CS 326: Operating Systems

Lecture 1

Welcome to CS 326!

- Looking forward to a great semester!
- *Lecture:* **MW** · 9:55am – 11:40am · **LS** 307
- *Lab Session:* **Monday** · 12:30pm – 2:00pm · **LS** G12

Staff

- **Instructor**: Matthew Malensek
  - mmalensek@usfca.edu
  - Office Hours: **W, Th** 1:00pm – 2:00pm in **HR** 407B
- **TA**: Anson Lee
  - Office Hours: TBA

So… What's an OS?

- Chances are, you already know of at least one operating system…
  - …or even a few…
- …but what do they do? And more importantly, what's the point of having one? Why should we even care?

Operating Systems

- The OS abstracts the underlying hardware and exposes a clean, easy-to-use interface to software applications
- What's the point of having one?
  - Without the OS, you don't have things like:
    - Multitasking
    - Hardware drivers
    - File systems
- Why should we even care?!
  - If you understand how the OS works, you will work more efficiently, and the *software you write for it* will be better!

Today's Agenda

- Syllabus
- Introduction to Operating Systems

Today's Agenda

- **Syllabus**
- Introduction to Operating Systems

Quick Note: Prerequisites

- CS 220 or CS 221 with a C or better
  - If it's been a while since you've programmed in C, plan on spending some time refreshing your memory!
- CS 245 with a C or better
  - Strictly enforced

Updates & Communication

- I will regularly update the **schedule** page on the course website
  - https://www.cs.usfca.edu/~mmalensek/cs326
  - Upcoming topics, readings, due dates
  - Lecture slides, videos
- Grades will be posted on Canvas
- Project submissions: GitHub
- If you need help: CampusWire
  - Q&A link on the website takes you there

You Will Learn

- How to work effectively from the command line
- Operating system components and their implementations
  - Threads, processes, synchronization primitives, system calls, scheduling, virtual memory, file systems
- How to develop UNIX systems software
- UNIX tools
- Debugging complex systems and low-level software

Course Overview

1. Command line, C refresher
2. Booting up and processes
3. System calls
4. Inter-process communication
5. CPU Scheduling
6. Memory management
7. Disk I/O
8. Networking, containers, distributed systems, etc…

Books

- *xv6: a simple, Unix-like teaching operating system*. Russ Cox, Frans Kaashoek, and Robert Morris.
- *Operating Systems: Three Easy Pieces*
  - Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau
  - Free and available online as a PDF!
    - http://pages.cs.wisc.edu/~remzi/OSTEP
- *Extreme C*. Kamran Amini.
- *The C Programming Language, 2nd Edition*. Brian W. Kernighan and Dennis M. Ritchie

Course Structure

- There's a decent amount of theory in CS 326, but we'll spend a lot of time applying it in practical situations
  - We'll be writing low-level code and working with the command line extensively
- Assignments: labs and ~4 large projects
- Assessment: quizzes every ~3 weeks
- And there's also a lab session on Mondays…

Lecture vs. Lab

- In lecture, we'll discuss new concepts and work through code together
  - Quizzes will be held during the lecture as well
- The lecture is also where we will introduce new lab assignments and projects
- In the lab, you'll work on a weekly **lab assignment** and turn in a **lab report**.

Grade Distribution

- Labs: 40%
  - Implementation: 25%
  - Lab Reports: 15%
- Quizzes: 40%
  - Individual Quizzes: 35%
  - Group Quizzes: 5%
- Projects: 20%

Lab Assignments

- Intended to be completed in class and checked off interactively by course staff
- Two parts:
  - Implementation
  - Report

Quizzes

- Given roughly every three weeks
- Helps us go back and review what we've covered
- The strategy to do well is:
  1. Understand the questions on your lab reports
  2. Take note of concepts that are **demoed** in class.
     - Sometimes questions are even revealed in advance…
  3. Review the slides.
  4. If something isn't clear, review the lecture videos and book chapters

Group Quiz

- After the regular quiz, you'll take it again
  - this time as a group (probably 4-5 students)
- Good way to review and discuss the problems

Final ~~Exam~~ Quiz

- There is no final exam, but there *is* a final quiz
  - Slightly larger than the usual quizzes
- Review only (no new material)
- I will drop your lowest quiz score
  - Do well throughout the semester and you won't have to take the final quiz!

Projects

- CS 326 projects can be time-consuming
  - Start them early
- Remember, C can be much more difficult to work with
  - You might only need to implement, X, Y, and Z.
    Really, that's not tooSEGFAULT SEGFAULT
    SEGFAULT
    - BUS ERROR
- *Plan* out your design, work methodically, and debug as you go

Academic Honesty

- **Do not cheat**. Review the Honor Code, and if in doubt about whether or not something is cheating, ask the professor. A few instances that are considered cheating include:
  - “Collaboration” that involves sharing code or solutions.
  - Submitting work or code that you did not write, create, or understand.
  - Working so closely with a classmate or TA that the work no longer represents your own individual contribution.
- Use AI to **learn**. Assignments will specify what usages are approved. If in doubt, cite it.

Late Policy

- Due dates are posted on the course schedule page. Assignments must be submitted and graded interactively by the due date.
- Lab reports are due on the day they are assigned. Late lab reports are not accepted.
- Lab implementations and projects may be submitted and graded late for up to **7 days** past the deadline with *no penalty*. Afterward, no credit will be given unless prior arrangements were made.

Today's Agenda

- Syllabus
- **Introduction to Operating Systems**

Your OS

- You'll be working on your own OS, based on **xv6**
  - …more on that later!
- It will target RISC-V CPUs and run on our virtualization/emulation server `gojira`
- So in other words, you'll be running your OS on top of another OS (Linux)

gojira

- Dual 16-core AMD Epyc CPUs
  - 32 cores
  - 64 hardware threads
- 128 GB of RAM
- Laser-guided missile defense system
- Dual cupholders

Logging in!

- I mentioned earlier that you all probably know of a few OS. Let's see how many we've all used.
- Dust off your CS credentials and log into stargate:
  - `ssh username@stargate.cs.usfca.edu`
- Next, `ssh gojira`
- If your login credentials don't work, it's time to remember them, or email support@cs.usfca.edu
- Ok, let's see what this machine can do…

OS List

- `cd /tmp/cs326` (I will create this directory now)
- Create a file named after your CS username with:
  - `touch $(whoami)`
- Open it in an editor (`micro`, `vim`, `nano`, `emacs`, ...)
- Write some info about yourself:
  - Preferred Name / pronouns
  - Your favorite color
  - A list of operating systems you've used
  - Rating on a scale from 1-10 for pineapple pizza

An Example

```
Name: Matthew Malensek (he/him)
Color: Blue
OS: MS-DOS, Windows 3.1, Mac OS 8, Windows XP
PineapplePizza: 8
```

Next

- After you're finished, see if you can figure out how to:
  1. View who is logged into `gojira`
  2. See the *processes* currently running on the machine
- Then we'll do some thorough and very scientific analysis on the files to see what kind of OS users this class has

Wow Nice Job, Matthew 💩

- Well, that was pointless
  - or was it?
- We just saw:
  - UNIX's multi-user capabilities
  - Remote logins
  - The command line
  - File creation, modification
  - Shell pipelines
  - Permissions handling
  - And a whole bunch of processes running it all

Operating Systems

- Let's get back to business. What **is** an OS?
  - An OS provides a layer of abstraction between your computer's hardware and software
- Computer science is all about layers of abstraction
  - Nobody wants to program in binary, so we have assembly
    - Nobody wants to program in assembly, so we have C
      - Nobody wants to manage their own memory, so there's Java
        - Nobody wants to write another AbstractFactoryBuilderPattern, so there's Python

Virtualization

- The basic concept of OS is **virtualization**
- Rather than exposing the hardware directly to programs, a nice set of virtual interfaces are provided
  - Razer Viper Ultimate Mouse  USB Mouse
  - Samsung Evo 1 TB SSD  Mass storage device
  - etc.
- Programs don’t have to worry about what’s going on in the system
  - Appear to have their own CPU, memory, etc.

Our Focus

- This course will cover two domains:
  - User space – most OS include bundled programs. These really aren't much different than programs we've already written
  - Kernel space – provides the functionality that makes **virtualizing** the hardware possible
    - Exposes an API to user space: **system calls**
- We'll write code for both!

Wrapping Up

- Welcome to class (again!)
  - Ask questions, and come to instructor/TA office hours. We're here to make sure you succeed.
  - Don't take the projects in this course lightly (especially if you haven't taken a 300-level yet)
    - The work can sneak up on you if you don't stay on top of it.