Basics

• Instructor: Yipeng (Roger) Song
  • I go by Roger 😊

• Email
  • Instructor: songyip@oregonstate.edu

• Office Hours: TBD @ TBD

• Requirements: Laptop/PC

• Programming Language: C, Assembly
More Basics...

• Be respectful (Establishing a Positive Community)
• Have a growth mindset
  • Most abilities could be developed through dedication and hard work
• Don’t cheat (0 tolerance!!)
  • [https://studentlife.oregonstate.edu/studentconduct/faculty-info](https://studentlife.oregonstate.edu/studentconduct/faculty-info)
• Be Proactive
  • Take control and cause something to happen, rather than just adapt to a situation or wait for something to happen
Course Description

• Goal: Learn how modern operating systems work

• Lecture & Lab
  • Learn high-level fundamental concepts of OS in the lecture
  • Practice engineering details with Labs
  • You will build your operating system (JOS)
  • Lab sessions: TAs will help you

• Topics
  • Virtual memory, Segmentation, Paging
  • Process, Isolation, Kernel, User
  • Interrupt, Exceptions, Synchronization, Concurrency
  • Filesystem
  • etc.

Image from: https://www3.nd.edu/~pbui/teaching/cse.30341.fa17/
Course Description

- **Goal:** Learn how modern operating systems work
- **Lecture & Lab:**
  - Learn high-level fundamental concepts of OS in the lecture
  - Practice engineering details with Labs
- **You will** build your operating system (JOS)
- **Lab sessions:** TAs will help you

**Topics**
- Virtual memory, Segmentation, Paging
- Process, Isolation, Kernel, User
- Interrupt, Exceptions, Synchronization, Concurrency
- Filesystem
- etc.

Image from: https://www3.nd.edu/~pbui/teaching
Course Description

• Goal: Learn how modern operating systems work

• Lecture & Lab

• Learn high-level fundamental concepts of OS in the lecture

• Practice engineering details with Labs

• You will build your operating system (JOS)

• Lab sessions: TAs will help you

• Topics

  • Virtual memory, Segmentation, Paging

  • Process, Isolation, Kernel, User

  • Interrupt, Exceptions, Synchronization, Concurrency

  • Filesystem

  • etc.
Course Objective

• Understand how modern computer systems work (in detail)
• Be able to answer the following questions:
  • What happens when we turn on the computer? How does it boot?
  • How an OS runs an application?
  • How an OS runs application that requires memory more than its physical memory?
  • How multiple applications can run on the system?
  • How an OS enforces privilege separation?
  • How an OS protects itself from malicious software?
  • How multiple programs synchronize each other? How can we implement a lock?
Important Links

• Website: https://canvas.oregonstate.edu/courses/1912331
• Instructor: Yipeng Song(songyip@oregonstate.edu)
  • I go by Roger 😊
• TAs:
  • Arman Hastings, Peiyuan Chen, Zexian Li (graduate TAs)
  • Caden Burke, Emmanuel Moncada, Alvin Johns II, Miles Wedemeyer (undergraduate TAs)
• GitHub classroom
• Discord: https://discord.gg/kJMsmvtbF
• Assignment server: os2.engr.oregonstate.edu, os1, oldos1, oldos2
Course Structure

- 10 weeks schedule
  - Virtualization (Week 1-5)
  - Concurrency (Week 6-9)
  - Persistency and others (Week 10)
- Textbook
  - [http://pages.cs.wisc.edu/~remzi/OSTEP](http://pages.cs.wisc.edu/~remzi/OSTEP)

- **Read**: prep materials posted on “Reading Materials”
- **Watch**: Lab tutorials
- **Study**: study JOS labs (tutorial videos / lab instructions)
- **Engage**: Lectures in person and office hours on Discord, discuss with peers!
Meeting Time

• Lectures (in person)
  • Attendance: Optional, but strongly recommended
  • TR 8:30 – 9:50 am at LINC 228
  • Recordings from Spring 2021 are posted

• My office hour
  • TBD

• Recitations: No in-person attendance, we will use recitation times as TA office hour as well 😊
TA Office Hours

• Office hours starts from Week 2
• Available via Discord (in person?)
• When? See Canvas main page: 
  https://canvas.oregonstate.edu/courses/1912331
Grading

• 70% JOS lab assignment
  • Lab 1 (10%), Lab 2 (15%), Lab 3 (20%), Lab 4 (25%)

• 30% Quizzes (mini-exam) (10% each)
  • Quiz 1 (4/25) : Virtual Memory
  • Quiz 2 (5/16): System calls, faults, and exceptions
  • Quiz 3 (6/1): Concurrency

Grading Scheme (tentative):
100 >= A >= 93 (96 for graduate students)
93 > A- >= 90 (93)
90 > B+ >= 86 (89)
86 > B >= 83 (86)
83 > B- >= 80 (83)
80 > C+ >= 76 (79)
76 > C >= 73 (76)
73 > C- >= 70 (73)
70 > D+ >= 66 (69)
66 > D >= 63 (66)
63 > D >= 60 (63)
F < 60 (63)
The Lab (70%)

• Four labs
  • JOS Lab 1 (10%): Booting a PC (2 weeks, due on 4/17)
    • Bootloader, protected mode, etc.
  
  • JOS Lab 2 (15%): Memory Management (2 weeks, due on 5/1)
    • Virtual memory, paging, etc.
  
  • JOS Lab 3 (20%): User Environment (3 weeks, due on 5/22)
    • Process, user, kernel, system call, etc.
  
  • JOS Lab 4 (25%): Preemptive Multitasking (3 weeks, due on 6/11)
    • Implementing context switching, multi-core support, inter-process communication, etc.
How to Conduct Lab Assignments?

• Visit Lab Tutorial Webpage
  • [https://canvas.oregonstate.edu/courses/1912331/pages/labs](https://canvas.oregonstate.edu/courses/1912331/pages/labs)

• Watch Lab Tutorial Video
  • I will explain necessary stuff for the lab assignments in the video (code/examples, etc.) and also give some tips...
An Exercise Example in Lab 1

Exercise 3. Take a look at the lab tools guide, especially the section on GDB commands. Even if you’re familiar with GDB, this includes some esoteric GDB commands that are useful for OS work.

Set a breakpoint at address 0x7c00, which is where the boot sector will be loaded. Continue execution until that breakpoint. Trace through the code in boot/boot.s, using the source code and the disassembly file obj/boot/boot.asm to keep track of where you are. Also use the x/i command in GDB to disassemble sequences of instructions in the boot loader, and compare the original boot loader source code with both the disassembly in obj/boot/boot.asm and GDB.

Trace into bootmain() in boot/main.c, and then into readsect(). Identify the exact assembly instructions that correspond to each of the statements in readsect(). Trace through the rest of readsect() and back out into bootmain(), and identify the begin and end of the for loop that reads the remaining sectors of the kernel from the disk. Find out what code will run when the loop is finished, set a breakpoint there, and continue to that breakpoint. Then step through the remainder of the boot loader.
The Lab Could be Difficult

• Coding KERNEL code in C
  • Any memory error -> Triple fault...

• Use GDB for debugging OS Kernel
  • Get familiar to tools ASAP..

• Assembly Languages
  • Intel x86

• Control hardware specific data
  • Page table
  • Global descriptor table (GDT)
  • Interrupt descriptor table (IDT)
The Lab Could be Difficult

- Coding KERNEL code in C
  - Any memory error -> Triple fault...

- Use GDB for debugging OS Kernel
  - Get familiar to tools ASAP..

- Assembly Languages
  - Intel x86

- Control hardware specific data
  - Page table
  - Global descriptor table (GDT)
  - Interrupt descriptor table (IDT)
The Lab Could be Difficult

- Coding KERNEL code in C
  - Any memory error -> Triple fault...

- Use GDB for debugging OS Kernel
  - Get familiar to tools ASAP..

- Assembly Languages
  - Intel x86

- Control hardware specific data
  - Page table
  - Global descriptor table (GDT)
  - Interrupt descriptor table (IDT)
The Lab Could be Difficult

• Coding KERNEL code in C
  • Any memory error -> Triple fault...

• Use GDB for debugging OS Kernel
  • Get familiar to tools ASAP..

• Assembly Languages
  • Intel x86

• Control hardware specific data
  • Page table
  • Global descriptor table (GDT)
  • Interrupt descriptor table (IDT)
The Lab Could be Difficult

- Coding KERNEL code in C
  - Any memory error -> Triple fault...
- Use GDB for debugging OS Kernel
  - Get familiar to tools ASAP.
- Assembly Languages
  - Intel x86
- Control hardware specific data
  - Page table
  - Global descriptor table (GDT)
  - Interrupt descriptor table (IDT)
The Lab Could be Difficult

- Coding KERNEL code in C
  - Any memory error -> Triple fault...
- Use GDB for debugging OS Kernel
- Get familiar to tools ASAP
- Assembly Languages
  - Intel x86
  - Control hardware specific data
  - Page table
  - Global descriptor table (GDT)
  - Interrupt descriptor table (IDT)

Attend lectures and watch lab tutorial videos on time and ask TAs for help!
Lab Rules

• **DO NOT SHARE YOUR CODE WITH OTHER STUDENTS**
  - *You are encouraged to discuss with others* about the assignments but do not ask/give the code to the others
  - *Do not copy* other students’ code or code available in online
  - *Do not publish* your code online

• You will be asked to submit a simple write-up for the assignment
  - Describe how you solve each exercise/questions
  - Mention your collaborators in the write-up
  - *Do not copy* other students’ write-up
  - *Do not publish* your write-up online
Lab Rules

• Plagiarism will be punished via the Office of Student Life..
  • E.g., getting F or zero point for the lab assignment that matters with plagiarism...

• Please refer the Code of Student Conduct
<table>
<thead>
<tr>
<th>Date</th>
<th>Lecture Topics(s)</th>
<th>Slides (Videos are from Spring 2021)</th>
<th>Lab Tutorial</th>
<th>Reading Materials</th>
<th>Due</th>
</tr>
</thead>
<tbody>
<tr>
<td>Week #1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/4 Tue</td>
<td>Course Intro</td>
<td>• Lecture 1</td>
<td>• Lab Setup</td>
<td>• Intro to OS 📚  • x86 Assembly 📚  • GDB Tutorial 1 📚</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• recorded.lec</td>
<td>• Slides</td>
<td>• GDB Tutorial 2 📚  • GDB cheat sheet 📚  • tmux cheatsheet 📚</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Video</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Lab Setup Guide</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week #2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/6 Thu</td>
<td>BIOS, Booting, and CPU</td>
<td>• Lecture 2</td>
<td>• x86 Address Translation</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• recorded.lec</td>
<td>• Address Spaces 📚  • Address Translation 📚</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• Segmentation 📚</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/11 Tue</td>
<td>Virtual Memory</td>
<td>• Lecture 3</td>
<td>• Paging and Intro</td>
<td></td>
<td>Lab 1 Due Monday 4/17 11:59 pm</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• recorded.lec</td>
<td>• Page Table 📚</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week #3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/13 Thu</td>
<td>Paging and Virtual Memory Translation</td>
<td>• Lecture 4</td>
<td>• Paging: Intro</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• recorded.lec</td>
<td>• Paging: TLBs 📚  • Paging: Smaller Tables 📚</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/18 Tue</td>
<td>Virtual Memory Layout</td>
<td>• Lecture 5</td>
<td>• Paging: Virtual Memory Layout</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• recorded.lec</td>
<td>• Paging: TLBs 📚  • Paging: Smaller Tables 📚</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/20 Thu</td>
<td>JOS Memory Management</td>
<td>• Lecture 6</td>
<td>• Paging: Virtual Memory Layout</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• recorded.lec</td>
<td>• Paging: TLBs 📚  • Paging: Smaller Tables 📚</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Week #4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/25 Tue</td>
<td>QUIZ</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4/27 Thu</td>
<td>Quiz 1 Review</td>
<td>• Lecture 7</td>
<td></td>
<td></td>
<td>Lab 2 Due Monday 5/1 11:59 pm</td>
</tr>
<tr>
<td>Week #5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Lab Rules – Late Submissions

• If you submit your assignment before the due date, then
  • You will get 100% based on the grading result

• If you submit your assignment within one week after the due date, then
  • You will get 75% based on the grading result

• If you submit your assignment more than one week after due, but before 6/14 11:59 pm, then
  • You will get 50% based on the grading result (75% for lab 4)
CS 544 Students

- Will have higher grade bar than CS444 (+3 pts)
  - E.g., A – 93 and over for CS 444, and 96 and over for CS 544

- *Note: I do round 😊
  - i.e. 89.45 → 89.5 → 90

Grading Scheme (tentative):
- 100 >= A >= 93 (96 for graduate students)
- 93 > A- >= 90 (93)
- 90 > B+ >= 86 (89)
- 86 > B >= 83 (86)
- 83 > B- >= 80 (83)
- 80 > C+ >= 76 (79)
- 76 > C >= 73 (76)
- 73 > C- >= 70 (73)
- 70 > D+ >= 66 (69)
- 66 > D >= 63 (66)
- 63 > D >= 60 (63)
- F < 60 (63)
Tips to the Lab

• Study in a group (**discussions are highly encouraged**)!
  • But please **write the code individually**!

• Follow tutorial video

• Ask questions (Discord)

• Understand your time budget (debugging will take **lots of your time**)!
  • **Plan ahead** to finish the labs on time

• Learn basic tools (e.g., C, gdb, assembly, editors, tmux, etc.) ASAP
  • This will help you earn more time on doing labs...
  • [https://missing.csail.mit.edu/](https://missing.csail.mit.edu/)
  • Up to Debugging and Profiling would be helpful...
Help Hierarchy

• Reread assignment, lecture slides, labs, syllabus
• Google/Bing/Open a textbook
• Ask a friend
• Check Discord for relevant posts or create a new question
• Ask a TA
  • You can attend office hours
  • TAs will also be monitoring Discord
• Ask Roger
Others

• Be active on Discord
• Read pinned messages in each lab channel
Assignment

• Lab Setup, and Lab 1 will be posted by today

• Lab 1 tutorial video will be posted before Thursday’s lecture