Shell Cheat Sheet

Jacob Koziej (EE '25)

File Paths

`/` — root directory
`.` — current working directory
`..` — parent directory of the current working directory
`/absolute/path` — absolute path
`relative/path` — relative path in the current working directory
`./relative/path` — relative path in the current working directory
`../relative/path` — relative path in the parent directory

Tips:

- absolute paths always start with `/`
- you can chain together relative path components with `/`

Command Structure

Navigation

`pwd` — print name of working directory
`ls` — list directory contents
`ls -al` — list all directory contents in the long listing format
`tree` — list contents of directories in a tree-like format
`cd` — change the working directory to `$HOME`
`cd dir/` — change the working directory to `dir`
`cd -` — change the working directory to the previous directory
`clear` — clear the terminal screen

File Manipulation

`touch foo` — create or update timestamps of a file named `foo`
`mkdir dir/` — create a directory named `dir`
`mkdir -p path/to/dir/` — create all parent directories up to `dir`
`rmdir dir/` — remove an empty directory named `dir`
`rm foo` — remove a file named `foo`
`rm -rf dir/` — recursively and forcefully remove `dir`
`cp src dst` — copy `src` to `dst`
`cp -r src/ dst` — recursively copy `src` to `dst`
`mv src dst` — move `src` to `dst`
`mv foo dir/` — move `foo` into `dir`
`ln -s src dst` — create a symbolic link from `src` to `dst`
`readlink dst` — resolve where link `dst` points to
`unlink dst` — remove link `dst`
`cat foo` — print the contents of `foo` to `stdout`
`head foo` — print the first 10 lines of `foo` to `stdout`
`tail foo` — print the last 10 lines of `foo` to `stdout`
`grep text foo` — search `foo` for instances of `text`
`grep -r text dir/` — recursively search files in `dir` for `text`
`wc foo` — get the word count of `foo`
`wc -l foo` — get the line count of `foo`
`sort foo` — sort the lines of `foo`
`sort -r foo` — sort the lines of `foo` in reverse

Tips:

- you **cannot** restore files deleted with `rm`
- both `cp` and `mv` will **overwrite** an existing destination file
- files don't need extension names
- directories don't need a trailing `/` when used as an argument
- everything is *technically* a file in UNIX-like environments

IO Streams

`cat foo > out` — redirect `stdout` of `cat` to file `out`
`cat foo >> out` — append `stdout` of `cat` to file `out`
`cat foo 2> /dev/null` — discard `stderr` of `cat`
`grep text foo | sort` — pipe `stdout` of `grep` to `stdin` of `sort`

Tips:

- you can chain multiple commands together with multiple pipes
- you can pipe into `tee` to view the output of a redirection

File Permissions

```
-rwxr-xr--            |  rwx r-x r--
|\_/\_/\_/            |  ||| ||| |||
| |  |  |             |  421 401 400  <-- dec notation
| |  |  +> other (o)  |  111 101 100  <-- bin notation
| |  |                |  \_/ \_/ \_/  +---------------
| |  +---> group (g)  |   |   |   |   |  r <-> read
| |                   |   v   v   v   |  w <-> write
| +------> user  (u)  |   7   5   4   |  x <-> execute
|                     +---------------+  - <-> denied
+--------> file type (regular file, directory, etc...)
```

`chmod 644 foo` — change `foo`'s permissions to `rw-r--r--`
`chmod +x foo` — set `foo`'s executable bit for everyone
`chmod g-w foo` — deny writes to `foo` by users in `foo`'s group
`chown jacob foo` — change `foo`'s owner to `jacob`
`chgrp wheel foo` — change `foo`'s group to `wheel`
`chown jacob:wheel foo` — set owner to `jacob` and group to `wheel`

Environment Variables

`$HOME` — user home directory
`$PATH` — executable path
`echo $FOO` — print the value of `FOO` to `stdout`
`env` — view current environment variables
`export FOO=bar` — set environment variable `FOO` to `bar`
`export PATH="$HOME/bin:$PATH"` — add `$HOME/bin` to `PATH`
`unset FOO` — unset environment variable `FOO`

Tips:

- Anything inside single quotes is treated as a string literal
- Some environment variables modify program behavior
- A leading `$` expands a shell variable
- Use `${FOO}` when expanding next to trailing letters

Shell Expansions

`~` — expands to the value of `$HOME`
`!!` — previous command
`!string` — last command starting with `string`
`$?` — exit code of the last command
`$(cat foo)` — treat the output of `cat foo` as a string
`*` — expand to all files in the current working directory
`foo*` — expand to all files that start with `foo`
`*.c` — expand to all files that end with `.c`

Reading the Manual

`man man` — read `man`'s manual
`man cat` — read `cat`'s manual
`man stdint.h` — read `stdint.h`'s manual
`man printf` — read the shell's `printf` manual
`man 3 printf` — read libc's `printf()` manual
`man 2 write` — read the system's `write()` manual

Tips:

- When in doubt, read the `man` page
- If there's no `man` page, check a command's `-h`/`--help` flag
- Adding a number to `man` specifies a different section

External Resources

`explainshell.com` — explain shell commands
`missing.csail.mit.edu` — MIT's Missing Semester of CS
`devhints.io/bash` — Bash scripting cheatsheet
`github.com/dylanaraps/pure-sh-bible` — pure sh bible

Spot any errors or have a suggestion?
Shoot me an email at <jacobkoziej@gmail.com> or <jacob.koziej@cooper.edu>
v0.1.1 — 2022-07-22