UCLA CS35L

Week 1

Tuesday

Course Intro

- TA – Daniel Meirovitch
  - Daniel.Meirovitch@ucla.edu
  - Office hours – https://ucla.zoom.us/j/6595885664
    - Wednesday 2 – 3 PM
    - Friday 11 AM – 12 PM
    - First 30 minutes for general questions, Next 30 Minutes for any debugging that is more 1-on-1
- Course Website - https://web.cs.ucla.edu/classes/spring20/cs35L/
- Questionly (Used instead of Piazza) – Access through CCLE Apps on Left
- Anonymous feedback for Daniel - https://forms.gle/tZwuMbALe825DBVn8

Course Logistics

- All lectures will be on Zoom
- For now – Recordings will be taken of each lecture and posted that night
  - Just listening to a 2 hr recorded lecture is not a good learning experience…
  - So I recommend coming to class where we can interact together

- Generally… Lecture's will be 1 hr. And then then remaining hour will be more like in-class Office Hours. You can work together and ask me questions about anything.
  - A few lectures will be longer just due to the material

Course Assignments

- Grading
  - Assignments 50%
    - 9 regular assignments + 1 Presentation/Report
    - Other Assignments may be updated as we go, so I'd wait before starting any of those
    - 2^n deduction for being late. (-1 for the first day, -2 for the second day, -4 for the third…)
    - Sign-up as a team of 2 for a topic and time-slot for presentation
      - https://docs.google.com/spreadsheets/d/1Dsim9vZGbQ5lKoUpp_f-l-6uR1whweAA6QwVzspjZuM/edit?usp=sharing
      - Or find link under Week 10 CCLE
  - Final Exam 50%
    - TBD
    - Will not necessarily be the same format as last quarter…

Course Goals

- The best match for this class is MIT's "Missing Semester of Your CS Education"
  - https://missing.csail.mit.edu/
- Main goal is to become more comfortable with software engineering tools and techniques that are important for future CS classes and industry.
- My personal takeaway – At the end of this course you want to become comfortable in a Unix and command-line based environment.

In Full Transparency…

- There are 3 main issues that are usually brought up in feedback about this course
  - Too many people cheat
  - It’s too hard and too much work
  - The taught material is too variable between TAs

Cheating

- You must submit your own work
  - That means **you** are the one responsible for writing it
  - It is ok and encouraged to collaborate when it comes to sharing ideas.
  - Anytime you are directly copying code is the problem
- We absolutely do check for cheating, and it’s not fun when it has to go to the Dean

Workload

- Bad News - this class is a lot of work.
  - A lot of students have told me they spend about 15 hours a week on it.
  - If you really know your stuff, it can probably be done in 5-6 hours a week
- Good News – we (as TAs) are here to help you learn, especially in this crazy quarter
  - Trying to update some weeks to make them more streamlined
  - Together with the class, I will go over strategies for starting each homework
  - We will typically share hints to help with each assignment
- It will still be a lot of work. But doing the assignments honestly will truly help you learn about the subject.

TA Variability

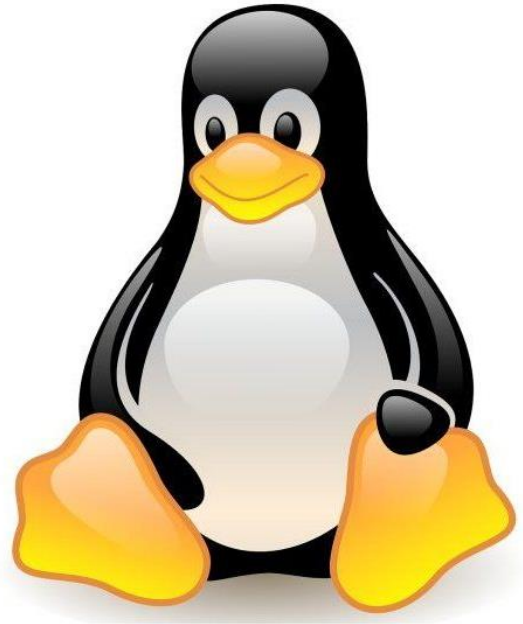
- An unfortunate reality of the structure of this class.
  - Professor Eggert himself is lecturing a version of this class CS 97 as a test
- What we are trying to do on our end:
  - We share topic lists, materials, etc to keep the overall lessons similar
  - Each homework assignment has a single TA in charge of it (except for 2 and 10). So your experience doing the homework will be consistent
- What you can do to help as well:
  - Please ask questions! We are always happy to go in more detail if you feel something is unclear
  - If you think another TA covered something that I missed, please let me know!

Questions on Course

Student Introductions

- I’ll go through the participant pane on Zoom and call each person. Then please say:
  - Your preferred name
  - Year and Major

Intro to Linux



What is Linux

- Linux is a kernel, the core of an operating system (OS)
  - Allocates system resources to running programs
  - Connects hardware to software
- GNU software + Linux kernel → GNU/Linux OS
- Many different Linux distributions
  - Popular ones include Ubuntu, Arch, Red Hat

Why do we care about Linux

- Linux is in the Unix family of operating systems. One of the first major OS’s to become popular.
  - Includes MacOS, Solaris, BSD, etc
- Linux specifically is free and open source
- Linux powers 96.3% of the top 1 million web servers

How do we interact with Linux

- Most common way via Shell – a Command Line Interface (CLI)
  - Gives pure control and allows scripting
  - Faster once you get used to the commands
  - Note there are different shell types. Bash vs Zsh etc
  - What we will primarily use in this class

- Some distributions also come with a desktop (GUI)
  - Easier for multi-tasking and certain applications
  - Limited for common developer and engineering activities

Connecting to a Linux Server

- Prereq for this class
  - Need a SEASnet account (username/password)
    - https://www.seas.ucla.edu/acctapp/
  - Unique this quarter, the school VPN
    - https://www.it.ucla.edu/it-support-center/services/virtual-private-network-vpn-clients

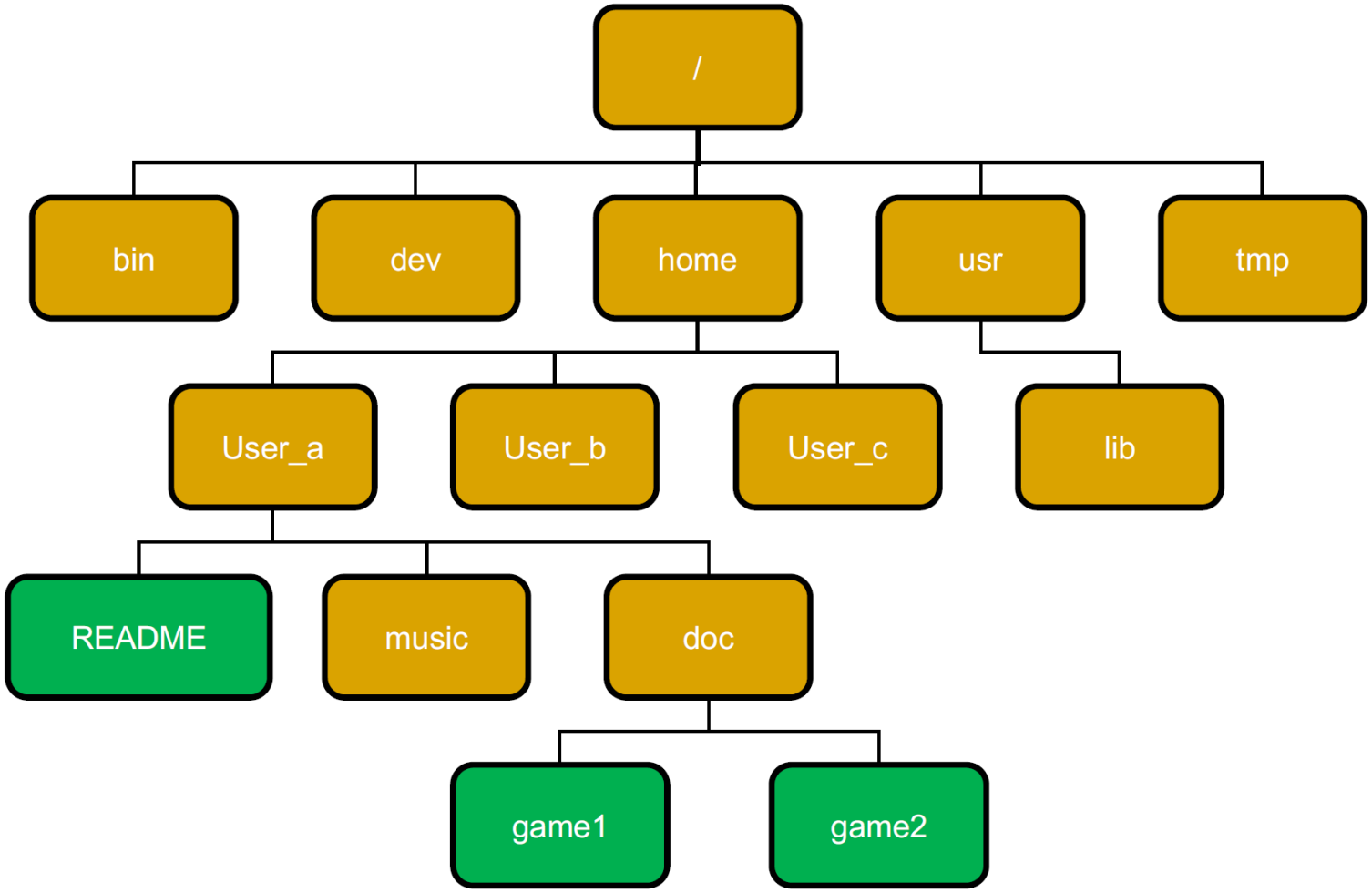
Logging into to the School Linux Server

- Login to the VPN with your UCLA username/password
- MacOS/Linux
  - Use terminal and type '*ssh SEASNETuser@lnxsrv10.seas.ucla.edu*'
    - Use your own personal SEASNET username in the command above
  - Enter SEASNET password when prompted
- Windows
  - Download a ssh client (PuTTY is popular).
  - When open, specify the Host Name as lnxsrv10.seas.ucla.edu, port 22, and connection type as SSH
  - Enter SEASNET username and password when prompted
- NOTE – in this class we will use lnxsrv06, 07, 09, and 10

Linux and Files/Processes

- Everything is a file or a process
- Really… Everything is a file. Directories are files, applications are files, etc
- A Process is a running instance of a program.
  - So you execute some command on a file, and a process is allocated and managed by the OS

File System Layout



Common Directories

/		Root directory
/bin		user binaries (ls, pwd, etc)
/boot		static boot files
/dev		device files
/etc		configuration files
/home		user home directory
/lib		shared libraries
/mnt		temp mounted filesystem
/opt		optional packages
/proc		kernel and process files
/root		root user home directory
/run		application state files
/sbin		system admin libs
/srv		data for services
/tmp		temporary files
/usr		user binaries
/var		variable data files

Working with the file tree

pwd	print working directory
ls [directory]	list directory contents
cd [directory]	change directory
.	current Directory
..	parent Directory
mkdir [directory]	make directory
touch [file]	creates a file
rm [file]	removes a file
rmdir [directory]	removes a directory
mv [SOURCE] [DESTINATION]	Move/rename a file
cp [SOURCE] [DESTINATION]	Copy files and directories

Relative and Absolute File Paths

- All files have an Absolute File Path that points to their exact location
  - Root directory is /
  - So an absolute path starts at the root and goes to the file
  - Example - /usr/local/bin
- Relative paths are based on the present working directory (pwd)
  - They do not begin with /
  - Examples
    - ../subdir/myFile
    - anotherSubDir/differentFile

man

- For when you need to learn more about a command, use man
  - Built-in docs for linux commands

- Example – man ls

Other useful commands

Use man to find out what these do

- cat
- head
- tail
- echo
- which
- find

Text Editors

- Command Line Environments have built-in text editors since there is no GUI environment
- VIM is a popular one (that I personally prefer)
- This class encourages Emacs
  - To start emacs, type '**emacs**' in shell
  - I recommend checking out the built-in and online tutorial to get comfortable with Emacs
  - There are lots of online cheat sheets in case you forget any command
  - Homework 1 will heavily involve using Emacs

Emacs Cheat Sheet

- Whenever you get stuck with Emacs, can refer here
  - https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf

Shell Variables

- The shell allows us to assign variables by assigning things like:

```
a=1
echo $a
```

- But shell variables disappear on logoff, they are specific to the session
- Will come back to these more next week

PATH Variable

- The PATH variable is an environment variable that specifies where executable programs are located
  - Whenever we run a program (like ls) our shell searches through the directory in the PATH variable from left to right
  - If it never finds anything – we get an error
  - Look at your path variable with `echo $PATH`

- Environment variables are a set of variables that the current shell and any child processes of that shell will have access to
  - Look at all of your environment variables with `printenv`

Note for Class

Add **export PATH="/usr/local/cs/bin:$PATH"** to your **~/.bash_profile** or the **~/.profile file.**