CSE 391

Syllabus

Introduction to Linux

Slides created by Josh Ervin. Based off slides made by Marty Stepp, Jessica Miller, Ruth Anderson, Brett Wortzman, and Zorah Fung

AGENDA

- Course introduction and syllabus
- Learning remotely using Zoom
- Unix and Linux operating systems
- Introduction to Bash shell

COURSE INTRODUCTION

- Course website: https://cs.washington.edu/391
- Learning Objectives: A collection of tools and topics not addressed in other classes that all CSE majors should know
- Credit/No Credit class, assessed on weekly homework assignments
- Textbook - *Linux Pocket Guide*
  - Optional, but we can only recommend it

ASSESSMENT

- Homework Assignments
  - There will be 9 assignments, each worth 2 points
  - Released after lecture, due the following Tuesday at 1:00pm
  - 1 point for a reasonable attempt on most questions*
  - 2 points for attempting all questions and most are correct*
  - * For most assignments this is graded by the autograder, and you will receive your score immediately after submission
  - No late assignments accepted
  - **Collaboration is encouraged, but all submitted work must be your own.**
- Passing the Class
  - Of the 18 possible points, you need 14 to pass the class
  - You may earn one extra point by doing the intro survey!

DISCUSSION BOARD

- Outside of office hours, the best way for you to get help with the class is to post on the course discussion board - Ed
- For personal questions, email the course staff. Otherwise, all questions should be posted to Ed. You may post anonymously to your classmates if you prefer!

COURSE STAFF

Lecturer: Josh Ervin

- Email: joshue@cs.washington.edu
- OH: Mon 3:30-4:30pm

Administrator: Hunter Schafer

- Email: hschafer@cs.washington.edu
- OH: Tue 10:30am-12:00, Wed 11:30am-1:30

REMOTE LEARNING

- All lectures and office hours will be held remotely via Zoom
- Lectures:
  - Josh will be giving “live” lectures during the allocated lecture time on Tuesdays from 1:30-2:30
  - Attendance is not recorded, but it will be beneficial for you to ask questions during this time.
- Access:
  - To access lectures and office hours, visit the Zoom tab on the course website.
  - We understand that not everyone will be able to attend all lectures, especially given the circumstances. All lectures are recorded and will be available via the Zoom section on Canvas.

ACCESSING LINUX/UNIX

- In a roughly suggested order
  - ssh into attu (CSE Majors), linuxNN(EE majors), or ovid (all UW students)
  - Use the Virtual Machine (VM) provided by the CS Department
  - Access basement lab computers via vdi
  - Install Linux on your own computer
- For detailed information, see the Working-At-Home tab on the course website.

COURSE TOPICS

- Linux command line interface (CLI)
- Shell commands, input/output redirection
- Version control using `git`
- Users, groups, and permissions
- Regular expressions, sed
- Basic data processing
- Shell scripting
- Industry applications of Linux (TBD)

A BRIEF HISTORY OF LINUX AND UNIX

- Unix
  - First developed in 1969 at Bell Labs by Dennis Ritchie and Ken Thompson
  - Many key ideas still used today
    - “Everything is a file”
    - Multiple users, hierarchical file system
    - “Glueing” together lots of smaller files
    - Documentation included
  - macOS is a unix operating system in disguise!
- Linux
  - Developed in 1992 by Linus Torvalds, who also developed git!

THE SHELL

- Shell: an interactive program that allows the user to interact with the operating system and its applications
- Why use a shell vs. the GUI (Graphical User Interface)?
    - Many complicated tasks are easier to do on the command line
    - Useful for working on remote servers
    - Programmable
    - Customizable

BASIC SHELL COMMANDS

command	**description**
`pwd`	**P**rint current **w**orking **d**irectory
`cd`	**C**hange working **d**irectory
`ls`	List files in working directory
`man`	Bring up manual for a command
`exit`	Log out of shell

SYSTEM COMMANDS

command	description
`clear`	Clears all output from console
`date`	Output the system date
`cal`	Output a text calendar
`uname`	Print information about the current system

RELATIVE DIRECTORIES

directory	**description**
`.`	References the working directory
`..`	References the parent of working directory
`~username`	`username`’s **home** directory
`~/Desktop`	Your desktop

UNIX FILE SYSTEM

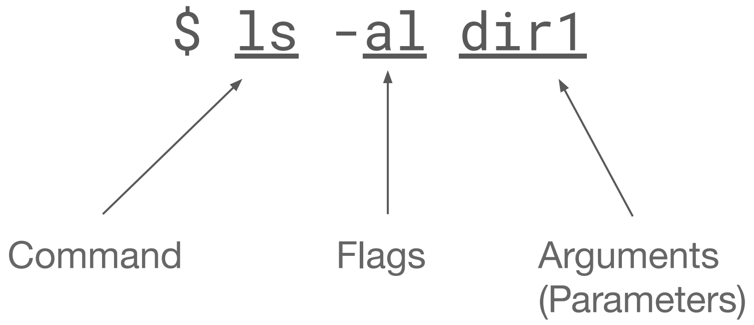
directory	description
`/`	Root directory that contains all directories
`/bin`	Applications/programs (i.e. binaries)
`/dev`	`Hardware devices`
`/etc`	Configuration files
`/home`	Contains user’s home directories
`/proc`	Running programs (processes)
`/tmp, /var`	Temporary files
`/usr`	Universal system resources

DIRECTORY COMMANDS

directory	description
`ls`	List files in working directory
`pwd`	**P**rint current **w**orking **d**irectory
`cd`	**C**hange working **d**irectory
`mkdir`	Make a new directory
`rmdir`	Remove the given directory (must be empty)

COMMAND LINE ARGUMENTS

- There aren't any consistent definitions when it comes to command line arguments, but for this class we will use the following way to describe the anatomy of a command



```
$ ls -al dir1
```

COMMAND LINE ARGUMENTS

- Much like methods in Java take arguments, so do commands on the command line
- Flags are modifiers which change a programs behavior slightly, and they are usually prepended with a -
- For example, to list all files in long-list format, run the following
  - `$ ls -l`
- Flags can be combined, to list all files in long-list format and list hidden files
  - `$ ls -la`
- Commands also take arguments, such as file names
- To view all files , in long-listing format, inside of `dir1`
  - `$ ls -l dir1`

FILE COMMANDS

directory	description
`cp`	Copy a file
`mv`	Move a file (also used to rename files)
`rm`	Remove the given file
`touch`	Create empty file, or change time-modified

- *Warning*: The above commands do **not** ask for confirmation. Be careful moving or copying files, as you might overwrite existing files!
- Check the `man` pages for flags to prevent this behavior

TEXT EDITORS

command	description
`nano`	Very simple editor
`vim`	More advanced text-editor
`emacs`	More advanced text-editor

- In many instances, you will be interacting with a Linux system with a graphical environment so a command-line text-editor is necessary
- vim/emacs are powerful text-editors preferred by many experienced Linux users. It’s up to you which one to focus on in this class.

VIM BASICS

Key stroke	description
`:w`	Write (save) the current file
`:wq`	Write (save) the current file and exit
`:q!`	Quit, ignoring all changes
`i`	Go into insert mode
`Esc`	Go back to normal mode
`hjkl`	Move cursor left, down, up, right
`u`	Undo last change
`x`	Delete character

EMACS BASICS

C = control key

M = alt/meta key

Key stroke	description
`C-x C-f`	Read a file into emacs
`C-x C-s`	Save a file to disk
`C-x C-c`	Exit emacs
`C-s`	Search forward
`C-r`	Search backwards
`C-v`	Scroll to next screen
`M-v`	Scroll to previous screen
`C-x u`	Undu