

# CS 35L: Software Construction Lab

Lab 6

Hengda Shi

Week 1 Lecture 1

# Course Logistics

- TR 12pm-1:50pm, BH 3760
- Office Hours: Tue 2pm-3pm and Wed 12pm-1pm (tentative), BH 3256S-A
- Course Website: <https://web.cs.ucla.edu/classes/winter20/cs35L/>
  - Syllabus, assignments, deadlines, grading policies
- Piazza: <https://piazza.com/ucla/winter20/cs35l/home>
- Email: [hengda.shi@cs.ucla.edu](mailto:hengda.shi@cs.ucla.edu)
- Don't buy Beaglebone yet (keep it for CS 111 if you already bought it).
- PTE

# Grading Policy

- 10 Assignments (50%)
  - Assignments 2-9 and their schedule are tentative. Changes will be noted in [news](#).
  - SEASnet account is mandatory, please create the account ASAP.
- Final Exam (50%)
  - Open book and open note
- Late penalty:  $2^N\%$  deduction for N days late
  - 1% for 0-1 day late, 2% for 1-2 day late, 4% for 2-3 day late, etc.
- Week 10 assignment (must be submitted on time)
  - 10 minutes presentation from selected CS news publications
  - 5 pages report (12pt Times New Roman font, double-spacing, one-inch margins all around)
  - A schedule signup sheet will be up later in the quarter
- Assignment 1 due on 13th Jan 23:55pm

# SEASnet Account

- Create an account: <https://www.seas.ucla.edu/acctapp/>
- Connect to linux server
  - `ssh <seasnet username>@lnxsrv07.seas.ucla.edu`
  - lnxsrv06, lnxsrv07, lnxsrv09, or lnxsrv10
  - `export PATH="/usr/local/cs/bin:$PATH"`
- Or, install PuTTY SSH client (for Windows users) : Follow instructions on <http://www.seasnet.ucla.edu/lnxsrv/>

# Syllabus

1. Introduction, files and editing
2. Commands and basic scripting
3. More scripting, VMs, and construction tools
4. Change management basics
5. Low-level construction and debugging
6. Systems programming
7. Faults, failures, errors, and holes
8. Security basics
9. Advanced change management
10. Trends in computing research and development

# Introduction to Linux

# What is Linux?

- Linux is a family of open source Unix-like operating systems
- Linux operating systems are based on linux kernel first released on Sep 17, 1991 by Linus Torvalds
- Kernel
  - The core of any operating systems
  - Allocate time and memory to programs
  - Allows communication between different processes: inter-process communication (IPC)
- Linux is typically packaged in Linux distributions
- Examples of Linux distributions: Ubuntu, CentOS, RedHat, Debian, Arch
- Linux distribution = GUI + GNU utilities (cp, mv, ls etc...) + installation and management tools + GNU compilers (c/c++) + Editors(vi/emacs) + ....

# What is an Operating System?

- Software program that enables computer hardware to communicate and operate with the computer software.
- Manages memory, processes, other softwares and hardwares
- Computer is useless without an OS!
- Brief history of Operating Systems: <http://bit.ly/2QSdGxx>



# Multiuser and Multi-process Operating System

- Allow many users to work on the same computer at the same time (as long as they have their own terminal)
- Allows many processes, programs and applications to run simultaneously.
- Supports multiprocessing and multitasking
- Multitasking OS examples
  - Windows
  - Linux
  - Unix

# Processes

- A computer program in action
- A process will be constantly changing as the machine code instructions are executed by processor.
- Each process is assigned a pid (process id)
- To see current processes running, use command ``ps``
- To send signals to a process, use command ``kill``
- To list all signals, use ``kill -l``

# Open Source Software

- What is an open source software?
- Source code is publicly available
- Modification by any individual is allowed on a global scale
- It is free for use
- Examples: Firefox, Android, Linux

# CLI vs. GUI

## CLI (Command Line Interface)

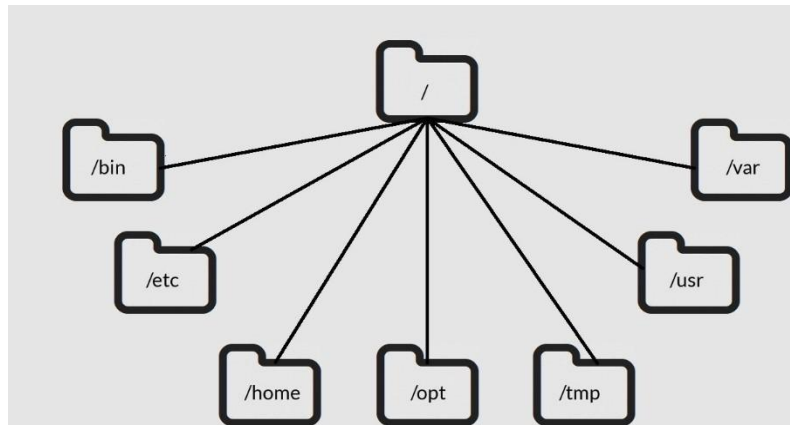
- Steep learning curve
- Pure control (e.g., scripting)
- Cumbersome multitasking
- Speed: Hack away at keys
- Convenient remote access

## GUI (Graphical User Interface)

- Intuitive
- Limited Control
- Easy multitasking
- Limited by pointing
- Bulky remote access

# Unix File System Layout

- Everything is a file
  - A wide range of I/O resources including documents, directories, hard-drives, modems, keyboards, printers, etc. are just streams of bytes exposed through file system namespace
  - No additional command is required to find your CPU information. The CPU information is kept in a file located in **/proc/cpuinfo**. Run command `$ cat /proc/cpuinfo` will give you the CPU information
  - More information: <http://bit.ly/36zKWA8>
- Absolute Path vs. Relative Path
  - Absolute Path starts with “/”, the root directory
  - e.g. `/usr/bin/python`, `/usr/local/bin`
  - Relative Path starts with everything else
  - e.g. `../../home/cs35l`, `./cs35l`
  - Relative Path is relative to the current directory



# Shell

- Outermost layer around the kernel; hence called shell!
- Examples:
  - CLI shell in Windows:
    - Command Prompt
  - CLI shell in UNIX:
    - bash, zsh, fish, etc.

# Linux Commands

# Moving Around

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



# Dealing with Files

- `mv <source> <dest>`: move/rename a file (no undos!)
- `cp <source> <dest>`: copy a file
- `rm <file>`: remove a file
- `touch <file>`: creates a file
- `mkdir <directory>`: make a directory
- `rmdir <directory>`: remove a directory
- `ls [option] <directory>`: 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
  - `-h`: show size with unit suffixes: Byte, Kilobyte, Megabyte, Gigabyte, Terabyte and Petabyte

# File I/O and redirection

- `cat <file>`: print contents in file
  - `cat /proc/cpuinfo`
- `echo <string>`: print string or environment variables
  - `echo "hello world"`
  - `echo $PATH`
- `> file`: write stdout to a file
- `>> file`: append stdout to a file
- `< file`: use contents of a file as stdin

# Mini-task with files

- Create two files
  - `$ touch foo.txt`
  - `$ touch bar.txt`
- Add text into the files and print it
  - `echo "Cat" > foo.txt`
  - `echo "Dog" > bar.txt`
  - `cat foo.txt bar.txt` (concatenates the output)
  - Output:

```
~/Desktop/test  
> cat foo.txt bar.txt  
Cat  
Dog
```

# History

- <up arrow>: previous command
- <tab>: auto-complete
- !!: replace with previous command
- ![str]: refer to previous command with str
- ^[str]: replace with command referred to as str
- history: show a list of commands executed in the past

# 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 <i>ugo</i>

# File Permissions (cont'd)

- 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

# File Types

- Three types of files
  - Regular files
  - Special files (This category has 5 sub-types)
  - Directories
- How to identify file type?
  - `ls -ld foo.txt`
  - `-rw-r--r-- 1 cs35lt6 taaccts 4 Jan 6 10:04 foo.txt`
    - ↑  
File type
  - The first character identifies the type of the file
  - **-** : regular file, **d** : directory, **c** : character device file, **b** : block device file, **s** : local socket file, **p** : named pipe, **l** : symbolic link

# Documentations

- 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>`
  - `q` - quit man page
  - `/<KEYWORD>` - search in man page
    - `n` - next occurrence
    - `N` - previous occurrence



# In command

- In command make links between files
- Two types of links
  - Symbolic Links
    - Create a link to the original file
    - The link will not be able to point to the original file if the original got deleted
    - Useful when the original file is huge
  - Hard Links
    - Create an actual mirrored copy of the original file
    - If the original file is removed, the linked file is unaffected

# wh commands

- `whatis <command>`: returns name section of man page
- `whereis <command>`: locates the binary, source, and manual page files for a command
- `which <command>`: locates only the binary

# find command

- -type: type of a file (e.g., directory, symbolic link)
- -perm: permission of a file
- -name: name of a file
- -prune: don't descend into a directory
- -o: or
- -ls: list current file

# Linux Wildcards

- Asterisk (\*) - matches one or more occurrences of a character
  - `$ touch file1 file2 file3`      # create three files named file1, file2, and file3
  - `$ ls -la f*`      # list all files and directories start with character f
- Question Mark (?) - matches on a single occurrence of character
  - `$ touch file1 file2 file3 f foo`      # create five files named file1, file2, file3, f, and foo
  - `$ ls -la f?`      # no matches found
- Brackets ([]) - matches any single occurrence of characters in the bracket
  - `$ touch fim flm film fm`      # create four files fim, flm, film, fm
  - `$ ls -la f[il]m`      # output only fim and flm

# Other Useful Commands

- diff - compare files line by line
- cmp - compare two files byte by byte
- wc - print newline, word, and byte counts for each file
- sort - sort or merge records (lines) of text and binary files
- head - display first lines of a file
- tail - display the last part of a file
- top - display and update sorted information about processes
- du - display disk usage statistics

# Task 1 (Hint: use “wc” command)

- Create 2 .txt files and add multiple lines in each one of them
- Get the line count for each of these files

# Assignment 1

- Submit on CCLE under your Lab <number> (if specified)
- No submissions will be accepted via email
- Test your files on seasnet linux server before submitting
- key1.txt should record keystrokes of Homework
- ans1.txt should have keystrokes and answers of the lab assignment
- For Lab 1.3, search programs under “/usr/bin”.