CSE 391: System & Software Tools

Shell

1. Logistics & syllabus day stuff
2. What is Unix & Linux (and why do we care)?
3. Live demo: command line interfaces, the shell, and Linux commands

What is 391?

Learning objectives: collection of tools and topics not addressed in other classes that we think all CSE majors should know.

- often called: the “missing semester” or “missing quarter” of a CS major
- routinely cited (by alums) as one of the most helpful classes for future software engineers, data scientists, etc.
- not a required class for the major or a prerequisite for any class; the department offers it because **students ask for it!**

Bottom line: **this class exists for your benefit!**

What is 391?

Logistically,

- 1-credit, credit/no-credit class, assessed on weekly homework
  - this class doesn't impact your GPA but could have other implications
- meets once a week (right now!)
- uses:
  - cs.uw.edu/391 for course materials
  - Ed discussion board for online questions
  - Gradescope for submitting homework
  - Canvas for tracking progress

Getting Help: Office Hours & Ed

Synchronous help: come to office hours!

- can be: content questions, homework help, diving deeper, ...
- can also be: just do the homework and have someone around

Asynchronous help: post on the Ed Discussion board!

- involves your solution code? make a private post.
- feel free to answer each other's questions!

Getting Help: Everything Else

Discuss something that requires privacy? Feel free to sign up for a meeting.

In particular:

- happy to work with you to navigate difficult personal situations
- feel that you're slipping? Let me know **early!**
- not sure? feel free to ask (politely) – the worst I'll say is no

Bottom line: **please reach out!**

You get what you put in

I am not expecting this class to be your #1 priority in life! But in order to learn the material you will need to:

- **actively engage** (watch the videos, come to class, do the HW)
- **plan & prioritize** (submit the homeworks – no extensions!)

This class should not be a major source of stress/work but, there's a ton of room to explore if you're interested

Operating Systems?

In the study of operating systems, a **kernel** is the program that manages software & hardware resources such as memory, networking, user input, etc.

Kevin's laptop runs ChromeOS and his phone runs Android: both of these operating systems are **based on the Linux kernel**.

macOS and iOS are **Unix-like** but not based on the Linux kernel.

Windows is not Unix-like.

Operating Systems?

What are the most common operating systems for…

- personal computers?
  - **~46% Android + ~19% iOS + ~5% macOS** as of 2024 (StatCounter) – **all Unix-like**
- server computers?
  - **~88% Unix** as of 2023 (W3 Techs)
- supercomputers?
  - **100% Linux** out of the Top 500 fastest supercomputers since 2017 (Top500)

Shell?

Despite differences, we often use computers via a **desktop environment** that helps you run apps and edit files using a mouse, keyboard, or touchscreen.

You've learned Java programming, which lets you compute new things. But it's often uncomfortable writing Java programs that interact with files or run apps.

e.g. printing out the contents of a file can take 20 lines of Java code

The **shell** is a more direct way to write programs that interact with files.

e.g. the same task can be done in 1 line with the `cat` shell command

Why Unix & Linux? (1/2)

Introduced many key ideas:

- **combining many little programs to solve one big problem**
- multiple users (with different permissions)
- hierarchical file system (with recursive folders)
- “everything is a file”
- documentation included with software
- also: shell scripts, coroutines, pipelines, regular expressions, ...

Why Unix & Linux? (2/2)

Widely impactful & influential:

- **easily the most deployed operating system in the world**
- now is realized as:
  - macOS & iOS (Unix -> BSD -> FreeBSD -> Darwin)
  - Android (uses Linux kernel)
  - Linux “distributions”, like Ubuntu, Debian, Arch, Red Hat, and Amazon Linux
  - (and integrated with Windows via WSL)
- among other things, powers most web servers & cloud computing

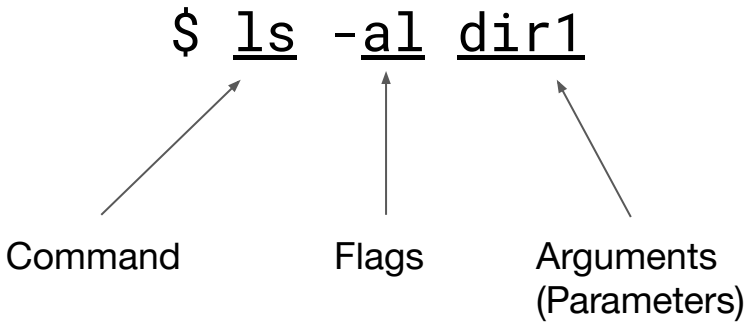
Bottom line: almost impossible to “dodge” Linux as a programmer!

the rest of this is a live demo :)

Appendix: Commands

Anatomy of a Command

Command line arguments surprisingly can be inconsistent, but for this class we'll split them up into the **command**, optional **flags**, and optional **arguments.**



Command Line Arguments

Much like methods in Java (or other languages), commands take "**arguments**".

For example, the following line lists all the files inside of the `dir1` folder:

```
$ ls dir1
```

Here, we'll say that `dir1` is an argument to `ls`. Arguments are separated by spaces.

Flags are a special type of argument that typically change the behaviour of a program slightly; they are usually prepended with a `-` or –. For example,

```
$ ls -al dir1
```

Combines two flags: the `-a` flag (list hidden files) and the `-l` flag (use long-list format).

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 Conventions

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)

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 when moving, copying, or deleting files!
(the man pages have flags that prevent this behaviour!)

Text Editors

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

In many instances, you'll be interacting with a Linux system only through a console – so a command-line text editor is necessary.

vim is both powerful text editors used by many experienced Linux users; it's up to you to pick one for 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`	Undo