CS 444/544 Operating Systems II

Prof. Sibin Mohan

Spring 2022 | Lec1: Course Intro

Adapted from content originally created by: Prof. Yeongjin Jang

Instructor:
Prof. Sibin Mohan

https://sibin.github.io/sibin

- Faculty at OSU since Fall 2021
  - At UIUC before that
- Main research areas: **systems**, **security** and **resiliency**
  - security for embedded and cyber-physical systems
  - security for drones/autonomous rovers/V2X
  - Networks
- Photography, travel, movies
- Security & Privacy Meetup on **Fridays 3:00 – 4:00 PM**.

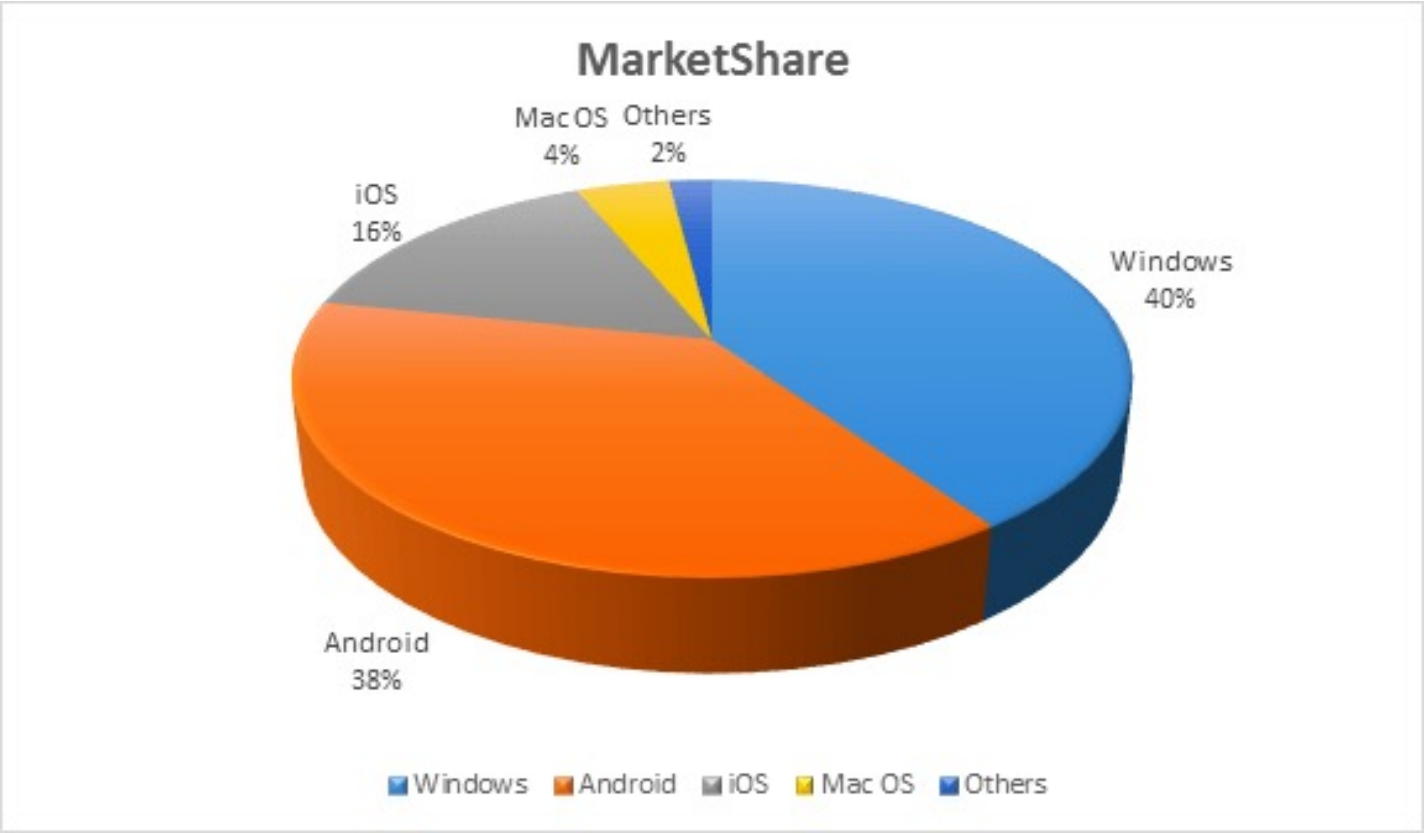
What is an Operating System [OS]?

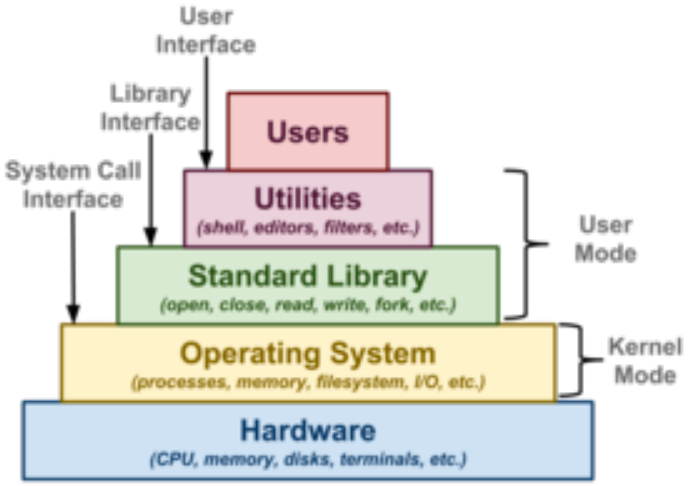
What is an OS?

- **Body of software**
- Allows users (and programs) to use the **low-level hardware**
  - Share memory, enable interactions with devices, etc.
- Manages **sharing** of resources across multiple programs
- Provides additional features like **security**, **isolation**, etc.

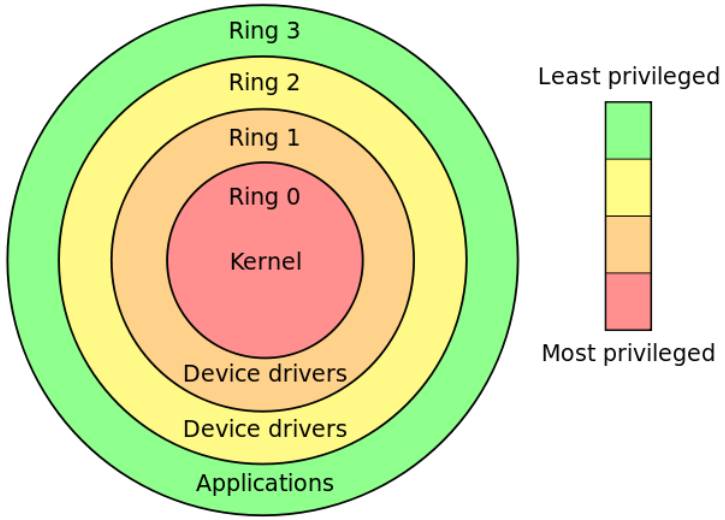
- In charge of ensuring system operates **correctly** and **efficiently**

OS Market Share

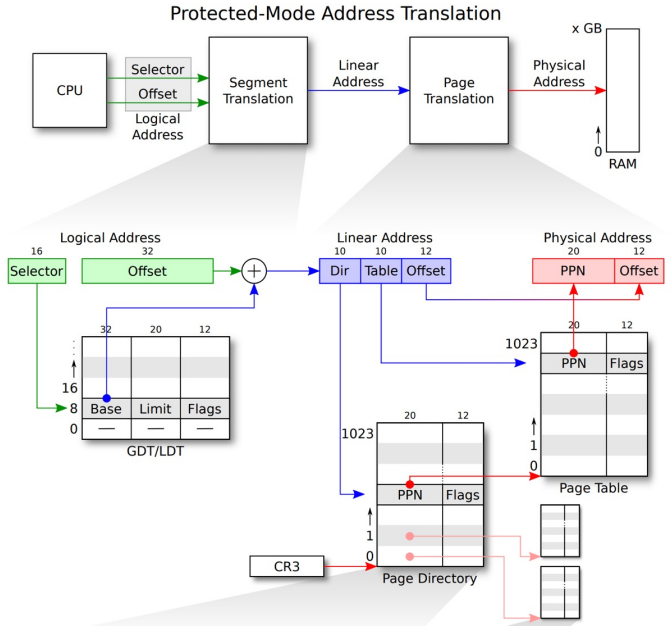




Layers/interface to **Hardware**



Rings of **Privilege**



Address Translation

Lots of concepts!

Implementation!

```
static void *
boot_alloc(uint32_t n)
{
    static char *nextfree;  // virtual address of next byte of free memory
    char *result;

    // Initialize nextfree if this is the first time.
    // 'end' is a magic symbol automatically generated by the linker,
    // which points to the end of the kernel's bss segment:
    // the first virtual address that the linker did *not* assign
    // to any kernel code or global variables.
    if (!nextfree) {
        extern char end[];
        nextfree = ROUNDUP((char *) end, PGSIZE);
    }
```

```
static inline physaddr_t
page2pa(struct PageInfo *pp)
{
    return (pp - pages) << PGSHIFT;
}
```

```
// These variables are set in mem_init()
pde_t *kern_pgdir;      // Kernel's initial page directory
struct PageInfo *pages;     // Physical page state array
static struct PageInfo *page_free_list; // Free list of physical pages
```

Course Description

Goal: Learn how modern operating systems work

Lectures & Labs

- 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 does an OS **run an app**lication?
- How does OS run application that requires **more memory than physical memory**?
- How can **multiple applications run** on the system?
- How does an OS enforce **privilege separation**?
- How does an OS protect itself from **malicious software**?
- How multiple programs **synchronize** each other? How can we **implement a lock**?

Important Links

- Website: https://sibin.github.io/teaching/cs444-osu-operating-systems/spring_2022/

- Instructor: Prof. Sibin Mohan [sibin.mohan@oregonstate.edu]
- TAs:
  - Sultan Alanazi, Peiyuan Chen, Avery Stauber [Graduate TAs]
  - Jacob Eckroth, Christian Hernickx [Undergraduate TAs]

- Gitlab: https://gitlab.unexploitable.systems
- Discord: https://discord.gg/He8YEKZF

- Assignment servers: **os2.engr.oregonstate.edu**, **os1**, **oldos1**, **oldos2**

Course Structure

- **10 weeks** of classes
  - Booting, Memory and Virtualization
  - Concurrency
  - Persistency
  - Scheduling
  - https://sibin.github.io/teaching/cs444-osu-operating-systems/spring_2022/cal.html
- Textbook: http://pages.cs.wisc.edu/~remzi/OSTEP/
- **In-person lectures & labs** (older videos will be available on YouTube)

Meeting Time (with me)

- Lecture (in-person, **synchronous**)
  - Tuesdays/Thursdays: **4:00 PM – 5:20 PM**
  - Location: **LINC 210**
  - Slides & video (old version) will be posted after 7PM

- My office hours
  - **Mondays**: **2:00 PM – 4:00 PM**

Labs [In Person]

Day	Times
Wednesday	10:00 AM – 11:50 AM
	12:00 PM – 13:50 PM
	14:00 PM – 15:50 PM
Thursday	10:00 AM – 11:50 AM
	12:00 PM – 13:50 PM

TA Office Hours

- Each TA host two sets of office hours/week
  - 2 hours of in-person
  - 2 hours scheduled time on discord

NOTE: we will try to answer your questions as soon as we can, **within reasonable expectations.**

Use the **Discord server** to ask questions or discuss among yourselves

Grading

- **70% JOS lab assignment**
  - Lab 1 (10%), Lab 2 (15%), Lab 3 (20%), Lab 4 (25%)
- **30% Quizzes** (mini-exam)
  - Quiz 1 (04/14): Virtual Memory
  - Quiz 2 (05/05): System calls, faults, and exceptions
  - Quiz 3 (05/24): Concurrency
- **All quizzes will be on CANVAS** (remote)
  - You will have up to 2 trials (I will take your best score)
  - ~60 minutes at most, but I will set the time as 120min
- NO FINAL EXAM!

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 Labs (70%)

- JOS Lab 1 (10%): **Booting a PC** (2.5 weeks, due on 04/13)
  - Bootloader, protected mode, etc.
- JOS Lab 2 (15%): **Memory Management** (2 weeks, due on 04/27)
  - Virtual memory, paging, etc.
- JOS Lab 3 (20%): **User Environment** (2 weeks, due on 05/13)
  - Process, user, kernel, system call, etc.
- JOS Lab 4 (25%): **Preemptive Multitasking** (3.5 weeks, due on 06/08)
  - Implementing context switching, multi-core support, inter-process communication, etc.

How to Conduct Lab Assignments?

Visit Lab Tutorial Webpage

https://sibin.github.io/teaching/cs444-osu-operating-systems/spring_2022/lab/lab1.html

Watch Lab Tutorial Video

It explain necessary concepts for the lab assignments (code/examples, etc.) and also share some tips…

But first...

- Necessary setup for the labs:

https://sibin.github.io/teaching/cs444-osu-operating-systems/spring_2022/lab/tools.html

- Follow **all** instructions diligently!
  - For lab setup as well as labs

We use a script for grading!

Any failures to follow instructions on your part ➔ a bad grade!

- **VERY Useful Resource**:
  - Missing Semester of CS Education: https://missing.csail.mit.edu

An Exercise Example in Lab 1

Note

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.

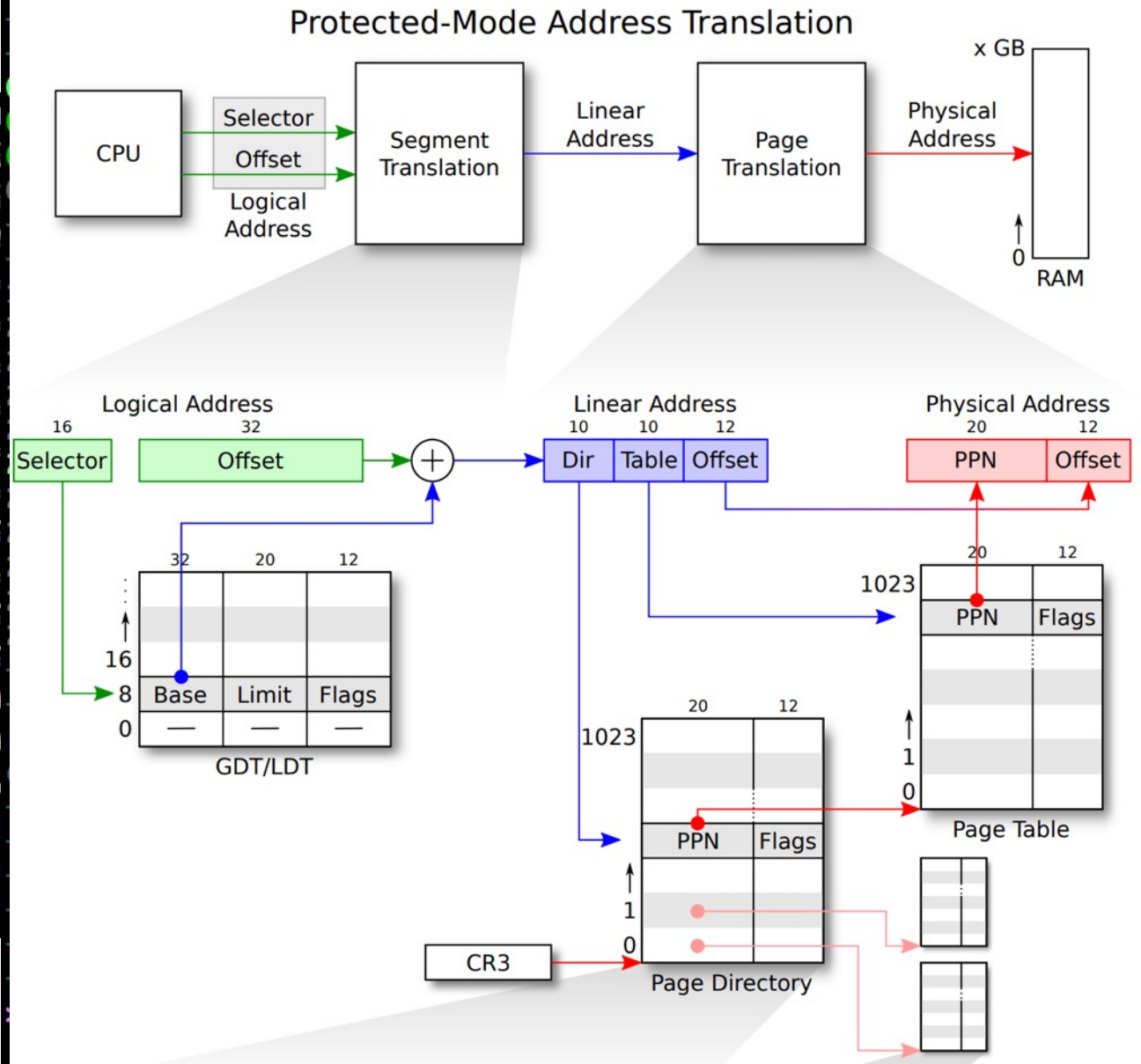
Set a breakpoint at address 0x7c00, which is where the boot sector will be loaded. Continue execution until that breakpoint. Trace through the code in `boot/boot.S`, using the source code and the disassembly file `obj/boot/boot.asm` to keep track of where you are. Also use the *x/i* command in GDB to 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

- Programming 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)

```
qem
EAX
ESI
EIP
ES
CS
SS
DS
FS
GS
LDT
TR
GDT
IDT
CR0
DR0
DR6
EFE
Tri
```



Watch both lectures and lab tutorial videos **on time** and **ask TAs for help**!

Lab Rules

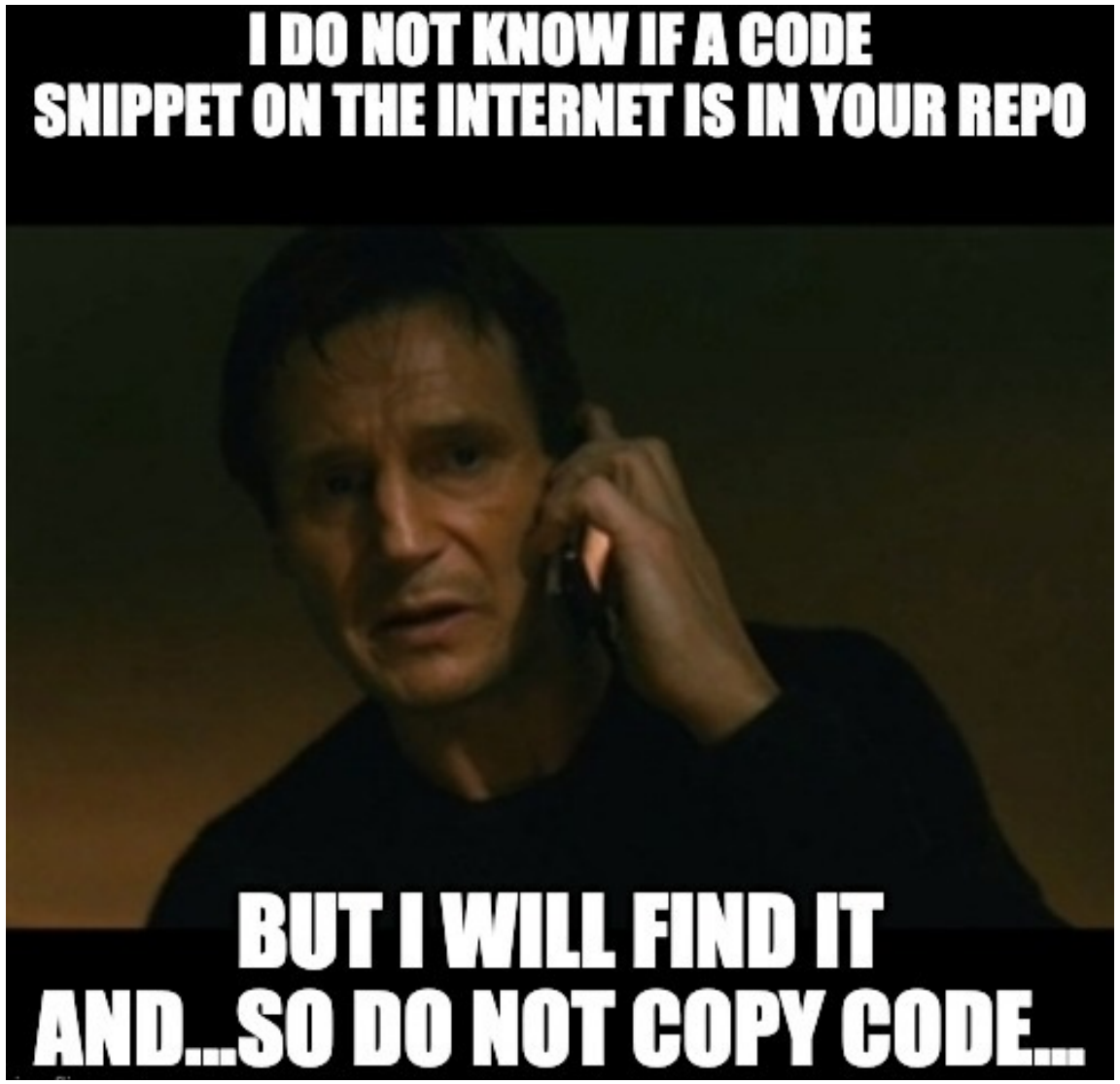
- DO NOT SHARE YOUR CODE WITH OTHER STUDENTS
  - **You are encouraged to discuss with others** about the assignments
  - Do not ask/give the code to the others
  - **Do not copy** other students' code or code available in online
  - **Do not publish** your code online

- 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

Plagiarism

- Punished via the Office of Student Life.
  - E.g., getting F or zero points for lab

- Please refer the Code of Student Conduct
  - https://studentlife.oregonstate.edu/studentconduct/academicmisconduct
  - https://studentlife.oregonstate.edu/sites/studentlife.oregonstate.edu/files/edited_code_of_student_conduct.pdf



Due Dates on the Calendar

https://sibin.github.io/teaching/cs444-osu-operating-systems/spring_2022/cal.html

Monday	Tuesday	Wednesday	Thursday	Friday
Mar 28	Mar 29 **LEC 1:** Intro to the course **Study, Lab 1:** Booting a PC **Read:** Textbook **Read:** at&t_asm GDB tutorial1 tutorial2 cheat-sheet **Read:** tmux cheatsheet (ctrl-b -> backtick) tmux-cheat-sheet **Read:** Missing Semester of CS *First day of class*	Mar 30	Mar 31 **LEC 2:** BIOS/Booting/CPU	Apr 1
Apr 4	Apr 5 **LEC 3:** Memory: Address Space, Segmentation, and Paging	Apr 6	Apr 7 **LEC 4:** Virtual Address Translation **Study, Lab 2:** Memory Management	Apr 8
Apr 11	Apr 12 **LEC 5:** Virtual Memory++	Apr 13 Lab 1 Due	Apr 14 Quiz 1: Virtual Memory	Apr 15
Apr 18	Apr 19 **LEC 6:** JOS Memory Management and Quiz 1 Prep	Apr 20	Apr 21 **LEC 7:** Quiz 1 Review **Study, Lab 3:** User Environment	Apr 22
Apr 25	Apr 26 **LEC 8:** User/Kernel Switch	Apr 27 Lab 2 Due	Apr 28 **LEC 9:** Handling Interrupts/Exceptions	Apr 29

May 2	May 3 **LEC 10:** System Calls and Page Fault	May 4	May 5 Quiz 2: System calls, faults, and exceptions	May 6
May 9	May 10 **LEC 11:** Virtualization Recap and Quiz 2 Prep	May 11	May 12 **LEC 12:** Multi-threading and Synchronization **Study, Lab 4:** Preemptive Multitasking	May 13 Lab 3 Due
May 16	May 17 **LEC 13:** Lock and Synchronization	May 18	May 19 **LEC 14:** Concurrency Bugs and Deadlock	May 20
May 23	May 24 Quiz 3: Concurrency	May 25	May 26 **LEC 15:** Schedulers	May 27
May 30	May 31 **LEC 16:** Schedulers (contd.)	Jun 1	Jun 2 **LEC 17:** Final Summary	Jun 3 *The last day of class*
Jun 6	Jun 7 Final exam week (NO FINAL!)	Jun 8 Lab 4 Due	Jun 9 Final exam week (NO FINAL!)	Jun 10 **DUE:** All labs (50% for all others)

Late Submissions

If you submit your assignment **before** the due date, then

You will get **100**% of credit from ‘make grade’ of your submission

If you submit your assignment within **one week** after the due date, then

You will get **75%** of credit from ‘make grade’ of your submission

If you submit your assignment **before 06/10 11:59 pm**, then

You will get **50%** of credit from ‘make grade’

CS 544 Grades

- Higher grade bar than CS444 (+3pts)
  - CS 444: A [>93]
  - CS544: **A [>96]**

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 for the Labs

- Study in groups (discussions are highly encouraged!)
  - But please **write the code individually**!
- Follow tutorial videos
- Ask questions on Discord (preferred), Canvas
- Understand your time budgets (debugging takes a **lots of 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/
  - Up to Debugging and Profiling would be helpful

Notices

- My office hours: **Mondays [2:00 – 4:00 PM]**

- TA office hours will be posted soon (as a calendar)

COVID-19 Class Safety Policy

- Masks are welcome but no longer required
- Be respectful/safe