CS 35L
Discussion 1A
Week 0

Linux and Emacs Basics

Logistics

1. Introduce Self
    a. TA - Yuxing Qiu (yxqiu@g.ucla.edu)
        i. Office Hour : Friday 12:30 -2:30 PM Boelter Hall 3256S-D (Starting from Week 1)
    b. LA - Ning Wang (ningwang1729@g.ucla.edu)
2. Discussions (10 am - 11:50 am Dodd Hall 167)
    a. 1st hour: review on specific skills for homework; add-on to the lecture materials
    b. 2nd hour: hands-on practice on worksheets with LA
    c. HW-oriented (Lab, hints ...)
    d. Join us!

Logistics

- Class Website
  - https://web.cs.ucla.edu/classes/fall22/cs35L/
- Qs and to get help
  - Piazza: http://piazza.com/ucla/fall2022/cs35l
  - Email
  - Office hour: F 12:30 -2:30 PM Boelter Hall 3256S-D
  - Office: Slichter Hall 3860 (I would be in office on M/W/F; please wear a mask if you would like to walk by, thanks!)

Let's start!

- This course: **Software Construction Laboratory**
- Let's think about some questions first:
  - Why did you enroll this course?
  - What are you expecting to learn in this course?

- Syllabus https://web.cs.ucla.edu/classes/fall22/cs35L/syllabus.html

Let's start!

- Syllabus https://web.cs.ucla.edu/classes/fall22/cs35L/syllabus.html

The Golden Circle

WHAT
Every organization on the planet knows WHAT they do. These are products they sell or the services

HOW
Some organizations know HOW they do it. These are the things that make them special or set them apart from their competition.

WHY
Very few organizations know WHY they do what they do. WHY is not about making money. That's a result. WHY is a purpose, cause or belief. It's the very reason your organization exists.

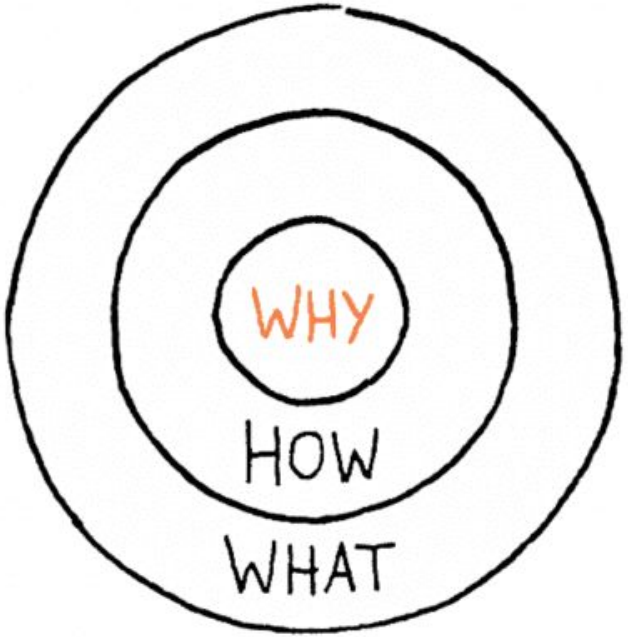


**Image from Internet*

Let's think about another "construction"

Building Construction

- Where to set up the building?
  - Understand the environment
  - Climate? To choose material
- How to play with the environment?
  - How can I set up buildings onto the ground?
  - How to make use of the resources?
  - How to remove/discard useless things?
- How to construct?
  - Design first
  - How to record and visualize your ideas？
    - Tools? Softwares?
    - Other materials/resources if you have sth unclear about?
  - How to manage your resources?
  - How to collaborate with others?
  - How to manage the progress of your work?

**Image from Internet*

Let's think about another "construction"

Building Construction

- Where to set up the building?
  - Understand the environment
  - Climate? To choose material
- How to play with the environment?
  - How can I set up buildings onto the ground?
  - How to make use of the resources?
  - How to remove/discard useless things?
- How to construct?
  - Design first
  - How to record and visualize your ideas？
    - Tools? Softwares?
    - Other materials/resources if you have sth unclear about?
  - How to manage your resources?
  - How to collaborate with others?
  - How to manage the progress of your work?

Software Construction

- Where to set up the software?
  - Environment: Operating System
  - Internet: client server
- How to play with the environment?
  - Set up your software on a computer
  - How to tell computer software what to do?
    - Commands and coding
  - How to manage your data?
- How to construct?
  - Design first
  - How to record and visualize your ideas (and codes)？
    - Tools? Softwares?
    - Look up for other materials/resources?
  - How to manage your resources?
  - How to collaborate with others?
  - How to manage the progress of your work?

Let's go back to our syllabus

Software Construction

- Where to set up the software?
  - **Environment: Operating System**
  - **Internet: client server**
- How to play with the environment?
  - Set up your software on a computer
  - How to tell computer software what to do?
    - Commands and coding
  - How to manage your data?
- How to construct?
  - Design first
  - How to record and visualize your ideas (and codes)?
    - Tools? Softwares?
    - Look up for other materials/resources?
  - How to manage your resources?
  - How to collaborate with others?
  - How to manage the progress of your work?

Files, editing, and shells

- Multiuser and multiprocess o
- CLI basics (e.g., Bash)
- Unix file system organization
- Basic shell commands: ls; pwd
- Documentation and man pag
- Emacs basics: introduction, or running shell commands, bui

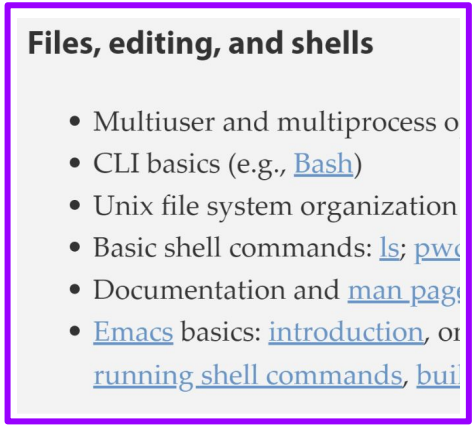
Client-server apps and user interfaces

- Scriptable user interfaces via JavaScript
- HTML and web protocols
- Case study: Node.js and React.

Let's go back to our syllabus

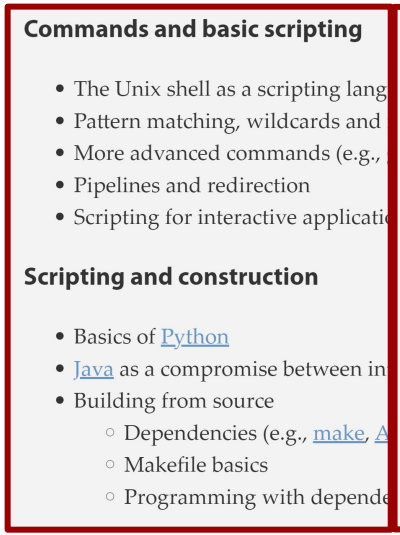
Software Construction

- Where to set up the software?
  - Environment: Operating System
  - Internet: client server
- How to play with the environment?
  - Set up your software on a computer
  - **How to tell computer software what to do?**
    - **Commands and coding**
  - **How to manage your data?**
- How to construct?
  - Design first
  - How to record and visualize your ideas (and codes)?
    - Tools? Softwares?
    - Look up for other materials/resources?
  - How to manage your resources?
  - How to collaborate with others?
  - How to manage the progress of your work?



Files, editing, and shells

- Multiuser and multiprocess o
- CLI basics (e.g., Bash)
- Unix file system organization
- Basic shell commands: ls; pwd
- Documentation and man pag
- Emacs basics: introduction, or running shell commands, bui

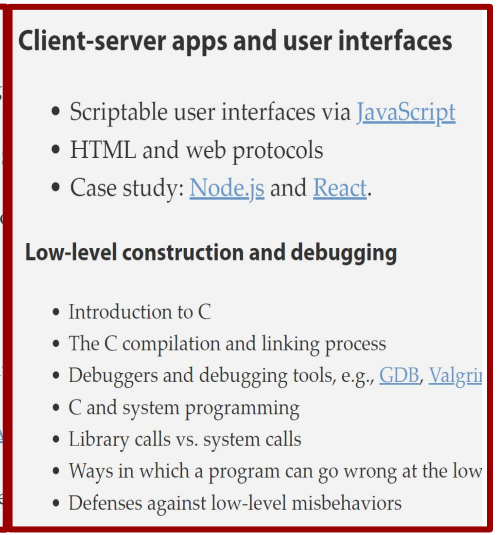


Commands and basic scripting

- The Unix shell as a scripting lang
- Pattern matching, wildcards and
- More advanced commands (e.g.,
- Pipelines and redirection
- Scripting for interactive applicati

Scripting and construction

- Basics of Python
- Java as a compromise between in
- Building from source
  - Dependencies (e.g., make, A
  - Makefile basics
  - Programming with depende



Client-server apps and user interfaces

- Scriptable user interfaces via JavaScript
- HTML and web protocols
- Case study: Node.js and React.

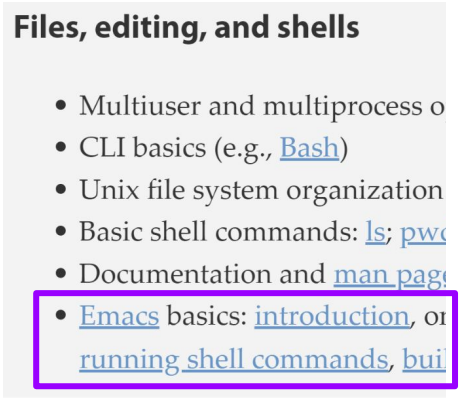
Low-level construction and debugging

- Introduction to C
- The C compilation and linking process
- Debuggers and debugging tools, e.g., GDB, Valgri
- C and system programming
- Library calls vs. system calls
- Ways in which a program can go wrong at the low
- Defenses against low-level misbehaviors

Let's go back to our syllabus

Software Construction

- Where to set up the software?
  - Environment: Operating System
  - Internet: client server
- How to play with the environment?
  - Set up your software on a computer
  - How to tell computer software what to do?
    - Commands and coding
  - How to manage your data?
- How to construct?
  - **Design first - Try it in your Group Project!**
  - How to record and visualize your ideas (and codes)?
    - **Tools? Softwares?**
    - Look up for other materials/resources?
  - How to manage your resources?
  - How to collaborate with others?
  - How to manage the progress of your work?



Google; Online Doc; Manuals; Blogs; Wiki; Online Examples; ....

Learn from reading others' code (open-source resources)

Let's go back to our syllabus

Software Construction

- Where to set up the software?
  - Environment: Operating System
  - Internet: client server
- How to play with the environment?
  - Set up your software on a computer
  - How to tell computer software what to do?
    - Commands and coding
  - How to manage your data?
- How to construct?
  - **Design first - Try it in your Group Project!**
  - How to record and visualize your ideas (and codes)?
    - Tools? Softwares?
    - Look up for other materials/resources?
  - **How to manage your resources?**
  - **How to collaborate with others?**
  - **How to manage the progress of your work?**

Package management

- Dependencies and packages
- Installing a package
- Creating a package
- Package management case study: (e.g., Python packaging)

Change management

- Creating and using patches (e.g., diff and patch)
- File-based version control, e.g., RCS
- Package-based version control, e.g., CVS
- Decentralized version control, with Git as a case study
  - Commits
  - Branching and merging
  - Rebasing
  - Branching workflows
  - Searching
  - Advanced merging and conflict resolution
  - Remote branches
  - Submodules
  - Git internals

Let's go back to our syllabus

- I know there are too many things/topics/tools to learn
- I understand this course is pretty heavy-loaded
  - 6 HWs - each may consume a lot of learning and implementation time
  - 1 Group project
  - 2 midterms

But! Let's always keep our big picture in mind !!

- **Why** do we want to study and practice with this topic?
  - Why it's important in software construction?
- **How** to find resources online to help you to learn to achieve your **goal**?
  - Impossible to remember every tech detail.
  - Make use of the online tutorials and examples. Learn from imitation and practice.
- What did you learn?
  - Try to remember the big picture, general idea, the pros and cons of each tech we've played with.

Contents

1. **Environment Setup for Assignments**
2. Linux Basics
3. Some Useful Shell Commands/Tricks
4. Emacs

Let’s leave things to next week if we’re not able to finish today.

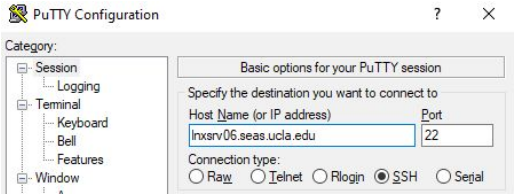
1. Environment Setup

- Local environment is fine
- We recommend SEASNet
    - Verify your assignments; Some resources

- [Step 1] Apply a SEASNet Account (https://www.seas.ucla.edu/acctapp/)
    - Please don’t wait till the last minute. It will take time to activate the account.
- [Step 2] UCLA VPN (https://www.it.ucla.edu/it-support-center/services/virtual-private-network-vpn-clients)
- [Step 3] Connect to the SEASNet (next slide)

1. Environment Setup -- Continue

- [Step 3] Connect to the SEASNet
  - Linux:
    - Secure Shell (SSH) is a cryptographic network protocol used for a secure connection between a client and a server.
    - `sudo apt install openssh-client`
    - `ssh username@lnxsrv11.seas.ucla.edu`
    - `11` can also be `12`, `13` or `15`
  - Windows:
    - Putty + Xming (https://www.seasnet.ucla.edu/putty/ and https://www.seasnet.ucla.edu/xming/)
    - Tutorial (https://laptops.eng.uci.edu/engineering-software/using-linux/how-to-configure-putty-xming-on-your-laptop)
  - MAC:
    - Similar to Linux
    - Built-in SSH client
    - For GUI (similar to Xming) XQuartz (https://www.xquartz.org/)
  - Some other useful tools: VSCode ...



1. Environment Setup -- Continue

- [Step 4] Copy Server Files to Local
  - scp command
    - `scp username@lnxsrv#.seas.ucla.edu:/remote/directory/filename /local/directory`
  - Git
    - No worry if you don’t know, we will learn it later
  - Tools with GUI
    - WinSCP
    - FileZilla
    - …
- [Step 5] Compress and submit to CCLE
    - `tar -tvf assign1.tgz`

Contents

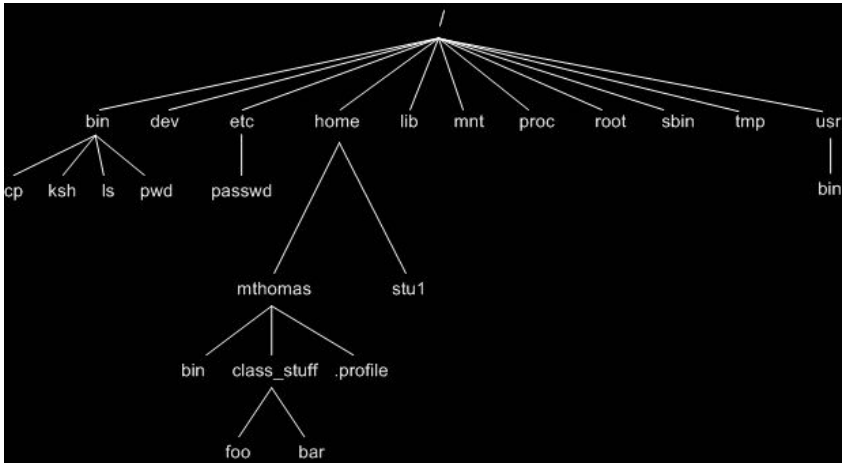
1. Environment Setup for Assignments
2. **Linux Basics**
3. Some Useful Shell Commands/Tricks
4. Emacs

2. Linux Basics

- Linux
  - A family of open-source Unix-like ***operating systems*** based on the Linux kernel
- Shell
  - A command-line interpreter that provides a command line user interface
  - Typically user interact with the unix shell using a ***terminal*** application
- The Unix File System
  - Methodology for logically organizing and storing large quantities of data (easy to manage)
  - File: (informally) collection of data
  - File system:
    - Files, relationships of files, attributes of each file (type, name, size...)
    - Tools which allow the manipulation of files
    - Beginner perspective: files and directories

2. Linux Basics -- Continue

- The Unix File System: Tree structure
  - *bin*: short for binaries; the directory for commonly used executable commands
  - *home*: contains user directories and files
- Navigate through the system



https://homepages.uc.edu/~thomam/Intro_Unix_Text/File_System.html

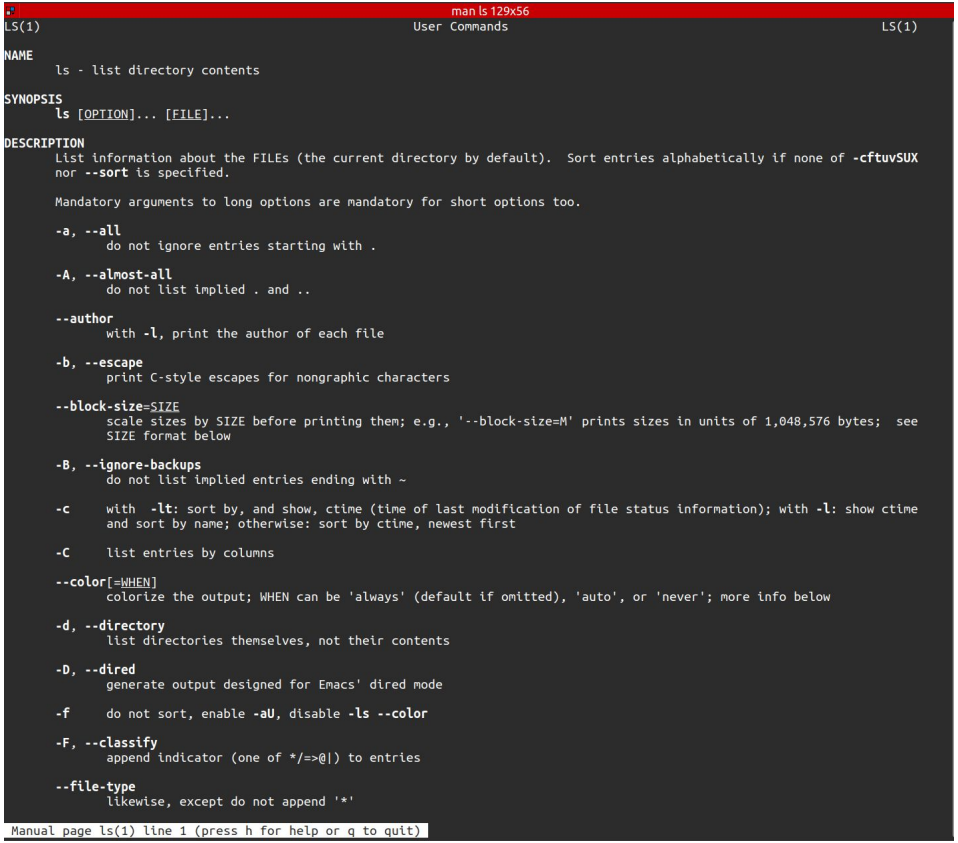
pwd	Print working directory
ls [directory]	List directory contents; -l for long format; -a for list all ...
cd [directory]	Change directory
.	Current directory
..	Parent directory
mkdir [directory]	Make a new directory
touch [file]	Create a file
rm [file] | rm -r [directory]	Remove a file / directory
cp [source] [destination]	Copy files; Copy directories (with -r)
mv [source] [destination]	Move/rename a file

Contents

1. Environment Setup for Assignments
2. Linux Basics
3. **Some Useful Shell Commands/Tricks**
4. Emacs

3. Shell Commands

- Most useful command!!! -- man
  - `man [command]`
  - Short for manuals
  - Built-in docs for commands
  - E.g.: `man ls`
- Learn new commands
  - Google for examples
  - Could be more friendly to new learners



3. Shell Commands -- Continue

- Other useful commands
  - Use `man` to check out these commands
  - `echo`, `grep`, `which`, `find`, `cat`, `head`, `tail` ...

3. Shell Commands -- Continue

- *chmod*
  - Change user permissions to a file/folder
  - Reading the bits
    - **u** owner, first 3 bits
    - **g** user group that owns the file, next 3 bits
    - **o** users not in **u** and **g**, last 3 bits
    - **a** all three groups, all 9 bits
  - Permission flags
    - **r** - can read the file / can list the files within the dir)
    - **w** - can write the file / can create/rename/delete files in the dir, or change dir attributes
    - **x** - can execute the file / enter dir and access files and dirs inside
  - Commands to modify permissions
    - \+ add permissions; - remove permissions; = set permissions
    - `chmod [options] mode[,mode] file1[file2 ...]`
  - e.g.
    - `chmod u+x filename; chmod go-wx filename;`
    - `chmod 777 filename (rwxrwxrwx)`
    - `chmod 550 filename (r-xr-x---)`

```
~/tst ❯ ll
total 28K
-rw-rw-r-- 1 yuxing yuxing   59 May 24 20:55 anothergrepFile
lrwxrwxrwx 1 yuxing yuxing   11 May 24 21:05 anotherlink -> anotherlink
-rw-rw-r-- 1 yuxing yuxing 3.1K Mar 27 21:24 atestoutput
-rw-rw-r-- 1 yuxing yuxing 2.7K Mar 27 22:09 atestoutput.txt
-rwxrwxrwx 1 yuxing yuxing  107 May 10 13:06 countless
-rw-rw-r-- 1 yuxing yuxing   44 May 24 20:54 grep.txt
-rw-rw-r-- 1 yuxing yuxing   14 Jun 11 09:44 testfile
drwxrwxr-x 3 yuxing yuxing 4.0K Jun 11 10:39 testfolder
-rw-rw-r-- 1 yuxing yuxing    0 Jun 11 09:45 test.test
```

#	Permission	rwx	Binary
7	read, write and execute	rwx	111
6	read and write	rw-	110
5	read and execute	r-x	101
4	read only	r--	100
3	write and execute	-wx	011
2	write only	-w-	010
1	execute only	--x	001
0	none	---	000

Contents

1. Environment Setup for Assignments
2. Linux Basics
3. Some Useful Shell Commands/Tricks
4. **Emacs**

4. Emacs

- Reference Card
  - GNU Emacs Reference Card
- Make your own reference card
  - Example(my own ref card)
  - https://docs.google.com/document/d/17cg2oXekV-vB6lnAhib4tYNgTbucMZ34aH5zTwQFD8A/edit?usp=sharing

Shell commands that might be useful

Emacs in Terminal

- Open emacs in terminal:
  - ***emacs -nw* (--no-window-system)**
- Leave Emacs (Emacs commands)
  - C-x C-c
- Suspend Emacs
  - C-z
- List all background jobs
  - ***jobs***
- Resume the job
  - ***fg***: resume the job that's next in the queue
  - ***fg %[number]***: resume the job with id <number>
  - ***fg %emacs***: resume Eamcs
- Kill the job
  - ***kill %[number]***: kill the job numbered <number>
  - ***kill -9 [PID]***: kill the job with process id PID (-9 is the SIGKILL signal)
  - ***killall -9 emacs***: kill all emacs process
- Process PID Number
  - ***pidof [program_name]***: eg: pidof emacs
  - ***ps -p [PID] -o comm=***: find process name using PID