ECE469: Operating Systems Engineering

Aravind Machiry

1/11/2022

About This Course

- ECE 469 - Operating Systems Engineering
  - Undergraduate-level operating systems
  - Basic OS concepts and mechanisms + hands-on assignments
- Prerequisite:
  - ECE368 (Data Structures)
  - (optional) ECE437 (Introduction to Digital Computer Design and Prototyping)
  - Programming proficiency in C is **absolutely required**

About Me (https://machiry.github.io/)

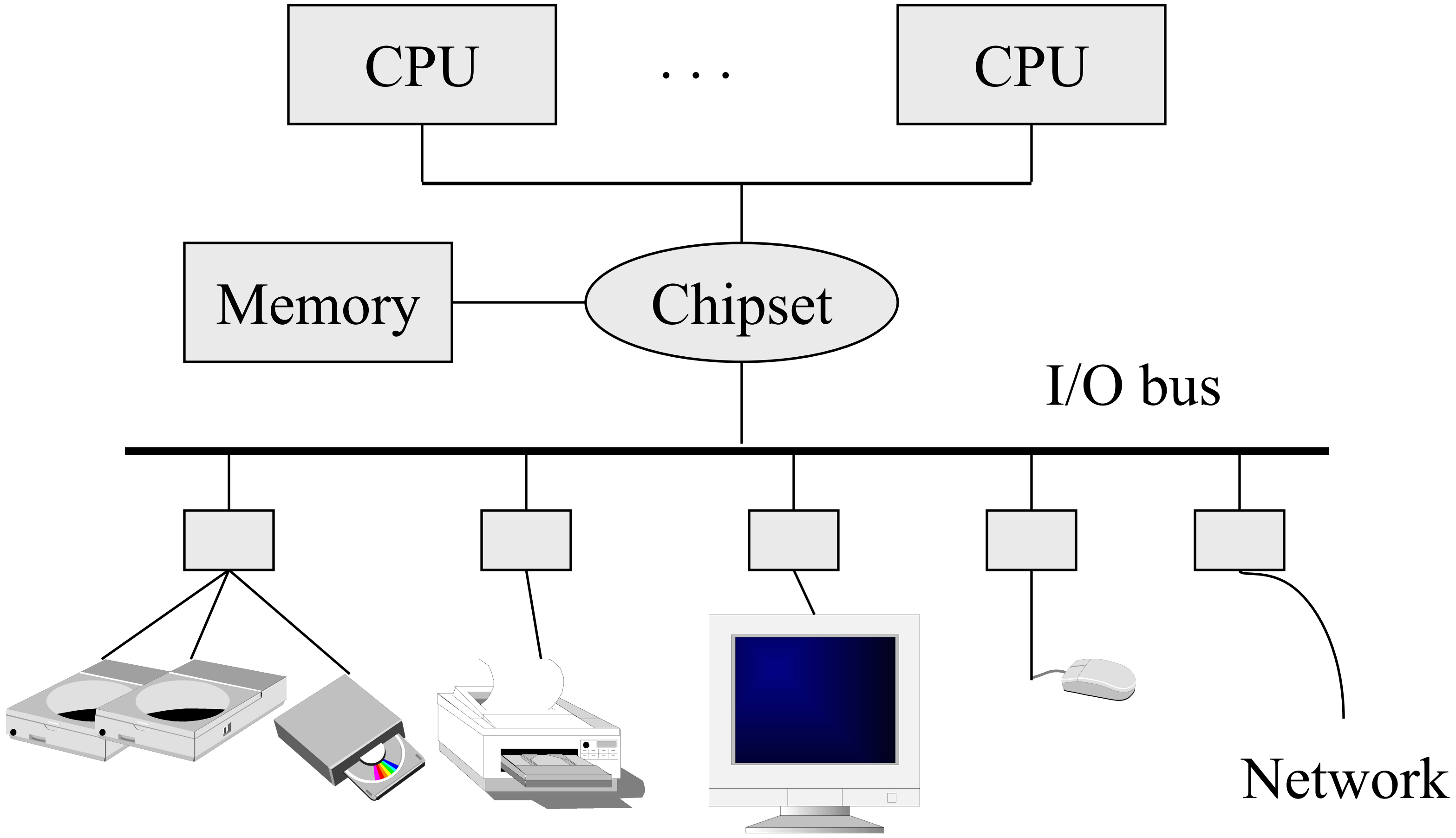
Aravind Machiry:

- Phd 2020, University of California, Santa Barbara.
- MS 2014, Georgia Institute of Technology, Atlanta.

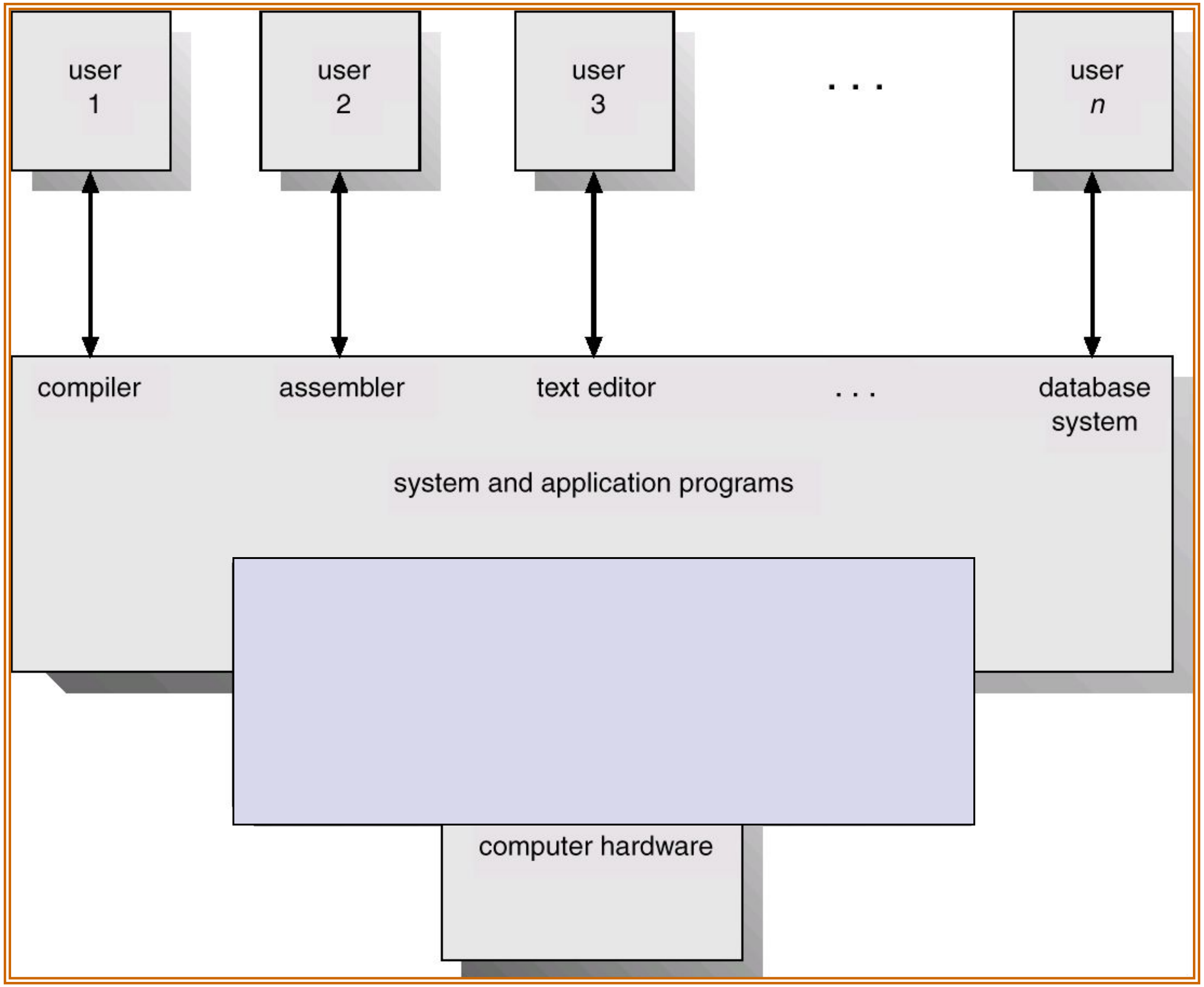
Research interests:

- PurS3 Lab: https://purs3lab.github.io/
- System Security:
  - Operating Systems and IoT devices
- Program Analysis:
  - Static and Dynamic

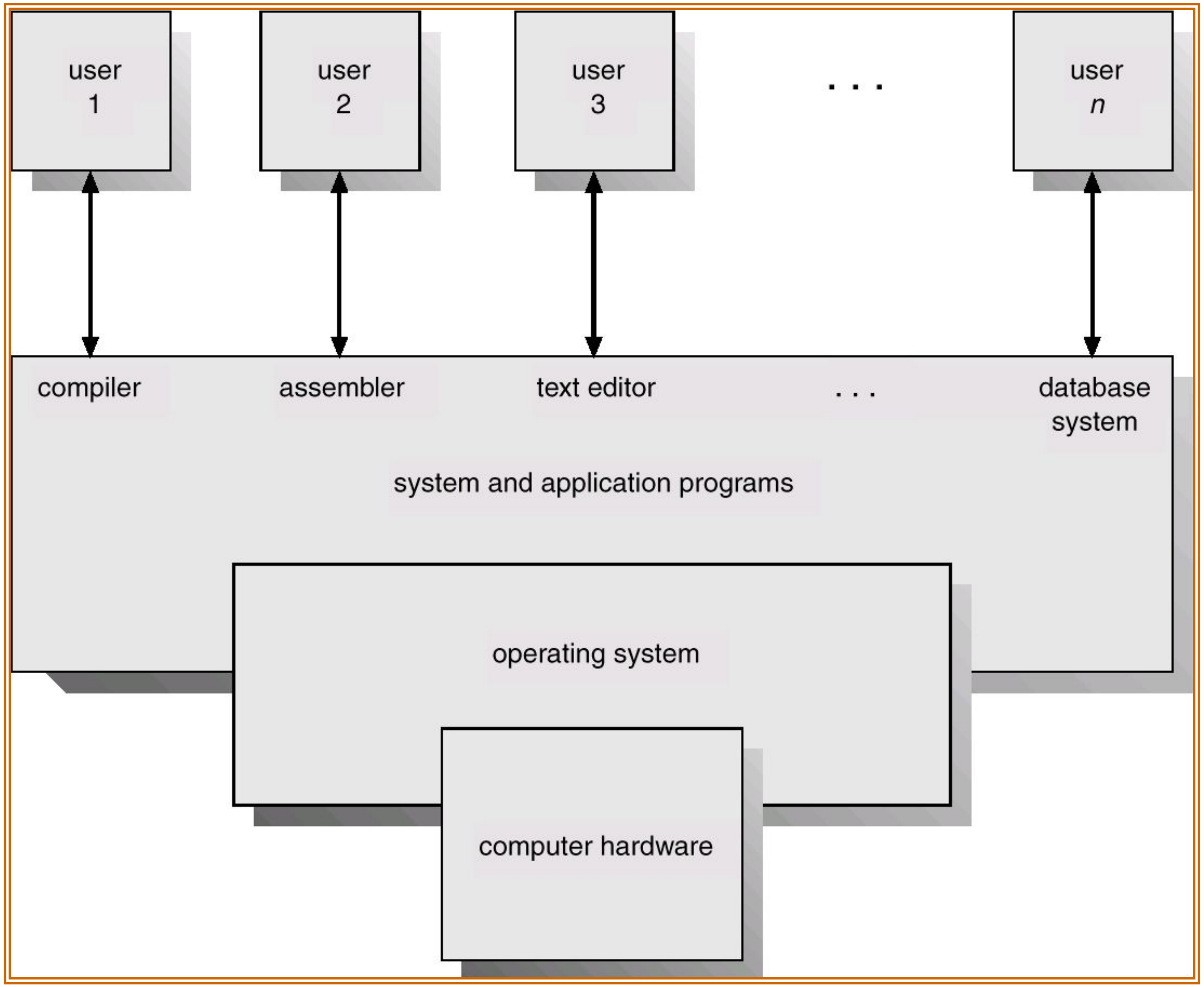
A Typical Computer from a Hardware Point of View



Computer System Components



Computer System Components



What is an OS?

"Code" that *sits between*:

- programs & hardware
- different programs
- different users

But what does it do/achieve?

What is an OS?

- Resource manager
- Extended (abstract) machine

Makes computers *efficient* and *easy* to use

What is an OS?

Resource manager (answer1)

- Allocation
- Reclamation
- Protection

What is an OS?

Resource manager

- Allocation
- Reclamation
- Protection

Finite resources

Competing demands

Examples:

- CPU
- Memory
- Disk
- Network

What is an OS?

Resource manager

- Allocation
- Reclamation
- Protection

"The OS gives
The OS takes away"

Implied at termination

Involuntary at run time

Cooperative (yield cpu)

What is an OS?

Resource manager

- Allocation
- Reclamation
- Protection

"You can't hurt me
I can't hurt you"

Implies some degree of
safety & security

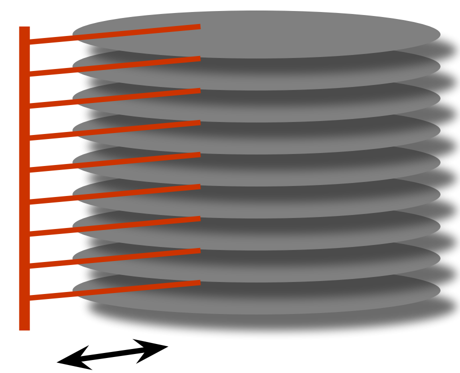
What is an OS?

Extended (abstract) machine (answer 2)

- Much more ideal environment than the hardware
  - Ease to use
  - Fair (well-behaved)
  - Supporting backward-compatibility
  - Reliable
  - Secure
- Illusion of infinite, private (reliable, secure) resources
  - Single processor → many separate processors
  - Single memory → many separate, larger memories

Example: programming hard drive

- Physical reality
  - Block oriented (e.g. 512 bytes)
  - Physical sector numbers
  - No protection among users of the system
  - Data might be corrupted if machine crashes
  - Programming:
    - Loading values into special device registers



"I will save my lab1 solution on platter 5, track 8739, sector 3-4."

Example: programming hard drive

- Physical reality
  - Block oriented
  - Physical sector numbers
  - No protection among users of the system
  - Data might be corrupted if machine crashes
  - Programming:
    - Loading values into special device registers
- File system abstraction
  - Byte oriented
  - Named files
  - Users protected from each other
  - Robust to machine failures
  - Programming
    - open/read/write/close

“I will save my lab1 solution on platter 5, track 8739, sector 3-4.”

“My lab1 solution is in ~amachiry/lab1/process.c.”

Separating Policies from Mechanisms

A fundamental design principle in Computer Science

Mechanism – tool/implementation to achieve some effect

Policy – decisions on what effect should be achieved

Example – CPU scheduling:

- All users treated equally
- All program instances treated equally
- Preferred users treated better

Separation leads to flexibility!

Is there a perfect OS?
(resource manager, abstract machine)

Efficiency

Fairness

Portability

Interfaces

Security

Robustness

- Conflicting goals
  - Fairness vs efficiency
  - Efficiency vs portablity
  - …
- Furthermore, …

Hardware is evolving…

- 60's-70's - Mainframes
  - Rise of IBM
- 70's - 80's – Minicomputers
  - Rise of Digital Equipment
- 80's - 90's – PCs
  - Rise of Intel, Microsoft
- 90's - 00's – handheld/portable systems (laptops)
- 2007 - today -- mobile systems (smartphones), Internet of Things, specialized hardware in the cloud
  - Rise of iPhone, Android, IoT

Implications on OS Design Goals: Historical Comparison

	Mainframe	Mini	Micro/ Mobile
System $/ worker	10:1 – 100:1	10:1 – 1:1	1:10-1:100
Performance goal	System utilization	Overall cost	Worker productivity
Functionality goal	Maximize utilization	Features	Ease of Use

Hardware is evolving (cont) …

- (once) New architectures
  - Multiprocessors
  - 32-bit vs. 64-bit
  - Multi-core
- New memory, storage, network devices
  - SSD, NVM, RDMA, SmartNIC
- New processors
  - GPU, TPU, FPGA

We Live in Interesting Times…

- Processor speed doubles in 18 months
  - Number of cores per chip doubles in 24 months
  - But meeting its limit!
- Disk capacity doubles every 12 months
- Global bandwidth doubles every 6 months

Performance/cost “sweet spot” constantly decaying

Applications are also evolving…

- New applications
  - Computer games, networked games
  - Virtual reality
  - Web 2.0 (search, youtube, social network, …)
  - Video streaming
  - Mobile apps (> 2.8 million iPhone, Android apps)
  - Big data
  - Machine learning, deep learning, reinforcement learning
  - Autonomous vehicles
  - …

Implications to OS Design

- Constant evolution of hardware and applications continuously reshape
  - OS design goals (performance vs. functionality)
  - OS design performance/cost tradeoffs

- Any magic bullet to good OS design?

There is no magic in OS design

This is Engineering

- Imperfection
- Tradeoffs (perf/func/security)
- Different Goals
- Constraints
  - hardware, cost, time, power
- Optimizations

Nothing's Permanent

- High rate of change
  - Hardware
  - Applications
- Cost / benefit analyses
- One good news:
  - Inertia of a few design principles

About this course…

Principles of OS design

- Some theory
- Some rational
- Lots of practice

Goals

- Understand OS design decisions
- Last piece of the “puzzle”
- Basis for future learning

To achieve the goals:

- Learn concepts in class
- Get hands “dirty” in labs

Topics we'll cover

- Memory management
- Process management
- I/O management
- A touch of advanced topics if we have time

Expect (some) pain

Somewhat fast pace

Lots of programming

Some difficult (abstract) concepts

Mechanics – Course Staff

Instructor:

Aravind Machiry, amachiry@purdue.edu, EE 333

Office hours (online): Fri 9:00 am - noon and by appt.

TAs:

Gokulan Ravi, ravig@purdue.edu

Heejin Park, bakhi@purdue.edu

Sahil Jaganmohan, sjaganm@purdue.edu

Pranab Dash, dashp@purdue.edu

TA office hours and location: Check course webpage

Mechanics – General Info

- Course home page: https://purs3lab.github.io/ee469/
- Announcements: Piazza/Brightspace
- Discussions: Piazza
- Grading: Brightspace

Mechanics – Q & A

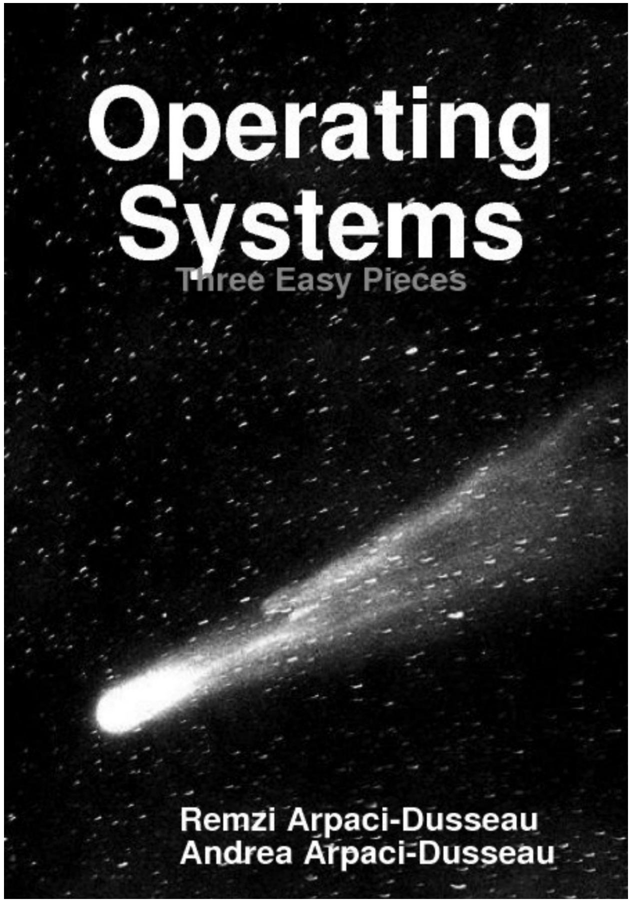
- Questions of general interests : Piazza
- Other questions : TAs (esp. grading-, lab-related) and instructor
- Announcements : Brightspace (and/or Piazza) (with email notice)

Mechanics – Textbook

Operating Systems: Three Easy Pieces, by Remzi and Andrea Arpaci-Dusseau

http://www.ostep.org

- Free online book,
- easy to understand and follow,
- useful for interview preparation too



Mechanics – Lecture Notes

When available, will be provided on the web

Not necessary self-contained, complete, or coherent

Not a substitute for lecturers

Ask questions!

Mechanics - Labs

- 5 labs.
  - **Use JOS.**
  - Build parts of a real OS.
- 1st not graded (Setup)

- 2-3 weeks each (excl. spring break)
  - explained in the corresponding first week's lab
  - due: *Schedule*

- Work in pairs (optional to work on your own)
  - Register your group on Brightspace.
  - Be decent to each other!

Mechanics - Labs

- Best Practices:
  - START EARLY!!!
  - Coding through screen sharing.
  - Debug sessions.
  - Make use of the lab sessions to ask questions.

Mechanics - Exams

- Midterm
  - before Spring break.
- Final
  - Non-cumulative
- Multiple choices, True or False, short answers, some design (derivation), very few programming problems

Mechanics – Grading

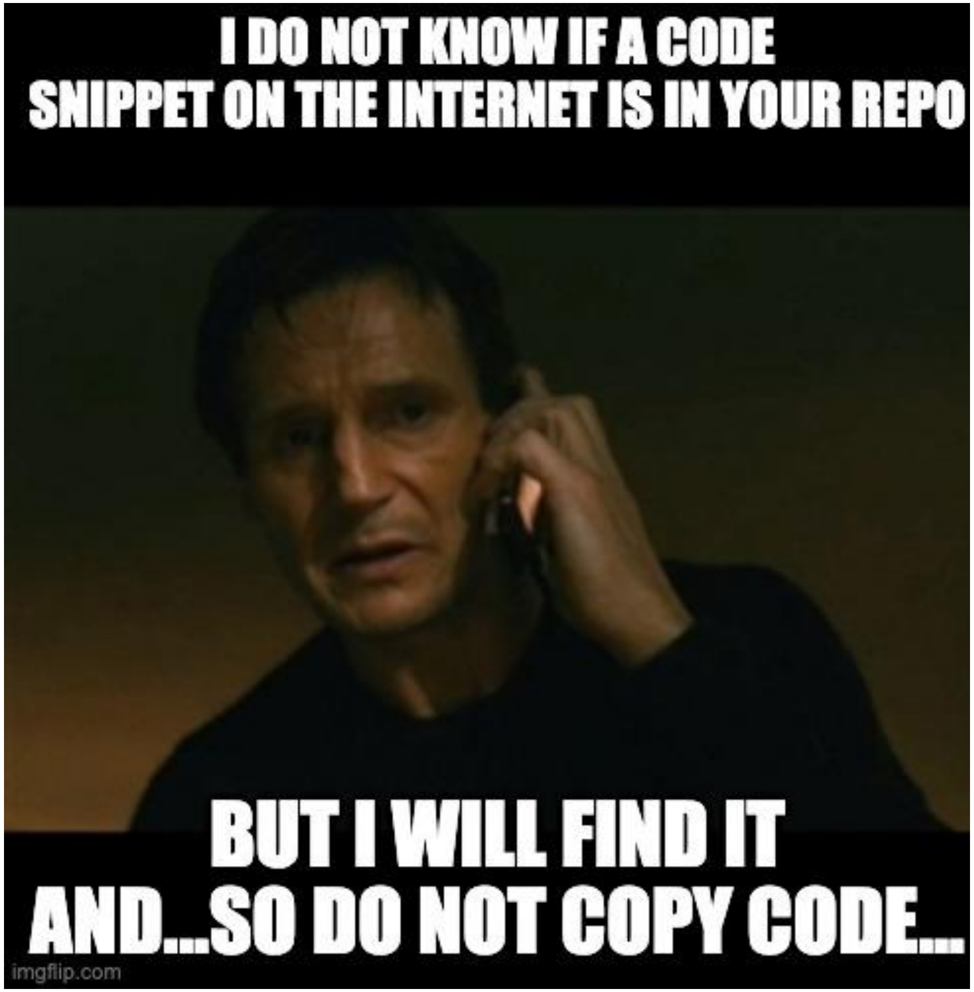
- Labs (80%)
- Midterm exam (10%)
- Final exam (10%)
- Participation + Extra credit (5 %)

- Late policy:
  - Refer: https://purs3lab.github.io/ee469/labs/

Academic Integrity

- Labs
  - Ask TAs / instructor for clarification
  - Each team must write their own solution
  - No discussion of or sharing of specific code or written answers is allowed
  - Any sources used outside of textbook/handouts/lectures must be explicitly acknowledged
  - Your responsibility to protect your files from
    - e-copying using UNIX file protection
    - public access, including disposal

Academic Integrity Policy



Academic Integrity Policy

- Cheating
  - **The first case of cheating on an assignment will result in zero for that whole assignment & reporting to university administration for disciplinary action**
  - **The second case will result in an immediate F grade for the course**

Questions?

- Reading assignment:
  - [Encouraged] Before the class.
- Find a lab partner and enroll for a group on Brightspace.
  - No later than Jan 14th
- Start lab 1 this week