The ultimate programming course cheat sheet for UNIX-like operating system newbies

# Directories and files

## Concepts

- Files are organised in a directory tree/hierarchy
- Everything is a file (e.g. keyboard, printers, ...)
- Each process has access to the files *stdin* (input), *stdout* (buffered output), *stderr* (unbuffered output)
- Each process operates in a *working directory*
- Each user has a *home directory*

## Paths

*Path* = Identifier for the location of file/directory

- Paths consists of a parent directory list + file/directory
- Files and directories are separated by a '/'
- Directory paths may contain a trailing '/'

*Absolute path* = Full location (first character = '/')
*Relative path* = Relative location (first character ≠ '/')

| | |
|---|---|
| `.` | Path to the directory itself |
| `..` | Path to the parent directory |
| `/usr/bin/ls` | Example for an absolute file path |
| `/home/foo/` | Example for an absolute directory path |
| `./a.out` | Example for a relative file path |

## File system hierarchy

| | |
|---|---|
| `/` | Root directory |
| `/bin` | Essential command executables |
| `/dev` | Device files |
| `/etc` | System-wide configuration files |
| `/opt` | Manually added software |
| `/sbin` | Essential administrative executables |
| `/tmp` | Temporary files |
| `/usr` | System resources for users |
| `/usr/bin` | Command executables |
| `/usr/local` | Site-local data |
| `/usr/sbin` | Administrative executables |
| `/var` | Variable files |

**`man hier`** (or **`man file-hierarchy`** on recent Linux distributions) to get a more detailed overview

# Terminal (emulator)

*Text terminal* = Computer interface for text entry/display
*Terminal emulator* = Application that emulates a *text terminal* in a graphical environment
Examples for terminal emulators: `xterm`, `urxvt`, `guake`

## Opening a terminal

| | |
|---|---|
| Unity/GNOME | Ctrl + Alt + T |
| Mac OS | Cmd + [ ] → "terminal" → ↵ |
| Bash on Windows | Win + R → "bash" → ↵ |

# Shell

*Unix shell* = User interface that accepts commands to operate a computer
**`man intro`** to get an introduction into basic shell usage

Examples for shell programs: `sh`, `bash`, `zsh`, `fish`, `ksh`

## Prompt

*Prompt* = Text sequence that precedes each line that prompts the user to enter a command

`[foo@bar /var/www]$` Example prompt in bash

⇒ *User* `foo` is operating in the *working directory* `/var/www` at the computer with the *host name* `bar`

## Line editing

| | |
|---|---|
| Ctrl + A / E | Go to beginning/end of the line |
| Ctrl + U / K | Erase to beginning/end of the line |
| Alt + B / F | Go one word forward/backward in line |
| Ctrl + W | Erase to beginning of current word |
| Ctrl + C | Cancel the current command line |

## Metacharacters

The following characters have special meaning and sometimes they can't be used directly as arguments/words:
`| & ; < > ( ) $ ` \" ' * ? [ ] # ~ = % ! { }` [ ] ⇤⇥ ↵

Their special meaning can be disabled:

| | |
|---|---|
| `\` | Preserves the literal value of the following character |
| `' '` | Preserves the literal values of enquoted characters |
| `" "` | Like `' '` but characters `` ` `` `$` `\` retain their meaning |

## Expansions

| | |
|---|---|
| `~` | *Home directory* of the current user |
| `*` | Matches any character sequence |
| `?` | Matches a single character |
| `[...]` | Matches any one of the enclosed characters |
| `${var}` | Value of the environment variable *var* |
| `$(cmd)` | Output of *cmd* |
| `$((expr))` | Result of the mathematical expression *expr* |

## Shell utilities

| | |
|---|---|
| **`apropos`** `text` | Searches the manual pages for *text* |
| **`cat`** `file` | Prints the contents of *file* |
| **`cd`** `dir` | Changes the *working directory* to *dir* |
| **`chmod`** `prm file` | Changes permissions of *file* to *prm* |
| **`cp`** `src dst` | Copies the file/directory *src* to *dst* |
| **`echo`** `text` | Prints *text* |
| **`file`** `file` | Determines the file type of *file* |
| **`find`** `dir expr` | Finds files in *dir* that match *expr* |
| **`grep`** `expr file` | Searches for pattern *expr* in *file* |
| **`ls`** `dir` | List the entries in the directory *dir* |
| **`man`** `cmd` | Displays the manual for *cmd* |
| **`mkdir`** `dir` | Creates the directory *dir* |
| **`mv`** `src dst` | Moves/renames *src* to *dst* |
| **`pwd`** | Prints the current *working directory* |
| **`rm`** `file` | Removes the file *file* |
| **`sort`** | Sorts lines of text from input |
| **`touch`** `file` | Creates the empty file *file* |

## Input output redirection

| | |
|---|---|
| ***cmd1*** **`\|`** ***cmd2*** | Runs *cmd1* and *cmd2* and redirects the output of *cmd1* to the input of *cmd2* |
| ***cmd*** **`>`** `file` | Runs *cmd* and redirects output to *file*, content of *file* is overwritten |
| ***cmd*** **`>>`** `file` | Like `>>` but appends output to *file* |
| ***cmd*** **`<`** `file` | Runs *cmd* and redirects *file* to its input |
| ***cmd*** **`<<<`** `text` | Runs *cmd* with input *text* |

## Job control

*Job* = Shell command and its associated process(es)

- Each job has a job id and corresponding process ids
- Jobs can run in the foreground or in the background
- The execution of a job can be temporarily suspended

| | |
|---|---|
| ***cmd*** `&` | Starts *cmd* as background job (id is printed) |
| **`fg`** `%jid` | Puts job with id *jid* in foreground |
| **`bg`** `%jid` | Continues job with id *jid* in background |
| **`jobs`** | Prints ids of jobs in current shell |
| **`kill`** `pid` | Terminates process with the PID *pid* |
| **`ps`** | Prints PIDs of processes in current terminal |
| Ctrl + S / Q | Suspends/resumes active job |
| Ctrl + Z | Puts job to background and suspends it |
| Ctrl + C | Aborts job (most of the times) |
| Ctrl + D | Sends an EOF character |