CS202 (003): Operating Systems Intro

Instructor: Jocelyn Chen

Most of the materials covered in this slide come from the lecture notes of Mike Walfish's CS202

Course Staff

- Instructor: Jocelyn Chen
- TA: Bob Yao
- Course webpage: See your Brightspace course page
- Course webpage contains syllabus, important information about assignment policy, handouts, reading, etc.

About this Course

Learn how operating systems work

Learn key abstractions and concepts in operating systems

These materials will be useful beyond OSes!

Understanding resource management, abstractions, design tradeoffs in large-scale systems

Textbooks

- *Operating Systems: Three Easy Pieces*, by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau.
- *Operating Systems and Middleware: Supporting Controlled Interaction*, by Max Hailperin.
- *Computer Systems: A Programmer's Perspective*, Third Edition, Randal E. Bryant and David R. O'Hallaron.
- *The C programming language (second edition)*, Brian W. Kernighan and Dennis M. Ritchie.

- Textbooks are not substitute for lectures!
- Class presentation may not follow the books
- Skip many chapters and cover extra materials

Campuswire

- We will be using Campuswire for all course-related discussions
- **Make sure you can access Campuswire! (link available on webpage)**
- Please use Campuswire (**with posts, not DMs**) for class-related questions. Please use common sense when posting questions: hints/ideas ok, but **cannot post full solutions**. If you include **code**, please mark your question **private**.
- Please use **DMs** (through Campuswire) for more personal questions. We will **ignore** emails about course administration and other class-related questions.
- We will response in 12-24 hours. Don’t expect us to answer questions minutes before the due time.

Office Hours

Name	Time	Location
Bob	TBD	TBD
Jocelyn	Wed 4-5 PM (This week only)	Remote (This week only)

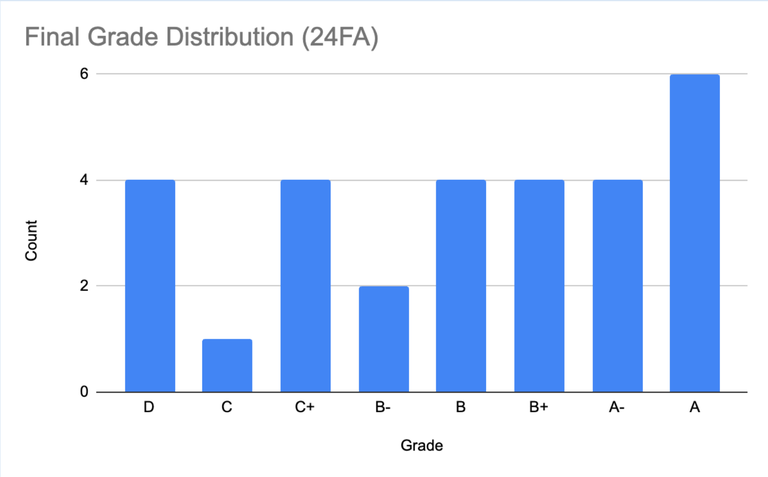
Please go to Bob for any debugging/programming/lab related questions!

Please go to Jocelyn for any course material/logistics related questions!

Periodically check course website and Campuswire for OH update!

Course Grades

- Exam: collectively 60% of final grade
- Labs: 25% of final grade
- Quizzes: 15% of final grade
- Final grades **will be curved**



- You can submit any graded item for a regrade, under the following conditions:
  - You submit a clear, written statement that explains the request
  - You submit your request within one week of when the graded work was returned
  - We will regrade the entire exam, quiz, etc. (so a regrade might decrease your grade)

Lab Policy

- Lab must be submitted by **midnight** at the **end** of due date
- Late labs will be accepted until **midnight** a **week after the due date (end of day)**
- Late labs incur penalties. However, you have **5** slack days that forestall the penalty clock.
- We will **ignore all extension requests** for reasons such as job interviews, work on research publications, etc. unless it is explicitly stated here
- You will get a **0** if you either do not hand in the assignment, or hand in a blank assignment.

Quiz Policy

- Happen In-class every Thursday (unless announced separately)
- Quiz questions will focus on recent course materials and labs
- Each quiz will take 15 minutes and in the form of multiple choice questions
- No electronics or cheatsheet allowed. **You will need a pencil.**
  (**Except this Thursday's quiz** — it will happen on Gradescope so you will need your laptop)
- There will be **no make-ups** of any form. We will **drop your three lowest score**

Collaboration Policy

- All assignments (labs) must be **done on your own**. That means,
  - **Not allowed** to do assignments in groups
  - **Not allowed** to check solutions with each other
  - **Not allowed** to discuss problems with each other through other channels that course staff does not have access to. You can discuss questions (in general terms) through Campuswire
  - **Not allowed** to discuss/show/debug code with any person other than the instructor and the TA
  - Collaboration with other students on assignments is considered **cheating**

Collaboration Policy, cont.

- You **may not** use any AI-Assisted code writing tools (such as Copilot) in this course.
- You **may not** use any AI-Assisted tools for written assignments and non-coding lab questions.
- You **may** use AI-Assisted chatbots for Lab 2 and onwards (more on this next slide)
- You **may not** look at, or use, (similar) solutions from prior years on the web, or seek assistant from the Internet.
- You **must** acknowledge your influences (either from any person you discussed with, websites, AI assistants, or any other sources). You **must** declare what ideas are borrowed from the source
- You **must** take reasonable steps to protect your work. You **must not publicize** your solutions in this semester or any future semester

What you **can** do with AI-assisted Chatbots

- **General Programming Language Queries:**
  - “How to declare a struct in C?”
  - “How do I traverse a linked list?”
- **Standard Library Functions:**
  - “What is the string length function in <string.h>?”

These two are the **only usages** allowed!

What you **cannot** do with AI-assisted Chatbots (include but not limited to)

- **Lab-Specific Code Completion**: "[lab instructions] [code snippet] Can you help me complete the code/give me a skeleton of the implementation/pseudocode?"
- **Code Style Improvement**: "Please improve the coding style of the following code using the guideline [coding style rubric]"
- **Lab Question Answers**: "[lab question]"
- **Debugging Assistance**: "[your code] I get the following error [error], what is going on?"
- **Algorithm Explanation**: "[code snippet from the lab] What is this part of the code doing?"

Honor Code

- Failing to adhere to the collaboration and integrity policy is **a violation of the NYU honor code**
- We take the honor code **extremely seriously**! If you cheat, you will own the consequences
- If you are unsure whether a particular source of external information is permitted, **contact the instructor** before looking at it
- Please make sure you read the policy page and understand **everything** in it. **Do not** wait until you violate the policy and then say “I thought this sentence means …”
- We may overweight the exam score or conduct oral exam if cheating is suspected.

More on Labs

- Lab is a **crucial component** of the operating system course. You will spend substantial time programming
- **Please start early**! The labs will take more than you think
- We are eager to help you but **please make sure you are making good use of our time** (i.e. please think a while before asking any questions, make sure there is no similar questions on Campuswire)
- Make sure you check out Setup, *The Missing Semester of your CS education*, and *Unix dev tools* before working on the labs

Practice Questions

- Practice questions is intended to reinforce the course material
- They will be released every week for your practice. **No submission required**
- We will release the solutions of the practice questions as well (usually ~1-2 weeks later)

Let’s get started!

What is an operating system?

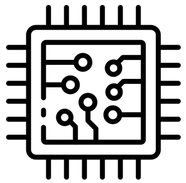
"Operating System is a program that abstracts and manages hardware resources for user programs."

Application

Web Browsers
Games
Office Softwares
…

Operating System

File System
Processes
Virtual Memory
…



Hardware

Disk
CPU
RAM
…

What is an operating system?

Job 1: managing the resources of the machine

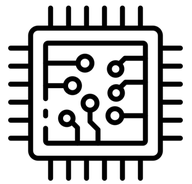
Scheduling: give each process some of the CPU

Virtual Memory: give each process some of the physical memory

Make sure one program won't screw up another (through multiplexing, isolation, protection, sharing)

Operating System

File System
Processes
Virtual Memory
…



Hardware

Disk
CPU
RAM
…

What is an operating system?

Job 2: Abstracting the hardware

(You do not want to program on hardware!)

Hide details of hardware for convenience and portability

Allow programs to run unaware of the hardware details

Application

Web Browsers
Games
Office Softwares
…

File System
Processes
Virtual Memory
…

Examples of resource management and abstraction

File Systems

Abstraction

"File is a continuous array of bytes" is a false illusion

Data in a file can be fragmented across different locations on the disk.

```
fd = open("/tmp/foo", WR_ONLY)
rc = write(fd, "abc...z", 26)
```

*"**tmp/fo**" abstracts actual location on the storage device*

*"**WR_ONLY**" simplifies access control*

*The **write** call abstracts the entire process of writing data to storage.*

Isolation

User program cannot write to a file unless it has permission

Examples of resource management and abstraction

Text Input

Abstraction

"Input from any source (a soft keyboard displayed on a touch screen, a physical keyboard, etc.) act the same" is a false illusion

There are significant differences in how different inputs are processed.

Programs are not aware of your input method

Isolation

Ensure that keystrokes go to a single program

Otherwise, would other application know your passwords?

Examples of resource management and abstraction

Memory

Abstraction

```
movl 0x1248 %rdx
```

“It is reading from memory address 0x1248”
is a false illusion
“0x1248” is a virtual address!

“the computer has a linear contiguous address space”is a false illusion
Physical memory can be fragmented and spread across different locations

Isolation

User program can’t write to another user’s memory

Examples of resource management and abstraction

Scheduling

Abstraction	Isolation
“this process is running continuously” is a false illusion In fact, processes are actually being rapidly started and stopped by the operating system.	User program that is hogging CPU gets switched out in favor of another user’s program

In this course

What is a operating system?

How does OS abstract hardware resources?

Why does OS provide these abstractions?

Why study operating system?

Even modern LLM systems have deep operating system requirements:
managing thousands of GPUs efficiently requires sophisticated resource allocation and scheduling to optimize training and inference performance

It is essential to know “how things work”,
especially given that your program will probably run on top of an operating system!

OS demonstrates fundamental design tradeoffs between performance, security and usability,
and that’s relevant across all of computer sincere

Many critical problems in operating systems, such as security vulnerabilities and concurrency,
remain active areas of research and development.

A little bit history on OS...

Operating System Emerges to Improve Throughput

To run a program, a deck of punched cards containing the program was loaded into a machine's memory.

This program was the only thing that ran on that computer until the program terminated.

When the job completed, an operator would load a program to dump memory and would remove the tape, cards, and any printout.

After that, the next job would be loaded on the computer.

https://people.cs.rutgers.edu/~pxk/416/notes/01-intro.html

History of Unix

In 1969 Kenneth Thompson and Dennis Ritchie developed the UNIX operating system.

Goal: Run on computers of all sizes

PDP-11: the smallest system in a decade that can run Unix

Written in assembly language

4th Edition (1973):
first version written in C

Its abstractions are still in use everywhere

https://www.historyofinformation.com/detail.php?id=872
https://en.wikipedia.org/wiki/PDP-11#/media/File:Pdp-11-40.jpg

Separate Strand: personal computer

monday afternoon

december 9

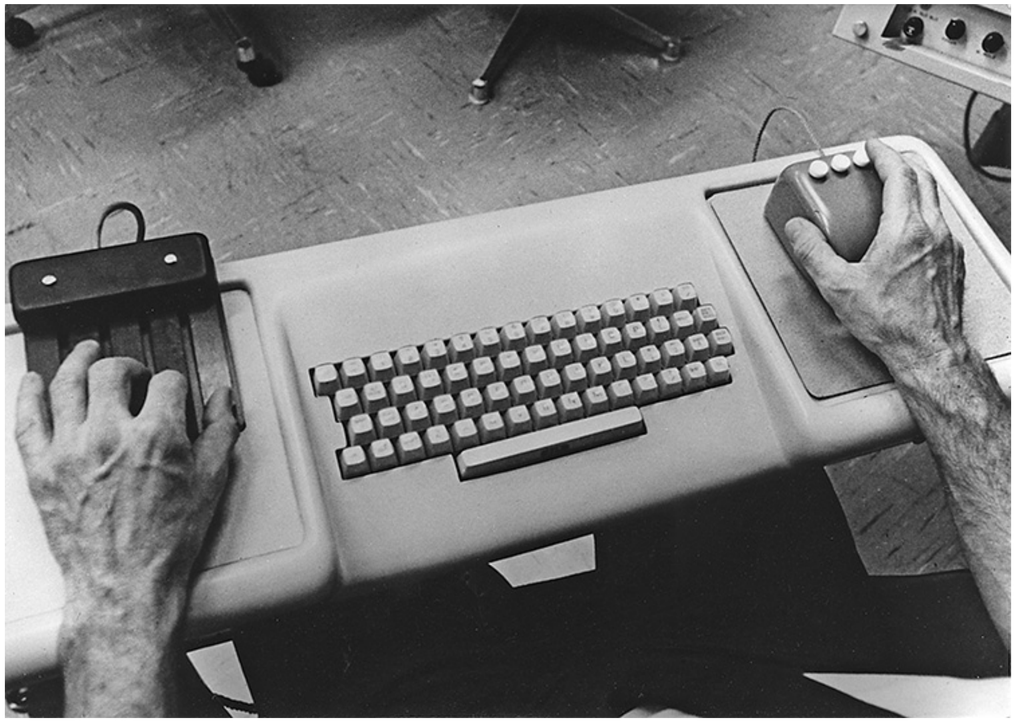
3:45 p.m. / arena

Chairman:
DR. D. C. ENGELBART
Stanford Research Institute
Menlo Park, California

a research center for augmenting human intellect

This session is entirely devoted to a presentation by Dr. Engelbart on a computer-based, interactive, multiconsole display system which is being developed at Stanford Research Institute under the sponsorship of ARPA, NASA and RADC. The system is being used as an experimental laboratory for investigating principles by which interactive computer aids can augment intellectual capability. The techniques which are being described will, themselves, be used to augment the presentation.
The session will use an on-line, closed circuit television hook-up to the SRI computing system in Menlo Park.
Following the presentation remote terminals to the system, in operation, may be viewed during the remainder of the conference in a special room set aside for that purpose.

Poster for the “mothers of all demos”



NLS’s 5-button chord keyset, a standard QWERTY keyboard, and 3-button mouse

Xerox Alto

https://invention.si.edu/invention-stories/mother-all-demos
By Joho345 - Own work, Public Domain, https://commons.wikimedia.org/w/index.php?curid=3699855

Lab 0 and Lab 1 are released today!

Notes on Lab 0

This is really just the setup of the subsequent labs

You will need to know/learn basics of Git and Docker
(which means you need to at least read
the Git portion of https://missing.csail.mit.edu/ and What is a docker?)

Knowing these concepts will be beneficial for future software development as well!

Submission of Lab 0

Finish “Lab 0" Quiz on Gradescope

Quiz has a 30-min time limit and will due on 1/26 at 12 AM.
No slack days are permitted for this lab.

Before you open the quiz, **make sure you**:

- Understand Git and Docker
- Finish all the procedures in Lab 0
- Your laptop should have your Docker engine ready
- You are able to access GitHub

Notes on Lab 1

Lab 1 should be easier than other labs.

The lab will help you review C and teach you to use gdb

If you are unsure about your C skills, I recommend looking at K&R: The C programming language.

Your TODO list after this lecture

Make sure you know where the course webpage is (for Section **003**)

Read Policies and grading **very very very carefully (that's the quiz)**

Make sure you have access to
Campuswire, Gradescope, and Brightspace for **this section**

Checkout the Schedule for readings be completed before class