CS35L
Software Construction Laboratory

Tyler Davis

Week 0 : Lecture 1

Fall 2019

Slides adapted from Nandan Parikh

About this course

- Course website: https://web.cs.ucla.edu/classes/fall19/cs35L/index.html
  - Course Syllabus
  - Assignments (including due dates)
  - Office Hours
  - Emails
  - Grading policies
  - Check it out!
- Why this course?
  - To get accustomed to the most commonly used software environments and tools to be used in upper division CS classes
  - Linux, scripting, VMs, version control management, systems programming, low-level construction, parallelism, etc.

Course Logistics

- **3 unit course**
- Structure
  - 9 assignments (Lab + HW)
  - 1 report + Presentation
- 1 Common Final exam – 3 hour exam
- SEASnet account mandatory!
- Use piazza for questions
  - https://piazza.com/ucla/fall2019/cs35l/home
- Office Hours: TBD

Course Logistics: PTEs

https://tinyurl.com/y2ot7tnj

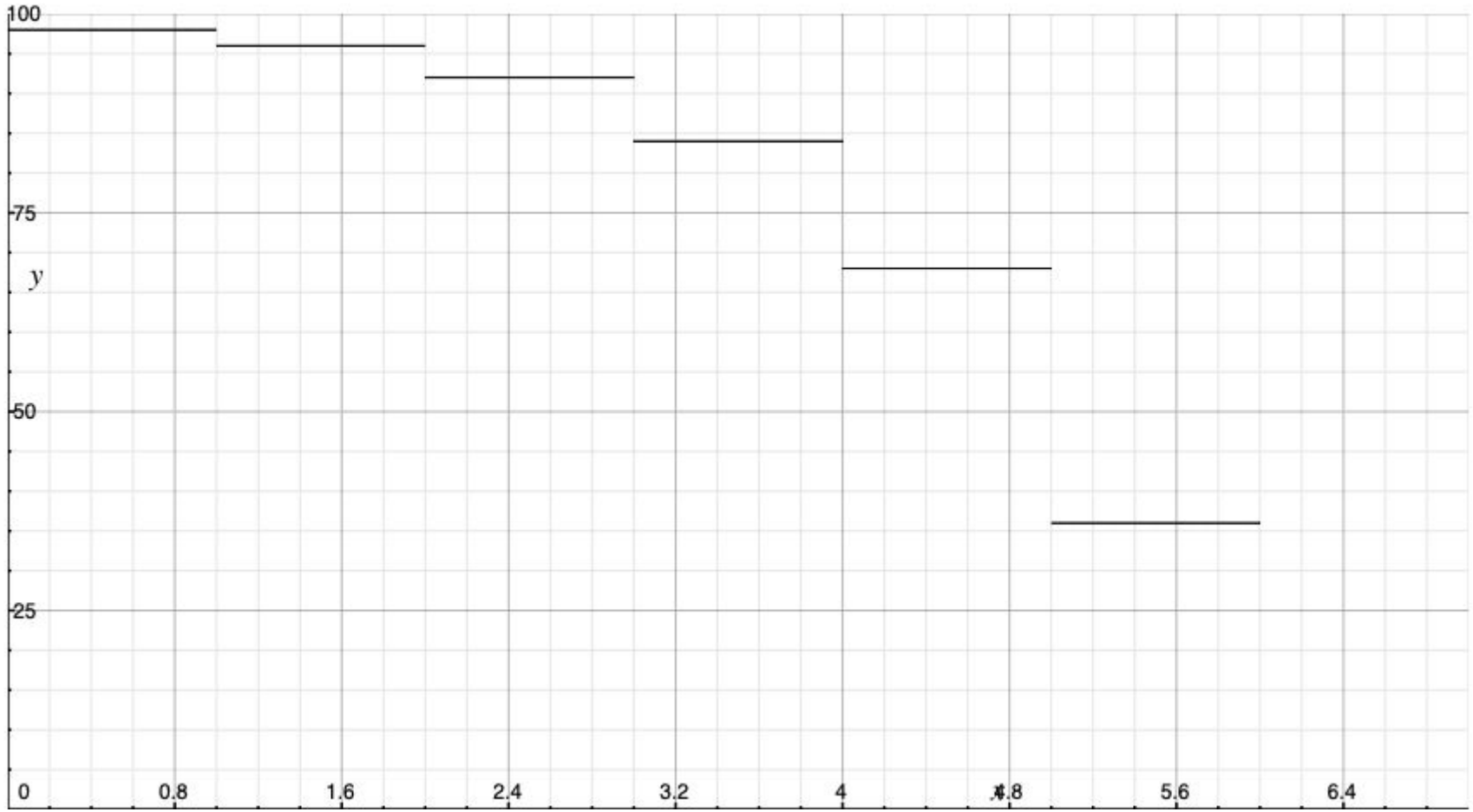
- No guarantees
  - Prioritized based on some different factors

Course Logistics contd…

- Grading : 50% HW and 50% Final exam
- Late penalty: N to N+1 days late -> 2^N % of score deducted
  - Final assignment CANNOT be submitted past 10th week Fri
- All assignments due by 23:55 on specified date
- Instructions for 3760 BH
  - No food or drink
    - Including water
  - Always logout when done with lab computer
- **My email :** tylerdavis@ucla.edu
  - Put “CS 35L” At the start of the subject line

Course Logistics: Cheating

- Don’t do it
- Never send someone a snippet of your code

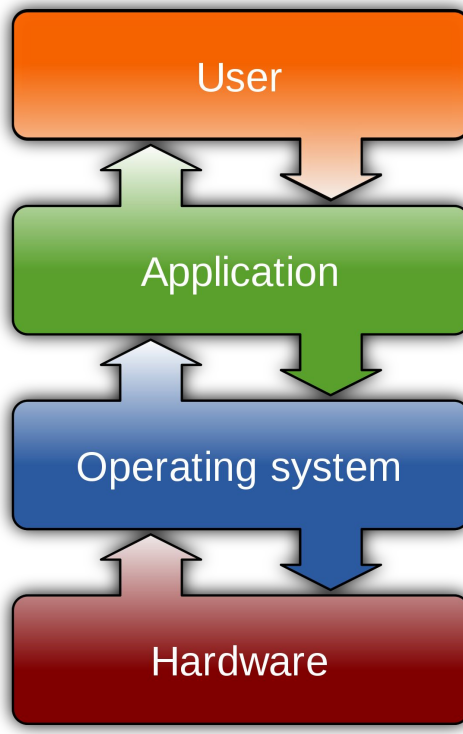

100
75
y
50
25
0
0.8
1.6
2.4
3.2
4
4.8
5.6
6.4

About SEAS account and its connection

- Create account from https://www.seas.ucla.edu/acctapp/
- Remember your SEAS username and password! (it should be different than your UCLA login and password)
- Connect to lnxsrv server
- Install PuTTY SSH client (highly recommended) : Follow instructions on http://www.seasnet.ucla.edu/lnxsrv/
  - Make the password different than your main UCLA login
- Use host lnxsrv.seas.ucla.edu and port 22 in PuTTY

What is an Operating System?

- Most important software that runs on a computer
- Responsible for many things
  - Managing other software
  - Abstracting hardware
- Brief history of Operating Systems: http://www.informit.com/articles/article.aspx?p=24972
- OS Examples: Windows (Windows 10, 8), macOS, FreeRTOS, Linux, Unix



Source : https://en.wikipedia.org/wiki/Operating_system

Multiuser and Multi-process Operating System

- Multi-User OS- Allow many users to access/work on a single system at the same time (as long as they have their own terminal)
- Multi-Process OS- Allows many processes, programs and applications to run simultaneously.
- Variants :
  - Single User Single Task
  - Single User Multi Task
  - Multi User OS

User Interfaces: CLI vs GUI

Command Line Interface

- Steep learning curve
- Pure control (e.g., scripting)
- Speed: Only keyboard, faster performance
- Consumes fewer resources
- No change; less diverse

Graphical User Interface

- More Intuitive
- Limited Control
- Mouse + keyboard; Slower
- More resources; e.g. loading icons, fonts, etc.
- Frequent changes; More diverse

Debian GNU/Linux

- Clone of UNIX
- Linux is just a kernel.
- What is a kernel?
  - Core of any OS
  - Allocates time and memory to programs
  - Interfaces applications with the physical hardware
  - Facilitates communication between different processes (inter-process communication (IPC))
- Linux distributions make the Linux kernel a completely usable OS by adding various applications
- Linux distribution = GUI + GNU utilities (cp,mv,ls,etc) + installation and management tools + GNU compilers (c/c++) + Editors(vi/emacs) + ….
- Shell : Interface between the user and kernel

Basics of Shell

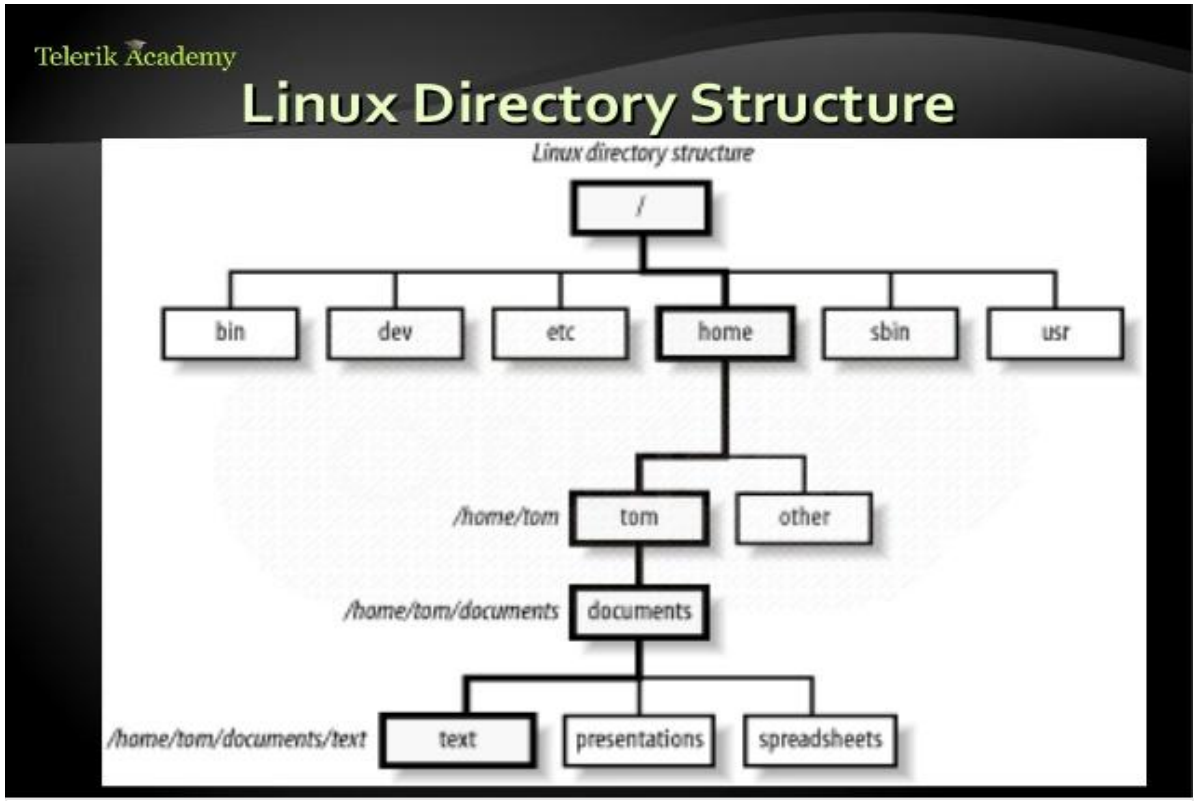
- Can be used as CLI as well as GUI depending upon the task/operation
- Examples:
  - CLI shell on Windows :
    - Command Prompt
  - CLI shell on UNIX :
    - Shell
  - CLI on macOS :
    - Terminal
- Basic shell commands:
  - **<up arrow>**: previous command
  - **<tab>**: auto-complete
  - **!!**: replace with previous command
  - **![*str*]**: refer to previous command with str

Files and Processes

- Everything is either a process or a file
- **Process**: an executing program identified by PID
- **File**: collection of data
    - A document
    - Text of program written in high-level language
    - Executable
    - Directory
    - Devices

Linux File System Layout

- Tree Structure Hierarchy



Source: https://www.slideshare.net/azilian/4-linux-file-systems-18175783

- Only One Root- '/'
- Directories are also files
  - E.g. home, tom
- Regular files can only be leaves
  - E.g. text, spreadsheets, etc

The Basics: Moving Around

- **pwd**: print working directory
- **cd**: change directory
  - **~** home directory
  - **.** current directory
  - **/** root directory, or directory separator
  - **..** parent directory

The Basics: Dealing with Files

- **mv**: move/rename a file
- **cp**: copy a file
- **rm**: remove a file
  - r: remove directories and their contents recursively
- **mkdir**: make a directory
- **rmdir**: remove an empty directory
- **ls**: list contents of a directory
  - d: list only directories
  - a: list all files including hidden ones
  - l: show long listing including permission info
  - s: show size of each file, in blocks

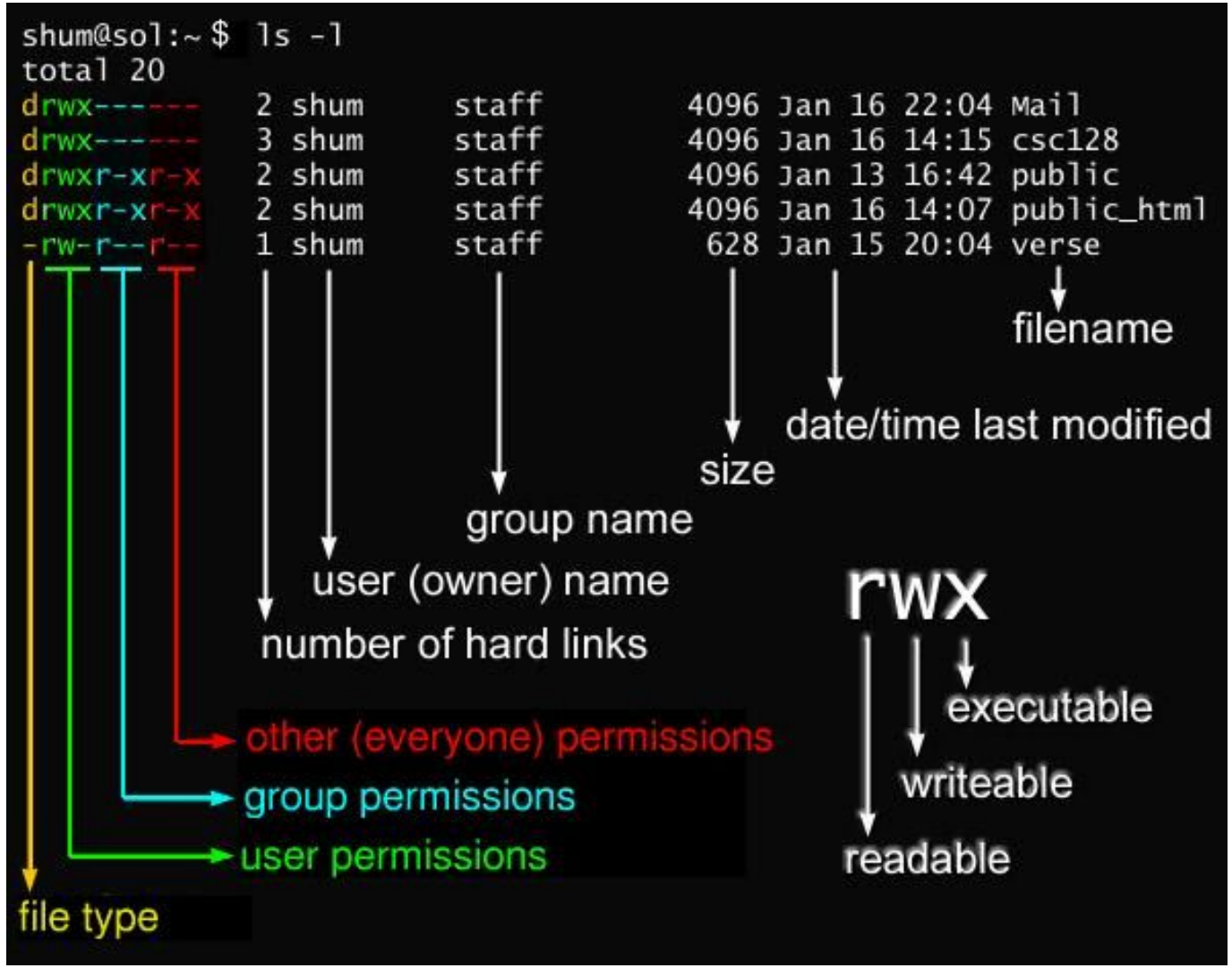
The Basics: Changing File Attributes

- **ln :** creates a link
  - **Hard links** : Point to physical Data
  - Additional name for an existing file
    - ln file1 hlink1
  - **Soft Links/ Symbolic Links (-s):** Point to file
    - ln –s <source file> <my file>
- **touch**: update access & modification time to current time
  - touch *filename*
  - touch -t 201101311759.30 *filename*
    - Change filename’s access & modification time to (year 2011 January day 31 time 17:59:30)

What on Earth are Hard and Soft Links?

- Rundown available in this blog post
  - can also run “man ln” to see more!
- tl;dr
  - Hard Links
    - An additional name for a file (.txt, .py, .doc, etc.)
    - If “original” is removed, the “copy” will still work
      - Copy is not actually a separate copy
    - Can’t point to directories (folders)
  - Soft Links (AKA symbolic links AKA symlinks)
    - Don’t point to the file, point to *something* in the file system
    - Can point to directories, or remote files
    - Deleting original file renders symbolic link unusable

The Basics: File Permissions



File Permissions

- chmod
  - read (r), write (w), executable (x)
  - User, group, others

Reference	Class	Description
u	user	the owner of the file
g	group	users who are members of the file's group
o	others	users who are not the owner of the file or members of the group
a	all	all three of the above, is the same as *ugo*

chmod contd…

- **Numeric**

#	Permission
7	full
6	read and write
5	read and execute
4	read only
3	write and execute
2	write only
1	execute only
0	none

- **Symbolic**

Operator	Description
+	adds the specified modes to the specified classes
-	removes the specified modes from the specified classes
=	the modes specified are to be made the exact modes for the specified classes

Mode	Name	Description
r	read	read a file or list a directory's contents
w	write	write to a file or directory
x	execute	execute a file or recurse a directory tree

Special permissions

- **setuid** : set user ID on execution
- Permits users to run certain programs with escalated privileges
- E.g. : chmod u+s file1
- When an executable file's setuid permission is set, users may access the program with a level of access that matches the owner
- E.g. passwd command

```
ls -l /usr/bin/passwd
```

```
-rwsr-xr-x 1 root 54192 Nov 20 17:03 /usr/bin/passwd
```

Special permissions contd…

- **setgid** : Grants permission of the group which owns the file
- E.g. : chmod g+s file2

```
ls -l myfile2
```

```
-rw-r-sr-- 1 user 0 Mar 6 10:46 myfile2
```

Basic Shell Commands

- man
- cat
- head
- tail
- du
- ps
- kill
- diff
- cmp
- wc
- sort
- echo

The Basics: Redirection

- **> *file*:** write stdout to a file
- **>> *file*:** append stdout to a file
- **< *file*:** use contents of a file as stdin

find command

- -type: type of a file (e.g: directory, symbolic link)
- -perm: permission of a file
- -name: name of a file
- -user: owner of a file
- -maxdepth: how many levels to search

find contd…

- ?: matches any single character in a filename
- *: matches one or more characters in a filename
- []*:* matches any one of the characters between the brackets. Use '-' to separate a range of consecutive characters.
- Examples:
  - find . -name my*
  - find . -name my* -type f
  - find / -type f -name myfile

man command

- Extensive documentation that comes preinstalled with almost all substantial Unix and Unix-like operating systems
- Usage
  - read a manual page for a Linux command
    - **man <**command_name**>**
- Hit “q” to get out of man page

wh commands

- `whatis` <string>: returns Name section of man pages containing given string

```
~ tylerdavis$ whatis python
pydoc(1)                - the Python documentation tool
python(1)               - an interpreted, interactive, object-oriented programming language
pythonw(1)              - run python script allowing GUI
```

- `whereis` <command>: checks the standard binary directories for the specified programs, printing out the paths of any it finds

```
~ tylerdavis$ whereis python
/usr/bin/python
```

- `which` <command>: searches the path for each executable file that would be run had these commands actually been invoked

```
~ tylerdavis$ which python
/usr/local/bin/python
```

diff command

- A file comparison utility that outputs the differences between two files.
- Usage:
  - diff file1 file2
  - diff –u file1 file2 (unified format)

wget command

- A computer program that retrieves content from web servers
- Usage
  - wget <URL>