Terminal Power User Cheat Sheet

A REFERENCE FOR DAILY SHELL WORK

`Ctrl` = ^ . `Alt` = Option on macOS ZSH zsh-only BASH MACOS

Type `man cmd` or `cmd --help` for full reference

LINE NAVIGATION

- `Ctrl+A / Ctrl+E` Jump to start / end of line
- `Alt+F / Alt+B` Move forward / back one word
- `Ctrl+F / Ctrl+B` Move forward / back one char
- `Ctrl+XX` Toggle cursor start and current pos

EDITING & KILL RING

- `Ctrl+U` Cut to start of line ZSH

zsh: cuts whole line

- `Ctrl+K` Cut to end of line
- `Ctrl+W` Cut word before cursor
- `Alt+D` Cut word after cursor
- `Ctrl+Y` Paste (yank) last cut text
- `Alt+Y` Cycle kill ring (after Ctrl+Y)
- `Ctrl+_` Undo last edit
- `Ctrl+T / Alt+T` Swap last two chars / words
- `Alt+U / Alt+L` Uppercase / lowercase word
- `Ctrl+X Ctrl+E` Open line in $EDITOR

Kill ring remembers multiple cuts. Ctrl+Y pastes most recent, Alt+Y cycles older ones.

HISTORY SEARCH

- `Ctrl+R` Reverse search history
- `Ctrl+S` Forward search history LINUX
- `Ctrl+G` Cancel search, restore line
- `Ctrl+P / Ctrl+N` Previous / next command
- `Alt+.` Last arg of prev cmd (repeat to cycle)
- `history | grep x` Search history for x

BANG (!) HISTORY EXPANSION

- `!!` Repeat entire last command
- `sudo !!` Re-run last command with sudo
- `!$` Last argument of previous command
- `!^` First argument of previous command
- `!:2` Second argument of previous command
- `!:2-4 / !*` Args 2-4 / all args of prev cmd
- `!str / !?str` Most recent cmd starting / containing str
- `^old^new` Re-run last cmd, replacing old with new
- `!:p` Print (don't exec) expanded command
- `!$:h` Head (directory) of last argument
- `!$:t / !$:r` Tail (filename) / remove extension

Recall & reuse parts of previous commands -- powerful, rarely taught. Ex: cp file.txt /long/path/ then cd !$:h -> cd /long/path/.

PROCESS CONTROL

- `Ctrl+C` Kill foreground process (SIGINT)
- `Ctrl+Z` Suspend foreground process (SIGTSTP)
- `bg / fg` Resume suspended in bg / fg
- `jobs` List background & suspended jobs
- `cmd &` Run command in background
- `kill %1` Kill job number 1
- `disown %1` Detach job 1 (survives logout)
- `nohup cmd &` Run cmd immune to hangups
- `Ctrl+D` Send EOF (exit shell if line empty)
- `Ctrl+L` Clear screen (keeps scrollback)

WILDCARDS & GLOBBING

- `*` Match any characters (zero or more)
- `?` Match exactly one character
- `[abc] / [a-z]` Match one of set / range
- `[!abc]` Match any character NOT in set
- `**` Recursive subdirs BASH

zsh: on by default. bash: needs shopt -s globstar.

*Globs are not regex. * means 'any string', not 'zero or more of preceding'.*

BRACE EXPANSION

- `*.{js,ts}` Expands to *.js *.ts
- `file{1..5}.txt` file1.txt file2.txt ... file5.txt
- `{01..10}` Zero-padded: 01 02 ... 10
- `mv f.txt{,.bak}` Expands to mv f.txt f.txt.bak
- `mkdir -p src/{lib,bin,test}` Create multiple dirs at once

Braces expand before globbing. {a,b} is text substitution; doesn't check files exist.

SPECIFYING MULTIPLE FILES

- `cmd *.log` All .log files in current dir
- `cmd **/*.go` All .go files recursively
- `cmd *.{js,css,html}` Multiple extensions at once
- `cmd file[1-3].txt` file1.txt, file2.txt, file3.txt
- `cmd !(*.log)` All EXCEPT .log BASH

needs shopt -s extglob

- `cmd ^*.log` All EXCEPT .log ZSH

needs setopt extendedglob

NANO ESSENTIALS

- `nano file` Open file for editing
- `Ctrl+O` Save (write out) -- confirm with Enter
- `Ctrl+X` Exit (prompts to save if modified)
- `Ctrl+K` Cut current line
- `Ctrl+U` Paste cut line
- `Ctrl+W` Search -- Ctrl+W again for next match
- `Ctrl+G` Show help with all shortcuts

nano is the default editor on most Linux distros. Set with export EDITOR=nano.

VI / VIM ESSENTIALS

- `vi file` Open file (starts in normal mode)
- `i` Enter insert mode (type text)
- `Esc` Return to normal mode
- `:w` Save
- `:q` Quit (fails if unsaved changes)
- `:wq / ZZ` Save and quit
- `:q!` Quit without saving (force)
- `dd / yy / p` Delete line / yank (copy) line / paste
- `/pattern` Search -- n next, N previous
- `u / Ctrl+R` Undo / redo

vi is on every Unix system. Set as default: export EDITOR=vi.

! PLATFORM SETUP -- READ THIS FIRST

macOS: Alt shortcuts need terminal config.
iTerm2: Profiles > Keys > Left Option key > Esc+.
Terminal.app: Preferences > Profiles > Keyboard > Use Option as Meta key.
Linux: Ctrl+S may freeze your terminal (XON/XOFF). Add stty -ixon to .bashrc / .zshrc to enable Ctrl+S search.
macOS: watch is not installed by default. Install: brew install watch.
macOS: sed -i needs an empty arg: sed -i '' 's/old/new/g' file.
WSL: Uses Linux userspace -- everything works as on native Linux.

`Ctrl` = ^ . `Alt` = Option on macOS ZSH zsh-only BASH MACOS

Type `man cmd` or `cmd --help` for full reference

DIRECTORY NAVIGATION

- `cd -` Go to previous directory (toggle)
- `cd ~ / cd ~user` Home / another user's home
- `pushd dir / popd` Push onto dir stack / pop and cd
- `dirs -v` Show directory stack with indices
- `cd ~2` Jump to stack entry 2 ZSH

COMMAND & PROCESS SUBSTITUTION

- `$(cmd)` Replace with output of cmd inline
- `echo "$(date)"` Embed command output in strings
- `<(cmd)` Treat cmd output as a file
- `diff <(cmd1) <(cmd2)` Compare output of two cmds
- `wc -l <(grep err log)` Count without temp files

Process substitution <() creates a temp file descriptor. Incredibly useful, rarely known.

GREP ESSENTIALS

- `grep pattern file` Search for pattern in file
- `grep -r pattern dir/` Search recursively
- `grep -i pattern file` Case-insensitive search
- `grep -n pattern file` Show line numbers
- `grep -l pattern *.py` List filenames with matches
- `grep -c pattern file` Count matching lines
- `grep -v pattern file` Invert: lines NOT matching
- `grep -w word file` Match whole words only
- `grep -A3 / -B2 pat f` 3 after / 2 before match
- `grep -E "a|b" file` Extended regex (OR match)
- `cmd | grep pattern` Filter any command's output

Use grep -rn for recursive + line numbers. Combine flags: grep -rinl.

FIND ESSENTIALS

- `find . -name '*.go'` By name (case-sensitive)
- `find . -iname '*.jpg'` By name (case-insensitive)
- `find . -type f / -type d` Only files / only dirs
- `find . -mtime -7` Modified in last 7 days
- `find . -size +10M` Files larger than 10MB
- `find . -empty` Empty files and directories
- `find . -name '*.log' -delete` Find and delete (careful!)
- `find . -exec chmod +x {} +` Find and exec on results

Use + not \; with -exec to batch args (faster). {} = matched file(s).

ENVIRONMENT & ALIASES

- `export VAR=value` Set environment variable
- `echo $PATH` View current PATH
- `export PATH="$PATH:/dir"` Append dir to PATH
- `unset VAR` Remove environment variable
- `env` List all environment variables
- `alias ll='ls -la'` Create a shortcut alias
- `unalias ll` Remove an alias
- `source ~/.bashrc` Reload shell config

Changes to PATH and aliases are lost when the shell exits. Add them to .bashrc / .zshrc to persist.

REDIRECTION & PIPES

- `cmd > file` Redirect stdout (overwrite)
- `cmd >> file` Redirect stdout (append)
- `cmd 2> file` Redirect stderr to file
- `cmd &> file / 2>&1` Both streams / stderr to stdout
- `cmd | tee file` Stdout AND write to file
- `cmd1 | cmd2` Pipe stdout to stdin
- `cmd1 | xargs cmd2` stdin lines become args for cmd2

PAGER NAVIGATION (LESS / MAN)

- `/pattern` Search forward for pattern
- `n / N` Next / previous search match
- `g / G` Jump to start / end of file
- `q` Quit the pager
- `h` Show help (full key reference)

man pages use less by default. All these shortcuts work in man too.

SED & AWK ONE-LINERS

- `sed 's/old/new/g' file` Replace all occurrences
- `sed -i 's/old/new/g' f` In-place LINUX
- `sed -i '' 's/old/new/g' f` In-place MACOS
- `sed -n '5,10p' file` Print only lines 5-10
- `awk '{print $2}' file` 2nd column (space-delimited)
- `awk -F: '{print $1}' f` 1st column (colon-delimited)
- `awk '$3 > 100' file` Lines where column 3 > 100

TAR, CHMOD & WATCH

- `tar czf arch.tar.gz dir/` Create gzipped archive
- `tar xzf arch.tar.gz` Extract gzipped archive
- `tar tf arch.tar.gz` List without extracting
- `chmod +x script.sh` Make file executable
- `chmod 755 file` rwxr-xr-x (owner all, others r+x)
- `chmod 644 file` rw-r--r-- (owner rw, others r)
- `watch -n 2 cmd` Re-run every 2s MACOS

brew install watch on macOS

SPECIAL VARIABLES

- `$?` Exit code of last cmd (0 = success)
- `$!` PID of last background process
- `$$` PID of current shell

SSH ESSENTIALS

- `ssh user@host` Connect to remote host
- `ssh -p 2222 user@host` Connect on non-standard port
- `ssh-copy-id user@host` Copy pub key (passwordless)
- `scp file user@host:path/` Copy file to remote
- `scp user@host:file .` Copy file from remote
- `ssh -L 8080:lo:80 host` Local port forward (tunnel)

FILE INSPECTION

- `which cmd` Show full path of command
- `type cmd` How cmd would be interpreted
- `file myfile` Detect file type (text, binary...)
- `stat file` File metadata (size, perms, dates)
- `ln -s target link` Create symbolic link
- `readlink -f link` Resolve symlink to absolute path

USEFUL COMBOS

- `wc -l file` Count lines in file
- `sort f | uniq -c` Count unique lines (sort first!)
- `head -20 / tail -20 f` First / last 20 lines
- `tail -f log.txt` Follow file (live log watching)
- `du -sh */` Disk usage per subdirectory
- `ls -lhS` List files sorted by size
- `cat -n file` Print file with line numbers
- `column -t file.csv` Pretty-print CSV/TSV aligned
- `xargs -I{} cmd {} arg` Run cmd for each stdin line
- `cmd1 && cmd2` cmd2 only if cmd1 succeeds
- `cmd1 || cmd2` cmd2 only if cmd1 fails