automatic line splitting |

## Scanning pattern formation

| Symbol | Meaning |
|---|---|
| ^ | beginning of line |
| $ | end of line |
| . | any character |
| \ | beginning of word |
| \> | end of word |
| [str] | any character in str |
| [^str] | ... not in str |
| [x-y] | ... between x and y |
| * | any number of preceding |

# 2 Vi Quick Reference

## Entering/leaving Vi

| Command | Action |
|---|---|
| `% vi $FILE` | edit `$FILE` at top |
| `% vi +$N $FILE` | ... at line `n` |
| `% vi + $FILE` | ... at the end |
| `% vi -r` | list saved files |
| `% vi -r $FILE` | recover `$FILE` |
| `% vi -t $TAG` | start at `$TAG` |
| `% vi +/$PATTERN $FILE` | search for `$PATTERN` |
| `% view $FILE` | read-only mode |
| `ZZ` | exit, saving changes |
| `^Z` | stop to resume later |

## The display

| Line | Function |
|---|---|
| Last | Error messages; echoing input to `:`, `/`, `?`, and `!`; feedback about I/O, etc. |
| `@` | Screen placeholder, not in file |
| `~` | Screen placeholder, lines past EOF |
| `~x` | Control characters, `?` is delete |
| tabs | expand to spaces, cursor at last |

## Vi states

| State | Entry | Notes |
|---|---|---|
| Command | normal / initial | others return here |
| Insert | `a, i, A, I, o, O, c, C, s, S, R` | arbitrary text terminates with `ESC` |
| Last line | reading input for `:, /, ?, !` | `ESC` terminates, `CR` executes |

## Counts before Vi commands

| Symbol | Meaning |
|---|---|
| `z, G` | line |
| `\|` | column |
| `a, i, A, I` | replicate insert |
| most others | repeat effect |

## Simple commands

| Symbol | Meaning |
|---|---|
| `dw` | delete a word |
| `de` | . . . leaving punctuation |
| `dw` | delete a word |
| `dd` | delete a line |
| `3dd` | delete three line |
| `itextESC` | insert `text`, return to command state |
| `cwnewESC` | change word to `new` |
| `easESC` | add `s` to end of word |
| `xp` | transpose characters (cut and paste) |

## Interrupting, cancelling

| Symbol | Meaning |
|---|---|
| `ESC` | end insert or incomplete command |
| `^?` | (delete or rubout) interrupts |
| `^L` | reprint screen if `^?` scrambles it |

## File manipulation

| Symbol | Meaning |
|---|---|
| `:w` | write back changes |
| `:wq, :x` | write and quit |
| `:q` | quit |
| `:q!` | quit, discard changes |
| `:e $FILE` | edit `$FILE` |
| `:e!` | re-edit, discard changes |
| `:e + $FILE` | edit `$FILE`, starting at end |
| `:e +$N` | edit starting at line `$N` |
| `:e #` | edit alternate file |
| `^^` | synonym for `:e #` |
| `:w $FILE` | write `$FILE` |
| `:w! $FILE` | overwrite `$FILE` |
| `:sh` | run shell, then return |
| `:!$CMD` | run `$CMD`, then return |
| `:n` | edit next file in arglist |
| `:n args` | specify new arglist |
| `:f` | show current file and line |
| `^G` | synonym for `:f` |
| `:ta $TAG` | tag file entry `$TAG` |
| `^]` | `:ta`, following word is tag |

## Positioning within file

| Command | Action |
|---|---|
| `^F` | forward one screenful |
| `^B` | backward one screenful |
| `^D` | scroll down half screen |
| `^U` | scroll up half screen |
| `G` | goto line (default EOF) |
| `/$PAT` | next line matching `$PAT` |
| `?$PAT` | previous line matching `$PAT` |
| `n` | repeat last `/` or `?` |
| `N` | reverse last `/` or `?` |
| `/$PAT/+$N` | `$N`'th line after `$PAT` |
| `/$PAT/-$N` | `$N`'th line before `$PAT` |
| `]]` | next section / function |
| `[[` | previous section / function |
| `%` | find matching `()` or `{}` |

## Adjusting the screen

| Command | Action |
|---|---|
| ^L | clear and redraw |
| ^R | retype, eliminate @ lines |
| zCR | redraw, current at window top |
| z- | ... at bottom |
| z. | ... at center |
| /$PAT/z- | $PAT line at bottom |
| z$N. | use $N line window |
| ^E | scroll window down 1 line |
| ^Y | scroll window up 1 line |

## Marking and returning

| Command | Action |
|---|---|
| `` | previous context |
| '' | ... at first non-whitespace character in line |
| m$L | mark position with letter $L |
| `$L | to mark $L |
| 'x | ... at first non-whitespace character in line |

## Line positioning

| Command | Location |
|---|---|
| H | home window line |
| L | last window line |
| M | middle window line |
| + | next line at first non-whitespace character |
| - | previous line at first non-whitespace character |
| CR | return, same as + |
| or j | next line, same column |
| ^ or k | previous line, same column |

## Character positioning

| Command | Location |
|---|---|
| `^` | first non-whitespace character |
| `0` | beginning of line |
| `$` | end of line |
| `l` or $\rightarrow$ | forward |
| `h` or $\leftarrow$ | backward |
| `^H` | same as `h` |
| space | same as `l` |
| `f$X` | find `$X` forward |
| `F$X` | find `$X` backward |
| `t$X` | upto `$X` forward |
| `T$X` | upto `$X` backward |
| `;` | repeat last `f, F, t, or T` |
| `,` | inverse f `;` |
| `\|` | to specified column |
| `%` | find matching `(, {, ), or }` |

## Words, sentences, paragraphs

| Command | Movement |
|---|---|
| `w` | forward word |
| `b` | backward word |
| `e` | to end of word |
| `)` | to next sentence |
| `}` | to next paragraph |
| `(` | backward sentence |
| `{` | backward paragraph |
| `W` | blank delimited word |
| `B` | back `W` |
| `E` | to end of `W` |

## Corrections during insert

| Command | Action |
|---|---|
| ^H | erase last character |
| ^W | erases last word |
| erase | your erase, same as ^H |
| kill | your kill, erase input this line |
| \ | escapes ^H, your erase and kill |
| ESC | ends insertion, back to command |
| ^? | interrupt, terminates insert |
| ^D | backtab over autoindent |
| ^^D | kill autoindent, save for next |
| 0^D | ... but at margin next also |
| ^V | quote non-printing character |

## Insert and replace

| Command | Action |
|---|---|
| a | append after cursor |
| i | insert before |
| A | append at end of line |
| I | insert before first non-blank |
| o | open line below |
| O | open above |
| rx | replace single char with x |
| R | replace characters |

## Operators (double to affect lines)

| Command | Action |
|---|---|
| d | delete |
| c | change |
| < | left shift |
| > | right shift |
| ! | filter through command |
| = | indent for LISP |
| y | yank lines to buffer |

## Miscellaneous operations

| Command | Action |
|---|---|
| `C` | change rest of line |
| `D` | delete rest of line |
| `s` | substitute chars |
| `S` | substitute lines |
| `J` | join lines |
| `x` | delete characters |
| `X` | . . . before cursor |

## Yank and put

| Command | Action |
|---|---|
| `Y` | yank lines |
| `p` | put back lines |
| `P` | put before |
| `"xp` | put from buffer x |
| `"xy` | yank to buffer x |
| `"xd` | delete into buffer x |

## Undo, redo, retrieve

| Command | Action |
|---|---|
| `u` | undo last change |
| `U` | restore current line |
| `.` | repeat last change |
| `"dp` | retrieve d'th last delete |

## Commands for LISP

| Command | Movement |
|---|---|
| `)` | Forward s-expression |
| `}` | . . . but don't stop at atoms |
| `(` | Backward s-expression |
| `{` | . . . but don't stop at atoms |