CIS Minicourses Shared Lecture

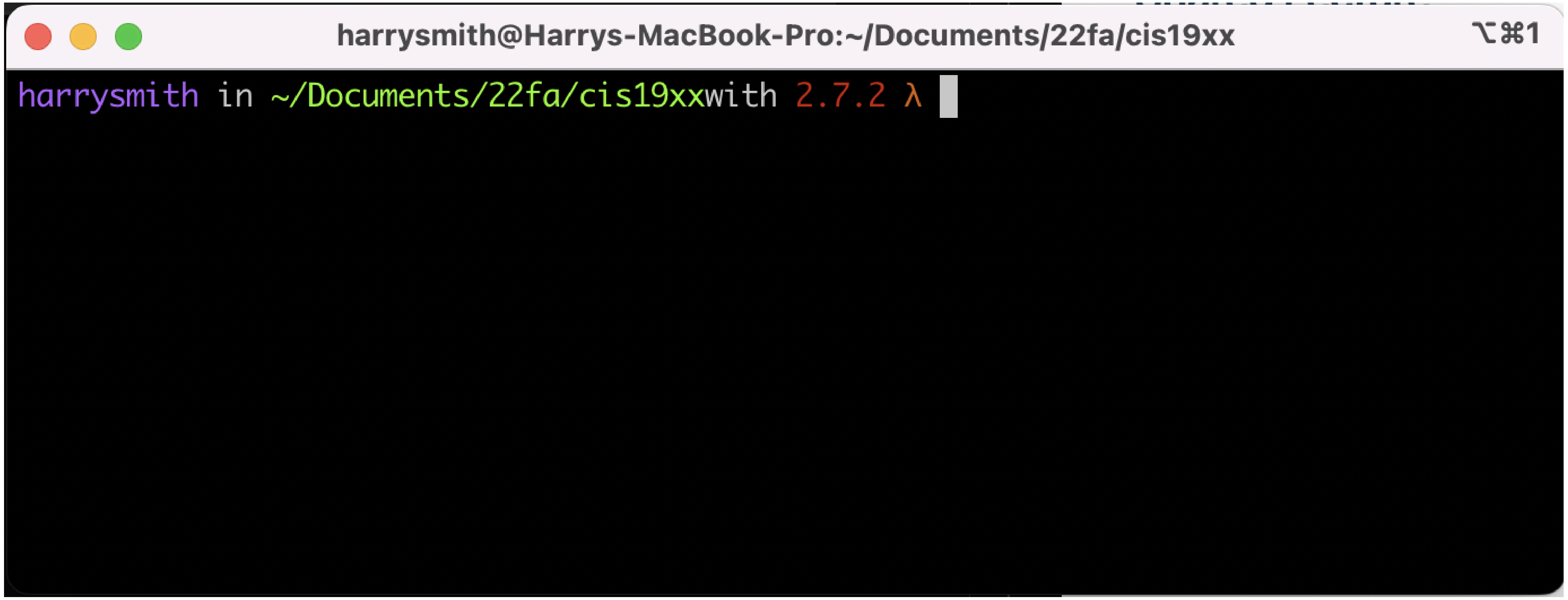
1. Unix Command Line

Shared Lecture

- Tuesday 5.15-6.15 (now) in Towne 100 (here) for the next 3 weeks
- Taught by me (Harry Smith, sharry@seas.upenn.edu)
- Swapneel Sheth (swapneel@seas.upenn.edu) is our other Faculty Coordinator for the 18xx/19xx courses
- The actual material (Go, C++, DevOps) is taught in the recitation section

Unix Command Line

- A simple, text-based interface to the computer


harrysmith@Harrys-MacBook-Pro:~/Documents/22fa/cis19xx ⌥⌘1

```
harrysmith in ~/Documents/22fa/cis19xxwith 2.7.2 λ
```

- One of many ways to interact with a computer; enduring because of how easy it is to work with text.

Why no graphics?

- Graphics are:
  - intuitive, easy to "feel around"
  - hard to automate
  - tedious use for repetitive tasks
- CLIs are:
  - a bit daunting
  - easy to automate
  - capable of describing big tasks in one short line

Shell vs. Command Line vs. `bash` vs. ...

- The **shell** is the type of program that a user uses to interact with the computer's filesystem.
- The **command line** is the interactive text input in the shell where the user can place their commands.
- **bash** is a common shell program; **zsh** is another popular one that's default on MacOS.

Basic Interaction

- Text appears at the bottom of the screen
- up/down arrows scroll through command history
- tab autocompletes when possible
    - if `g` could complete to `gala` or `granny-smith`, then pressing tab twice would show both options

Commands, pt 1

- `clear` - clear the screen
- `ls` - list contents of the current directory
- `pwd` - **p**rint **w**orking **d**irectory

Linux Filesystem

- It's a tree, where each node is a directory
- The root of the tree is the `/` directory
  - the `/` character is the "separator" between the directory names
- The **absolute** name of any directory—the name that unambiguously describes its location—is generated by following the path from the root to that directory, adding `/` between directory names.

Commands, pt 2

- `cd` - **c**hange **d**irectory
  - `cd ..` - go up one directory
  - `cd dir_name` - go into directory called `dir_name`

Absolute vs. Relative Paths

- As we saw, absolute paths start from the root directory `/`
  - e.g. `/home/sharry`
- **Relative paths** start from the current
  - if `pwd` ➡️ `/home1/c/cis19x/tmp/linux-basics/`, and we `cd apples`, then we are now in `/home1/c/cis19x/tmp/linux-basics/apples`
  - `apples` was the relative path for `/home1/c/cis19x/tmp/linux-basics/apples`

Path exercises

- For a given source directory, what is its absolute path?
- For another given target directory, what is its absolute path?
- What is the relative path of the target starting from the source?

Commands, pt 3

- `ls -l` - list contents of the current directory in long format
- `ls -a` - list contents of the current directory, including hidden files (those starting with `.`)
- `ls -s` - list contents of the current directory, sorted by size

Flags

- Command line arguments starting with `-`, are called **flags**
- They change the behavior of the command
    - order invariant
    - can be combined
        - try `ls -las` and `ls -sal`
- Can also be combined with the actual argument
    - e.g. `ls -l /home/sharry`

`man`

- There's a lot to remember, would be nice to have some kind of `man`ual...
- Keep in mind:
  - takes over the shell, `q` to exit
  - might have to "disambiguate" between different implementations

Commands, pt 4

- `mkdir <dir_name>` - create a directory called `<dir_name>`
- `rmdir <dir_name>` - remove a directory called `<dir_name>`
    - only works on empty directories!
- `cat <file_name>` - print the contents of `<file_name>`
- Editing files:
    - `pico <file_name>` - edit `<file_name>` in the pico editor
        - `ctrl-o` to save, `ctrl-x` to exit, more at the bottom
    - `emacs` or `vim` - other popular (complicated) editors

`ssh`

- Secure command line remote access to another computer's shell
- `ssh <user>@<host>` - connect to `<host>` as `<user>`
  - e.g. `ssh sharry@eniac.seas.upenn.edu`
  - usually prompted for password, but can set up a key to save on typing

Commands, pt 5

- `mv <filename> <new_name>` - rename `<filename>` to `<new_name>`
- `mv <filename> <dir_name>` - move `<filename>` to `<dir_name>`
  - this and previous can be used to move directories, too
- `cp <filename> <new_name>` - copy `<filename>` to `<new_name>`
- `cp -r <dir_name> <new_name>` - copy `<dir_name>` and all its contents to `<new_name>`
- `rm <filename>` - remove `<filename>`
  - `rm -rf <dir_name>` - remove `<dir_name>` and all its contents
  - ⚠️ these removals are immediate and permanent! ⚠️

`scp`

- like `cp`, but **s**ecure over internet connections
- useful for transferring files from your laptop to eniac
  - `scp file.txt username@to_host:/remote/directory/` is local to remote
  - `scp file.txt username@to_host:/remote/directory/` is remote to local
- to quickly pop files off eniac, you can use eniac's built-in `mail` command, e.g. `mail -a attachment.txt sharry@seas.upenn.edu`