CS 97 - Discussion 1F
Week 1

Linux and Emacs Basics

Logistics

- Introduce Self
  - Yuxing Qiu (yuxqiu@gmail.com; yxqiu@g.ucla.edu)
  - Office Hour : Monday 9:30-10:30 am & Monday 9:30-10:30 pm
- Discussions
  - Roughly half led by TA; the other half by LAs
  - Review and add onto the lecture materials
  - HW-oriented (Lab, hints ...)
  - Join us!
- Qs and to get help
  - Piazza
  - Email
  - Office hour

Logistics

- Workload
  - ~6 assignments (~25%)
  - midterm (~15%)
  - group project (~35%)
  - groups of 5 members, to limit number of projects
  - final (~25%)
- Late submissions
  - Penalty 2^(N-1)%

➔ Grading is kept fairly lenient
- No auto grading scripts
- Only check for a "reasonable" attempt
- "Minor" deviations, mistakes, etc. are ignored
- Focus on thorough understanding
- Avoid assignment anxiety

➔ Assignment
- Available
  https://web.cs.ucla.edu/classes/winter21/cs97-1/assign/assign1.html
- Due on 15th Jan, 11.55pm ucla time
- Submission link is on ccle (under week 2)
  https://ccle.ucla.edu/mod/assign/view.php?id=3563091

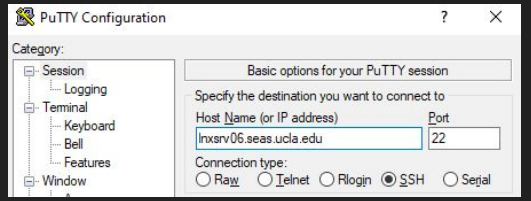
Contents

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/)
- [Step 2] UCLA VPN (https://www.it.ucla.edu/it-support-center/services/virtual-private-network-vpn-clients)
- [Step 3] Connect to the SEASNet

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@lnxsrv06.seas.ucla.edu`
    - `06` can also be `07`, `09` or `10`
  - Windows:
    - Putty + Xming (https://www.seasnet.ucla.edu/putty/ and https://www.seasnet.ucla.edu/xming/)
    - Tutorial (http://laptops.eng.uci.edu/software-installation/using-linux/how-to-configure-xming-putty)
  - MAC:
    - Similar to Linux
    - Built-in SSH client
    - For GUI (similar to Xming) XQuartz (https://www.xquartz.org/)



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

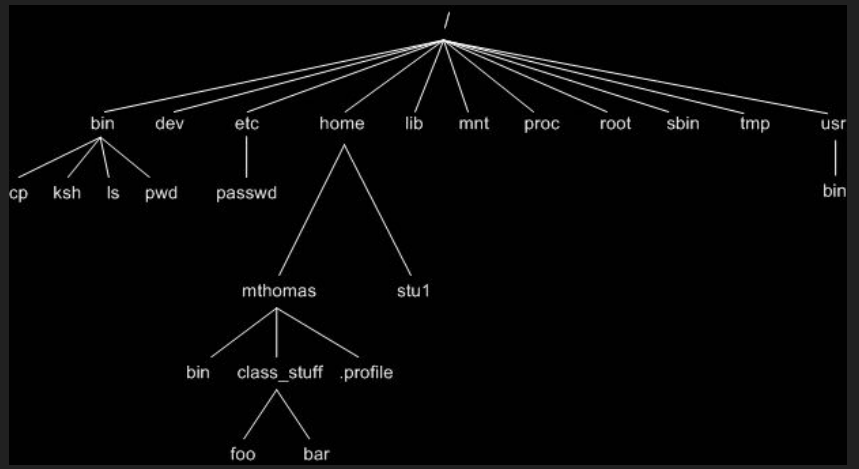
2. Linux Basics

- 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

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

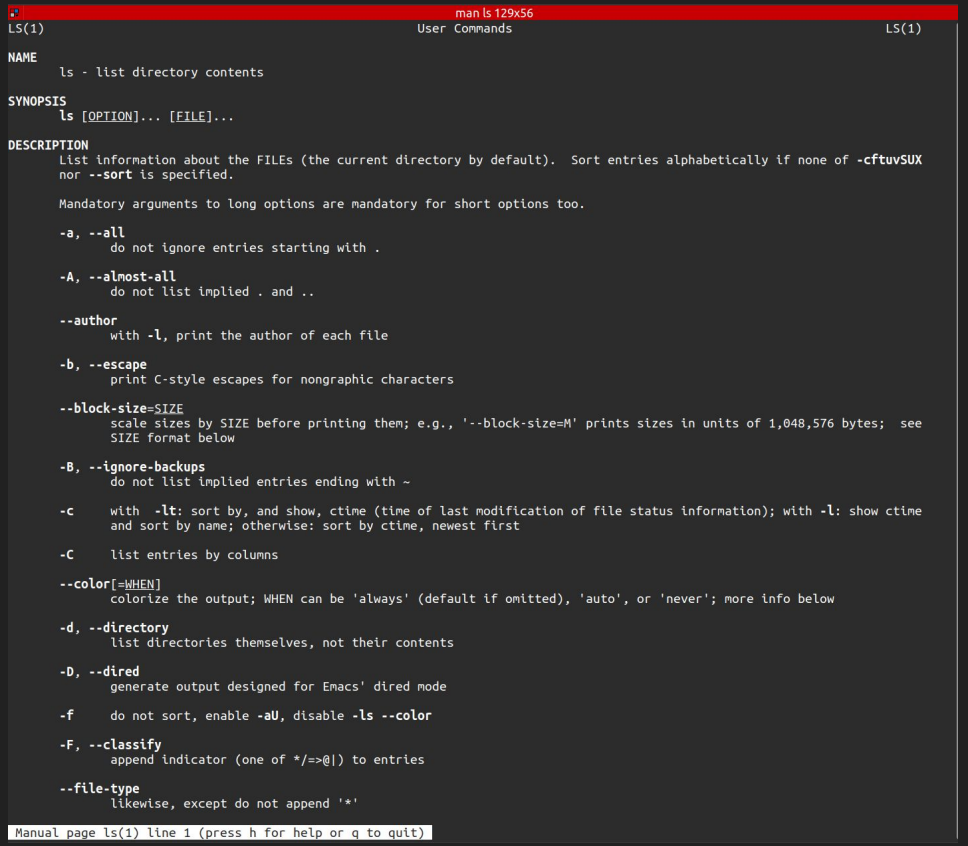


https://homepages.uc.edu/~thomam/Intro_Unix_Text/File_System.html

Contents

3. Shell Commands

- Most useful command!!! -- man
  - `man [command]`
  - Short for manuals
  - Built-in docs for commands
  - E.g.: man ls



3. Shell Commands -- Continue

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

#	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

- *chmod*
  - 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
    - **w** - can write
    - **x** - can execute
  - Commands to modify permissions
    - \+ add permissions; - remove permissions; = set permissions
    - `chmod [number] [file]`
  - e.g.
    - `chmod u+x filename; chmod go-wx filename;`
    - `chmod 777 filename (rwxrwxrwx)`
    - `chmod 550 filename (r-xr-x---)`

3. Shell Commands -- Continue

- Some useful shell operators
  - Pipe operator |: passes the output of one command as input to another
    - E.g.: `ls . | grep “a”`
    - More control operators: https://pubs.opengroup.org/onlinepubs/9699919799/basedefs/V1_chap03.html#tag_03_113
  - Redirection operator: https://www.gnu.org/savannah-checkouts/gnu/bash/manual/bash.html#Redirections
    - `command <` : gives input to a command, e.g.: `grep "int" -n < hello.c`
    - `command > out.txt` : Directs the output of a command into a file out.txt
    - stdin, stdout and stderr: data streams created when you launch a Linux command
      - 0: stdin, 1: stdout, 2:stderr
      - `E.g.: command 0<&- >out.txt 2>error.txt`
      - `# turn off the stdin, redirect the stdout to out.txt, stderr to error.txt`
    - `>` v.s. `>>`:
      - `>` : overwrite the file (erase previous contents)
      - `>>` : append to the file (preserve previous contents)

3. Shell Commands -- Continue

- Shell script
  - Create a file
  - Add 1st line `#!/bin/bash` or `#!/usr/bin/bash`
  - Edit your script
  - Add execute permission with `chmod +x myscript`
- `#!` In the first line
  - Tell the shell what program to interpret the script with, when executed
  - When the shell runs a program, it asks the kernel to start a new process and run the given program in that process. It knows how to do this for compiled programs. But for a script, we should tell the OS how to run the file. With `#!/usr/bin/bash`, the shell knows to use the bash interpreter to run the file.
  - Normally `#` starts a comment line

3. Shell Commands -- Continue

- Shell variables
  - Assign variables by assignments
    - `a=1`
    - `echo $a` (“echo a” will not work)
    - Shell variables disappear once log off, they are specified to the current session
  - Built-in shell variables (can be accessed in the shell script)

$#	Number of arguments provided to script
$0	Name of script
$1, $2, etc	1st and 2nd argument, etc
${15}, ${23}, etc	For arguments greater than 9
$?	Exit status of last command
$$	Current running process ID

Contents

4. Emacs

- Reference Card
  - https://www.gnu.org/software/emacs/refcards/pdf/refcard.pdf
- Make your own reference card
  - Example(my own ref card)
  - https://docs.google.com/document/d/1R0HLUqBzhnfghK761BKtfXFfmeHr_f3tOnDzzE9289Q/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