CSCI 3753: Operating Systems Fall 2024

Dylan Sain

Department of Computer Science

University of Colorado Boulder

Welcome to Operating Systems Recitation

- **Dylan Sain**
  - CS Masters student

- Email: Dylan.Sain@colorado.edu

Welcome to Operating Systems Recitation

- Office hours:
  - **Wed 10:00am – 11:00am and Thurs 12:30 PM – 2:30 PM**
  - Zoom link: https://cuboulder.zoom.us/j/2163683260
  - CSEL on Thursdays (message me on slack if you cannot find me)
  - Available other times as needed, send me an email or message on slack to make an appointment

Administration

- What to do when you have a question?
    1. Post to the Slack
    2. Email or message the TA responsible for your recitation

- What should NOT be posted publicly on Slack?
    - Source code
    - Direct answers to the questions in problem sets or quizzes

- All quiz, PA submissions, and exams should be done from **Canvas**.

Quiz Logistics

- Completion time: 8am – 11:59pm every Friday
- Time limit: 10 minutes
- Attempts: 1
- Quiz contents:
    - Reading chapters requested on Canvas within the week
    - Lectures taught in Class within the week

Recitation Logistics

- Not taking attendance

- Deep dive into topics covered in lectures
- Questions regarding the past and current PA
- Questions on last weeks quizzes
- 1 on 1 time with a TA (Me!)
- Small activities

PA0: Offline Environment vs Cloud VM

	Offline VM	**Cloud VM**
Pros	- Flexibility to do your work at anywhere and anytime - Fast interaction speed to the interface	- No worries at required large storage - Instructors and TAs can help debug hardware-related issues
Cons	- Big constraint on computer physical configuration + 4 processor cores + 8GB of RAM + 64GB+ to do kernel compilation exercises (or 32GB of free HD space)	- Require Internet connection - Slow interaction speed to the interface

Linux Shell Environment

Bash Basics

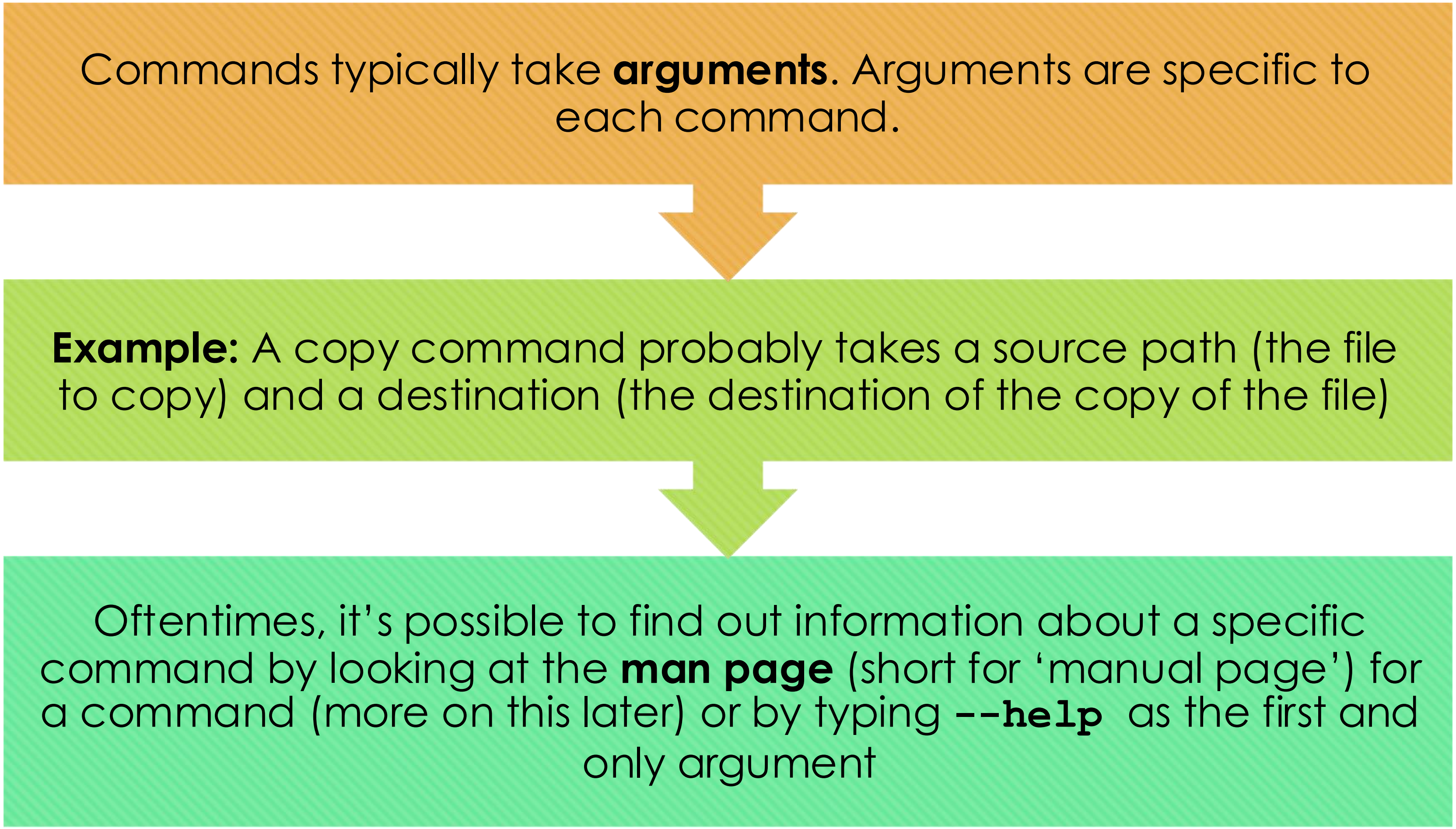
1. Introductory Vocabulary (shell and commands)
2. File System
3. Permissions and Groups
4. Files
5. Combining Commands
6. Tips

What is a shell?

- A shell is an interface one can use to instruct a computer to take some action

- Instructions are typically given in the form of a command. Commands can be:
  - A built-in command is specific to the shell itself
  - An external command runs an external executable file. When you call an external command, the kernel or operating system will run the executable and it becomes a process.

Commands



Commands typically take **arguments**. Arguments are specific to each command.

Example: A copy command probably takes a source path (the file to copy) and a destination (the destination of the copy of the file)

Oftentimes, it's possible to find out information about a specific command by looking at the **man page** (short for 'manual page') for a command (more on this later) or by typing **`--help`** as the first and only argument

Anatomy of a Shell

```
user@host:/home$ command arg1 arg2 arg3
```

Anatomy of a Shell

```
user@host:/home$ command arg1 arg2 arg3
```

- A command prompt
- Shown when opening a shell
- May contain some useful information such as
  - Your username (`user` in this example)
  - Hostname (`host` in example), and
  - Current directory (`/home` in example)

Anatomy of a Shell

```
user@host:/home$ command arg1 arg2 arg3
```

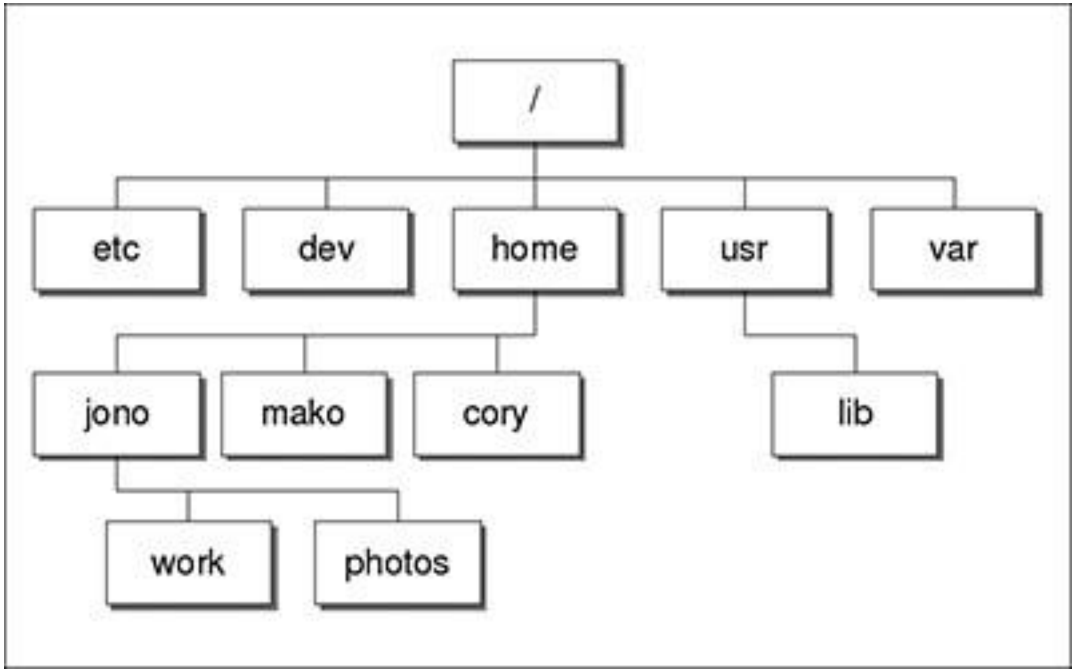
The command that will be run when you select the ‘return’ or ‘enter’ key

Exercise 1: Practice Commands

- `whoami` : your user id
- `pwd` : your present working directory
- `uname -a` : information about the system
- `echo hello` : prints a string to standard out

File System

- The Linux file system has
  - A single root /
  - Several common directories underneath



Relative vs. Absolute Paths

- An absolute path is a path that is relative to the root (`/`) of the file system.
  - Example: `/tmp/this/path/is/absolute.txt`
- A relative path is the path that is relative to the current directory (pwd)
  - Example: `this/path/is/relative.txt`
  - Equivalent to `<pwd>/this/path/is/relative.txt`
- The `/` at the front of the path lets you know which kind of path that it is.

Common File System Commands

`rm`	remove
`cp`	copy
`mv`	move
`mkdir`	create a directory
`rmdir`	remove a directory
`ls`	list file system
`pwd`	print the current working directory
`cd`	change the working directory
`touch`	updates the modified timestamp of a file (if it exists) or creates a new empty file

`ls` has a ton of useful options

`ls -a` : show all (including dot files)

`ls -l` : long listing

`ls -al` : long listing of all files

`ls -lh` : long listing in human readable format

`ls -t` : sort based on time

`ls -tr` : sort based on time reverse

`ls -altrh` : string them all together

Exercise 2: File System Commands

Test out some commands you've learned:

- □ Create an empty file in `/tmp` called `example`
- □ List `/tmp` to see your file. Try out the different arguments you can give to `ls` to get a more detailed listing.
- □ Copy `example` and name the copy `example2`
- □ Rename `example2` to `example3`
- □ Delete `example3`

Permissions and Groups

- Permissions exist to control access to files and directories. `ls -l` will show the permissions of both files and directories:

```
-rw-rw-r-- 1 user user 6 Sep 22 00:33 test.txt
```

Permissions

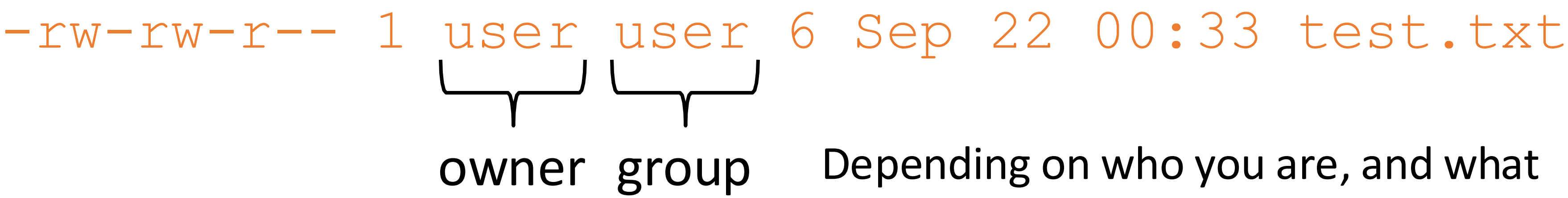
Permissions are displayed with three groups of three letters:

- `r` for read
- `w` for write
- `x` for execute.

The first group is for user (owner), the second is for group, and the last is for everyone else.

Permissions and Groups

- Permissions exist to control access to files and directories. `ls -l` will show the permissions of both files and directories:



```
-rw-rw-r-- 1 user user 6 Sep 22 00:33 test.txt
```

Depending on who you are, and what groups you are in (displayed with the `groups` command), compared to those of the file, you can tell whether you have read, write, and/or execute permission for a given file.

Permissions and Groups Commands

`groups`	Lists the groups you (or another user) is a member of
`chgrp`	Changes the group ownership of a file
`chmod`	Changes the file mode bits

Exercise 3: Setting Permissions

- The syntax and format of mode bits can be hard to understand. Google can be a lot of help with `chmod`.
- `touch /tmp/file1`
- Run `ls -l /tmp/file1` to check what the mode bits are for file1
- Run `chmod 755 /tmp/file1`
- Run `ls -l /tmp/file1` to check what the mode bits are for file1
- Run `chmod 777 /tmp/file1`
- Run `ls -l /tmp/file1` to check what the mode bits are for file1
- Run `chmod 640 /tmp/file1`
- Run `ls -l /tmp/file1` to check what the mode bits are for file1

File Commands

- `head` shows top 10 lines
  - `head -n` shows top N lines
- `tail` shows bottom 10 lines
  - `tail -f` follows as the file grows. GREAT for troubleshooting.
- `cat <filepath>` dumps the contents of a file to stdout
- `less <filepath>` is a minimal way to show some portion of a file.
- `more <filepath>` is also a command. This is a joke on the phrase, “less is more”

File Extensions

- Note: Unlike Windows, Linux is much more casual about file extensions. You don’t even need to specify one!

- The file command can help you determine the type of file something is, based on its content (specifically header data).

Exercise 4: File Commands

Test out some commands you've learned:

- □ View the first 15 lines in the word list found at `/usr/share/dict/words`
- □ View the last 30 lines in the word list found at `/usr/share/dict/words`
- □ View the entire word list using `less`. Type `/hello` to find all words that contain 'hello' in the list. Quit `less` with `q`
- □ Run `file /usr/share/dict/words`. You will realize that the file is not what you thought it was – google "linux symbolic link" to find out more information. Now run `file <new_path>` until you find the actual location of the file.

Combining Commands & More

- Some of the commands we've covered are very powerful – but it can be even more powerful to combine individual commands. Some commands come with build-in utilities to try to combine them.

- For instance, `find` has a `-exec` option which allows you to run a command (or bash script) on each of the files it find.
  - It's often useful to find files and then grep to find certain strings in those files.

Combining Commands & More

- For example, if you want to find files called `fs.h` on your system, and then search that file to find the string `file_operations` and print the line number of that value you could run:

```
find / -type f -name fs.h -exec grep -ni file_operations "{}" \;
```

| operator

- An example of using the | operator is:

```
cat /usr/share/dict/words | grep hi | wc -l
```

  - The first section (`cat`) will print the contents of `/usr/share/dict/words` to stdout.
  - The first | will send that output to the grep command, which will search the text for the string ‘hi’.
  - The last pipe will send all the words that contain ‘hi’ to `wc –l`, which will count the number of lines in the output.

- The end result is a command of how many words contain the substring ‘hi’ in the dictionary.

`&&`, `||`, and `;` operator

- An example of chaining commands is:

```
cd /bin; ls -lah
```

- This allows you to change directories and do a listing with a single command. Of note, the second command will always be run, even if the first one fails. You could also run:

```
cd /bin && ls -lah
```

- This command will do the same thing. However, the second command will only run if the first is successful. As a note, `||` does the opposite thing as `&&`: with `||` the second command will only run if the first command is unsuccessful.

Exercise 8: Chaining & Commanding

- Count the number of files in `/etc`, use `| wc -l`
- Get a list of all files in `/bin` with `bash` in the name
- Get a count of the above

Tips

- You can tab complete files names when using many commands such as `cd`, `ls`, and more

- You can view command history (commands you've previously run) using the up and down arrow keys.

- To compile a source code, run `gcc <file_path>` or `gcc <file_path> -o <output_name>`

- To execute a compiled program (i.e., the binary code), run `./<binary_file_path>`