# Terminal

•••

07-131 Great Practical Ideas in CS

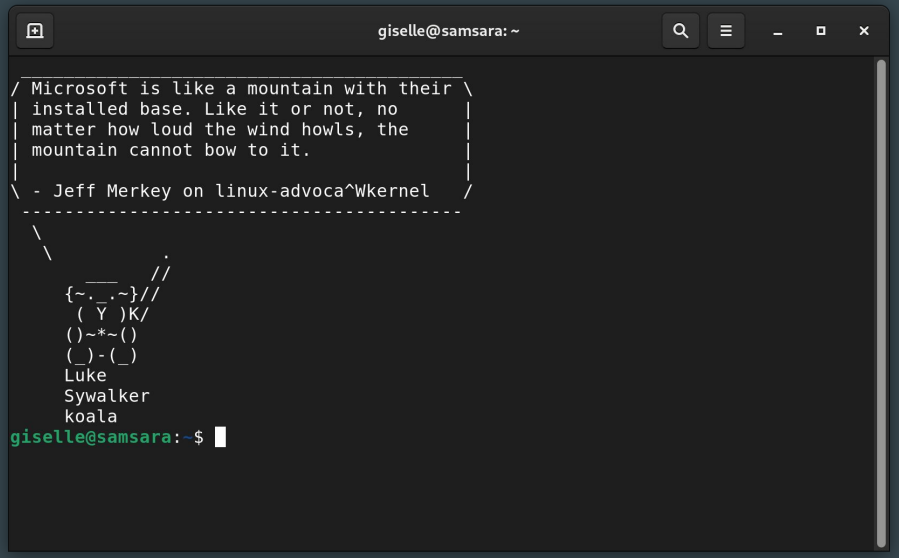
Instructors: Eduardo Feo-Flushing & Giselle Reis

Course website: https://web2.qatar.cmu.edu/cs/07131/

# Terminal and Shell

# What is a terminal?

- A program that provides a simple command-line interface with a computer.
- The (usually) black screen hackers are using in movies.
- What your computer defaults to when the GUI breaks.
- Many different terminals* exist:
  - gnome-terminal
  - terminator
  - xterm
  - ...




**Loose analogy:** If the computer is python, then the terminal is the IDE you use to program.

* Also called terminal emulators.

# What is a shell?

- A language that provides a simple command-line interface with a computer.
- Includes the language and the interpreter that talks to the computer via a terminal.
- Usually a full on programming language.
- Many different shells exist:
  - bash
  - zsh
  - tcsh
  - ...

**Loose analogy:** If the computer is python, the terminal is the IDE you use to program, then the shell is python's interpreter

```
giselle@samsara:~$ fortune | cowsay -f "$(cowsay -l | sed '1d' | tr ' ' '\n' | sort -R | head -1)"
 _____________________________________
< If you fail to plan, plan to fail. >
 -------------------------------------
         \     ,-.       .-,
          \    |-.\ __ /.-|
           \   \   `   `   /
                /_       _ \
               <  _`q  p _  >
               <.._=/  \=_. >
                  {`\()/`}`\
                  {      }  \
                  |{    }    \
                  \ '--'   .- \
                  |-       /    \
                  | | | | |     ;
                  | | |.;,,..__ |
                .-"";`          |
               /    |           /
               `-../___,..---'`
giselle@samsara:~$
```

Bash shell command to print a character and a quote.

# Warning!

Terminal, terminal emulator, shell, console, etc are sometimes used interchangeably when people do not know or care about the difference.

# Do I really need to?

- Using the terminal is more efficient than pointing/clicking once you become familiar with the commands.
- **Take time to learn once, and save time for the rest of your life.**
- You will need it for CMU courses (15-122, 15,150, 15-213, etc...).
- It is the only way to interact with the university (and many other) servers.

# Shell Commands

- Mastering shell commands makes using the terminal smooth and quick.
- Command structure:

```
command <flags/options> <arguments>
```

- Examples:
  - `ls` [lists files in current directory, no options, no args]
  - `ls -a` [lists all files, including hidden, in current directory, one option, no args]
  - `ls -a Desktop` [lists all files in Desktop, one option, one arg]
- Use tab to autocomplete!

# Man Pages

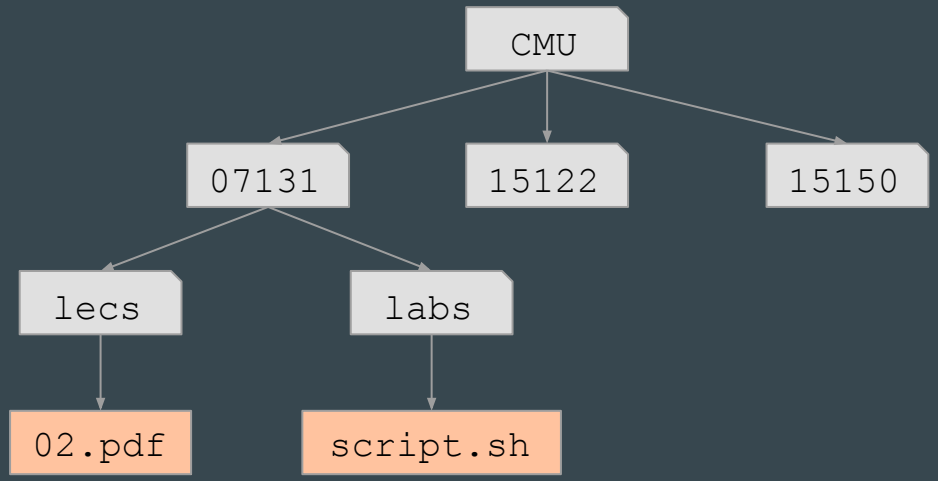
- Most (all?) commands have man(ual) pages explaining usage, options, etc.
- Sometimes this is enough and saves a search on google.
- Access them with: `man command`
- Example: `man ls`

```
LS(1)                                User Commands                                LS(1)

NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List information about the FILEs (the current directory by default).  Sort en-
       tries alphabetically if none of -cftuvSUX nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              with  -l,   scale   sizes   by   SIZE   when   printing   them;   e.g.,
              '--block-size=M'; see SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~
 Manual page ls(1) line 1 (press h for help or q to quit)
```

# File System

# File System

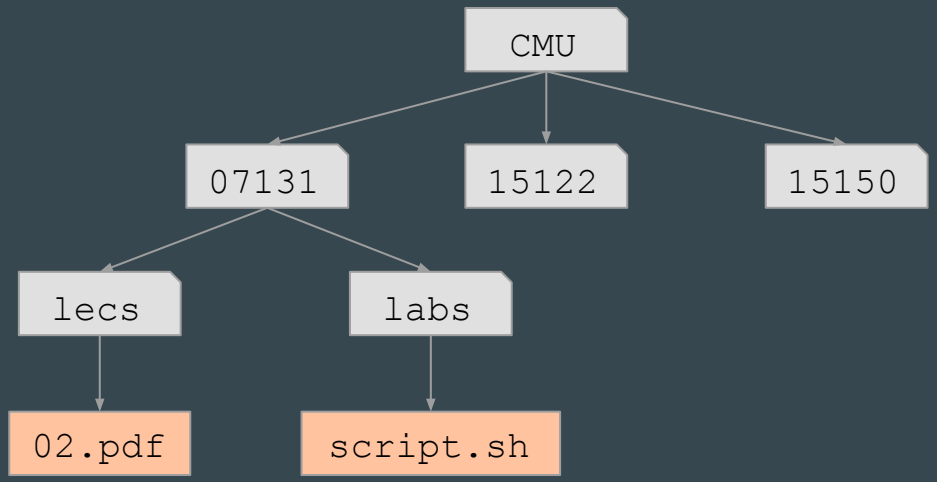
The file system is a tree: folders are nodes, files are leaves.



# Paths

In Unix, paths are separated by a forward slash. For example:
`CMU/07131/lecs/02.pdf` or `CMU/15122/`

Absolute paths (starting at the root of the file system) begin with `/`
`/home/greis/CMU/15150/`




Relative paths may use: `.` for current folder and `..` for previous folder.
For example: `../../` means two directories above the current one.

# Navigating Paths

- Done mostly using the `cd` command (for "change directory"):
  `cd path/of/goal`
- Path may be absolute or relative
- You may specify the whole path at once.
  Instead of `cd CMU`, enter, `cd 07131`, enter, `cd labs`, enter;
  type `cd CMU/07131/labs`, enter. Much quicker (specially using tab)!
- Remember, `..` can be used in paths: `cd ..` goes back to the previous directory.
- `~` denotes your home directory, so `cd ~/www` takes you to the www directory in your home folder.
- `cd` always takes you back home.

# Finding Yourself

- Usually indicated next to your username in the shell:
  `andrewID@unix-01:~/CMU/07131/lecs$`
  (can be customized in the .bashrc file)
- Got lost? `pwd` to the rescue!
- `ls` lists all files and directories <u>in</u> the current directory.
- `tree` lists all files and directories <u>under</u> the current directory in a structured way:

```
giselle@samsara:~/CMU$ tree
.
├── 07131
│   ├── labs
│   │   └── script.sh
│   └── lecs
│       └── 02.pdf
├── 15122
└── 15150

5 directories, 2 files
```

Tip: To list the files and directories only up to level X, use: `tree -L X`

# Interacting with Files and Directories

| | File | Directory | Command args |
|---|---|---|---|
| create | `touch` | `mkdir` | `<name>` |
| copy | `cp` | `cp -r` | `<from> <to>` |
| rename | `mv` | `mv` | `<from> <to>` |
| delete | `rm` | `rm -r` | `<target>` |

**Attention!!** There is no undo from `rm`'ing files. Use it wisely.

# Hidden Files

- Hidden files and directories start with `.`
  For example: `.bashrc` (bash's hidden configuration file) or `.cache` (a local hidden cache directory)
- To see them all, add the flag `-a` to `ls` or `tree`.
- They are hidden because we typically do not need to use them often.

# Executing Files

An executable file can be run by invoking it with `./` as a prefix.

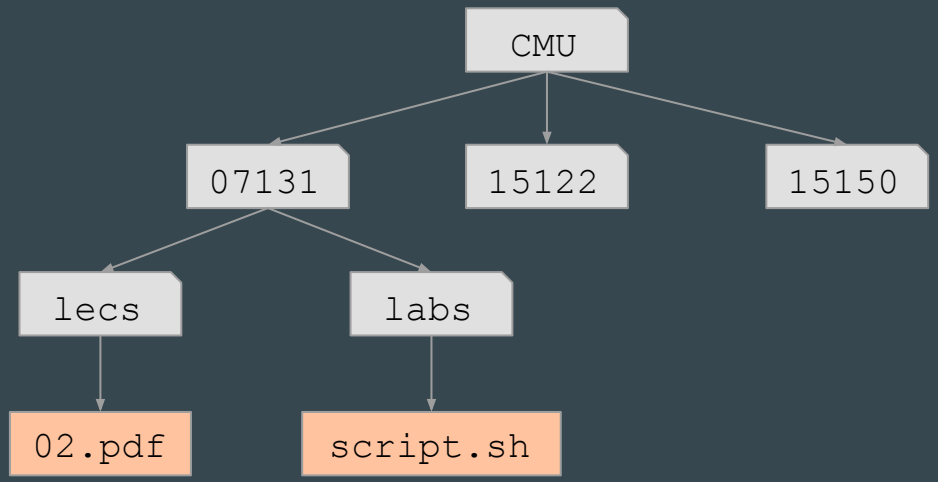
Suppose script.sh is executable. Then

```
./script.sh
```

will execute it if you are in the `CMU/07131/labs` directory.

Or you can run it from anywhere by using its path. For example, if we are in the `CMU` directory:

```
./07131/labs/script.sh
```



# SSH

# What is SSH?

- `ssh` stands for *secure shell*.
- Used to access the shell in another machine safely.
- This is how you access CMU's unix servers.
- Allows you to access and run commands in another computer remotely.
- To connect to CMU's unix servers (aka Andrew machines) from your computer, open a terminal and type:
  `ssh andrewID@unix.qatar.cmu.edu`
- Then type your andrewID password.

# Copying files across computers

- Use `scp` to copy files to/from machines you have ssh access.
- Examples:
  - Copy local `hw01.pdf` to your home folder in the unix server:
    `scp hw01.pdf andrewID@unix.qatar.cmu.edu :~/`
  - Copy remote `example.sh` file from your private directory in the unix server to your local machine:
    `scp andrewID@unix.qatar.cmu.edu :~/private/example.sh .`
- Keeping files in sync across different machines using scp is generally a bad idea...
- Better choose one machine to work, and store and edit files in the same place.

# It's trainerlab time!