CS 318 Principles of Operating Systems

Fall 2020

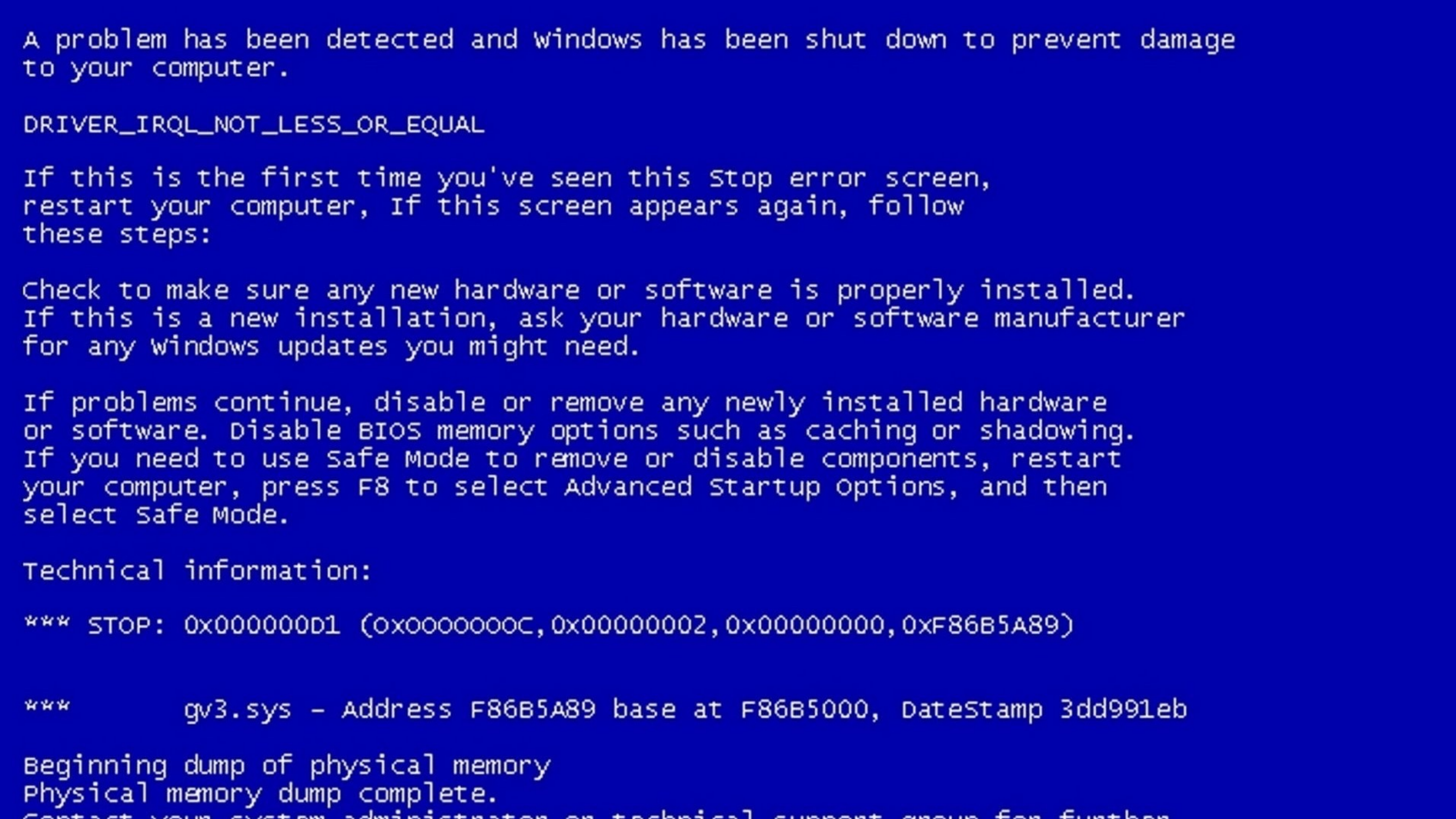
Lecture 1: Introduction

Prof. Ryan Huang

JOHNS HOPKINS
WHITING SCHOOL
of ENGINEERING

I hope you are all safe and well during these times

Virtual Greetings to You



```
A problem has been detected and windows has been shut down to prevent damage
to your computer.

DRIVER_IRQL_NOT_LESS_OR_EQUAL

If this is the first time you've seen this Stop error screen,
restart your computer, If this screen appears again, follow
these steps:

Check to make sure any new hardware or software is properly installed.
If this is a new installation, ask your hardware or software manufacturer
for any Windows updates you might need.

If problems continue, disable or remove any newly installed hardware
or software. Disable BIOS memory options such as caching or shadowing.
If you need to use Safe Mode to remove or disable components, restart
your computer, press F8 to select Advanced Startup Options, and then
select Safe Mode.

Technical information:

*** STOP: 0x000000D1 (0x0000000C,0x00000002,0x00000000,0xF86B5A89)


***       gv3.sys - Address F86B5A89 base at F86B5000, DateStamp 3dd991eb

Beginning dump of physical memory
Physical memory dump complete.
```

Virtual Greetings to You: Retry ☺

Lecture 1 Overview

- **Course overview**
- **Administrative**
- **What is an Operating System?**
- **Walk-through of OS basics**

Staff: Instructor

- **Prof. Ryan Huang**
  - Web: https://cs.jhu.edu/~huang
  - Office Hours: Tue 9:30-10:30am, Thu 4-5pm ET (or by appointment)
- **Research Areas**
  - Operating Systems
  - Cloud and Mobile Computing
  - Systems Reliability and Availability

Staff: Teaching Assistants

- **Yuzhuo Jing (TA)**
  - Office Hours: Mon 10-11am, Wed 9-10pm ET

- **Haoze Wu (CA)**
  - Office Hours: Wed 2-3pm, Fri 2-3pm ET

- **Gongqi Huang (CA)**
  - Office Hours: Tue 10:30-11:30am,
    Thu 10:30-11:30am ET

Quick Survey

- How many juniors? seniors?
- Graduate students?
- Any non-CS majors?
- Any has some prior experience with OS?

Bad News…

- **This is a TOUGH course**
- **Requires proficiency in systems programming**
  - “*Low level (C) programming absolutely necessary.*”
  - “*Need to be fearless about breaking code (and then fixing it later).*”
  - “*Need to be confident in touching and modifying large systems of code*”
- **Requires significant time commitment**
  - “*The projects are insanely time consuming*”
  - “*If you're worried about your course load this semester, maybe consider putting this class off for a later year*”
  - “*The workload is much much heavier than your average CS course…Be prepared to spend entire weeks working on nothing but the material for this course. If you start only one week in advance you WILL NOT finish without at least two all-nighters!* ”

Good News

- **There aren’t many such hardcore courses in CS curriculum ☺**
  - Typically the final checkmark for a solid CS degree
  - You don’t have to take it if you are not interested in it at all
- **It’s hard, but rewarding in the end**
  - “*The project are very hard. But completing them is very rewarding.*”
  - “*I loved this course, it was very challenging but very satisfying and I learned a lot.*”
  - “*You learn a lot about operating systems and computers in general.*”
- **A highly valued skill after graduation**
- **We will try our best to help you**

Course Overview

- **An introductory course to operating systems**
  - Classic OS concepts and principles
  - Prepare you for advanced OS and distributed system course
  - OS concepts often asked in tech interview questions

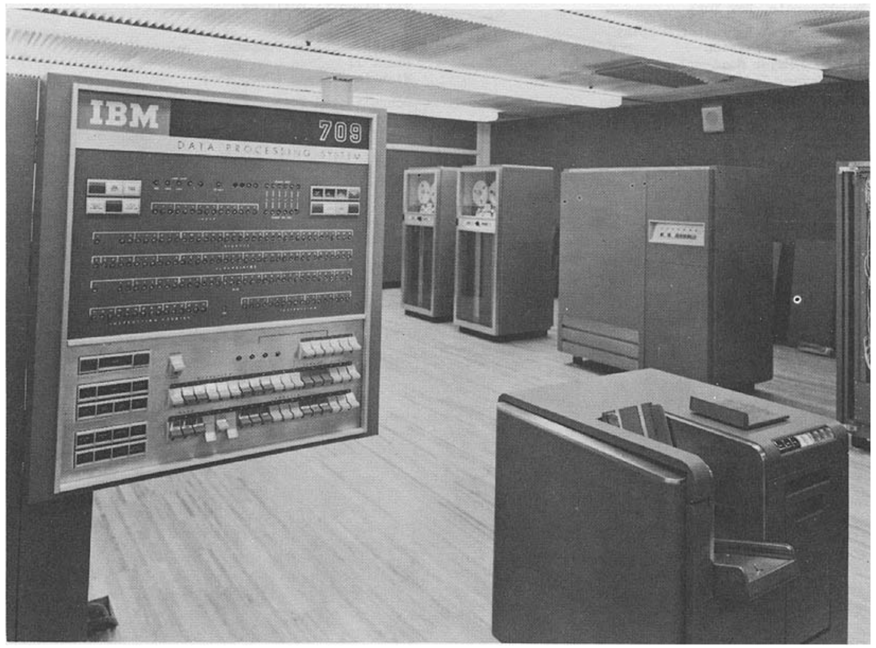
- **A practice course for hands-on experience with OS**
  - Four large programming assignments on a small but real OS
  - Reinforce your understandings about the theories

Topics Covered

- Threads, Processes
- Concurrency, Synchronization
- Scheduling
- Virtual Memory
- I/O
- Disks, Filesystems
- Protection & Security
- Virtual Machines

Why Study Operating Systems?

- **Technology trends**



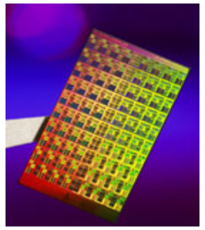
IBM 709

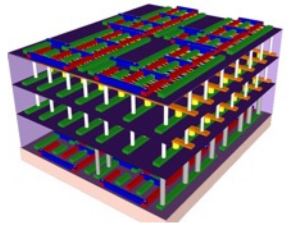
CPU: ~4000 mult/div per sec.	**CPU:** 1.85 GHz dual-core
memory: 32K 36-bit words	**memory:** 2 GB
price: $2,630,000+	**price:** $329
size: half room	**size:** 9.4 in × 6.6 in

iPad (2017)

Why Study Operating Systems?

- **Technology trends**


manycore


3D stacked chip

persistent memory

accelerators

Tensor Processing Unit

smartphones

IoT device

self-driving cars

robots

data centers

…

Why Study Operating Systems?

- **An exciting time for building operating systems**
  - New hardware, smart devices, self-driving cars, data centers, etc.
  - Existing OSes face issues in performance, battery life, security, isolation

some of you

- **Pervasive principles for systems in general**
  - Caching, concurrency, memory management, I/O, protection

many of you

- **Understand what you use**
  - System software tends to be mysterious
  - Understanding OS makes you a more effective programmer

all of you

- **Complex software systems**
  - Many of you will go on to work on large software projects
  - OSes serve as examples of an evolution of complex systems

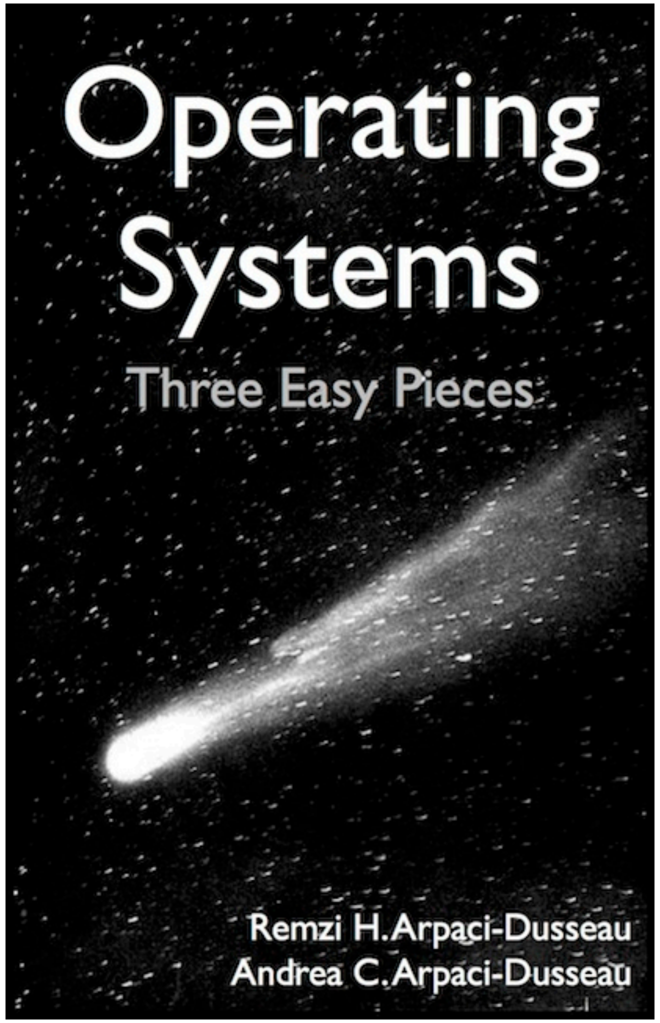
all of you

Course Materials

- **Course materials**
  - Lectures are the primary references
  - Textbooks are supplementary readings
  - Occasionally non-required papers

Textbook

http://from-a-to-remzi.blogspot.com/2014/01/the-case-for-free-online-books-fobs.html

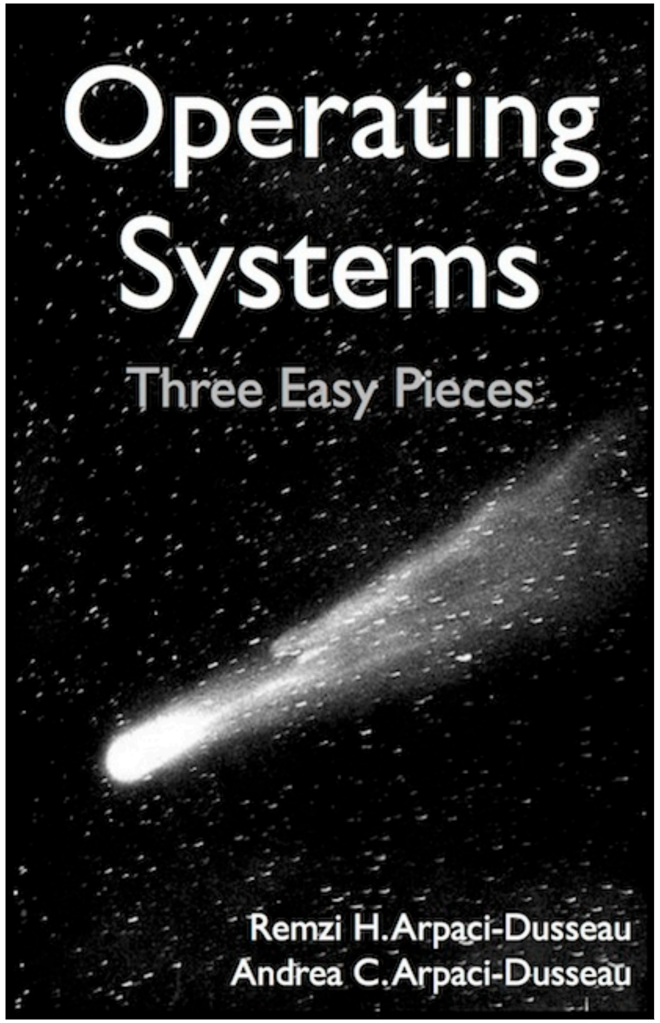


Operating Systems: Three Easy Pieces, Version 0.91

By *Remzi Arpaci-Dusseau* and *Andrea Arpaci-Dusseau*

Textbook

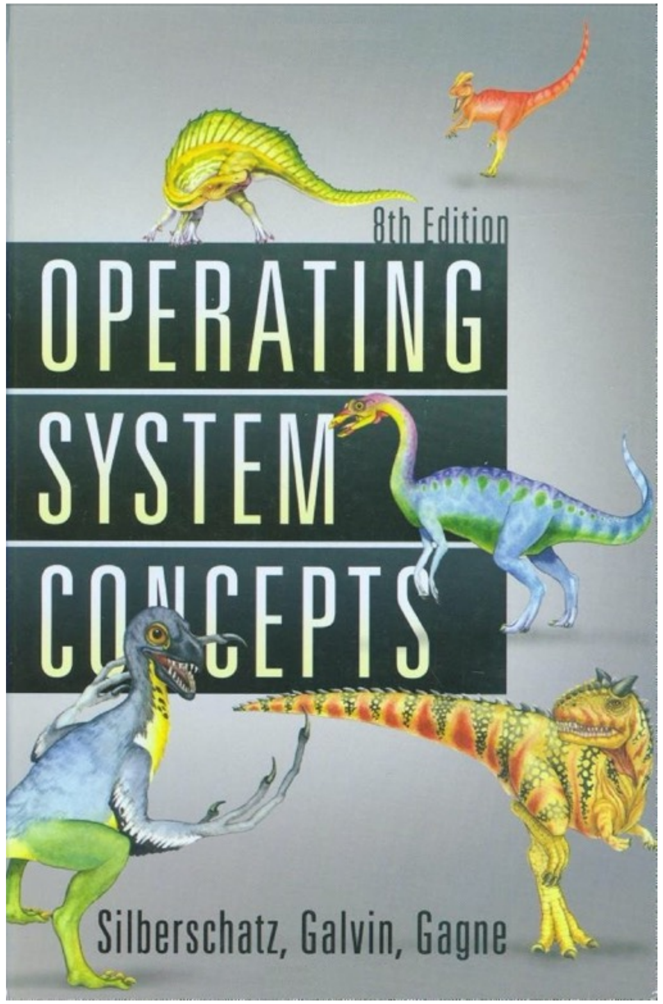
http://from-a-to-remzi.blogspot.com/2014/01/the-case-for-free-online-books-fobs.html



Operating Systems: Three Easy Pieces, Version 0.91

By *Remzi Arpaci-Dusseau* and *Andrea Arpaci-Dusseau*

Textbook

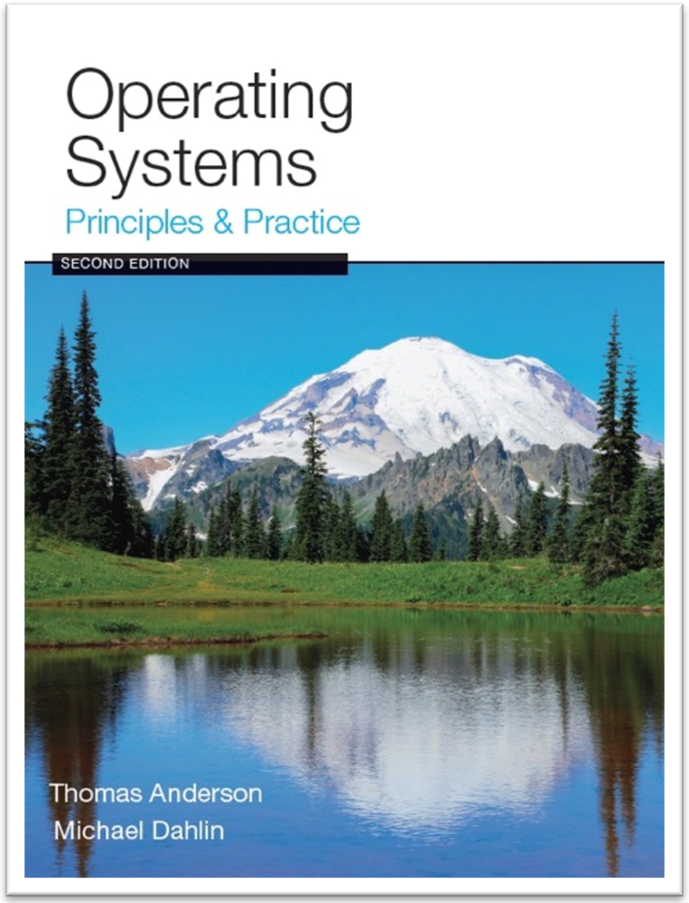


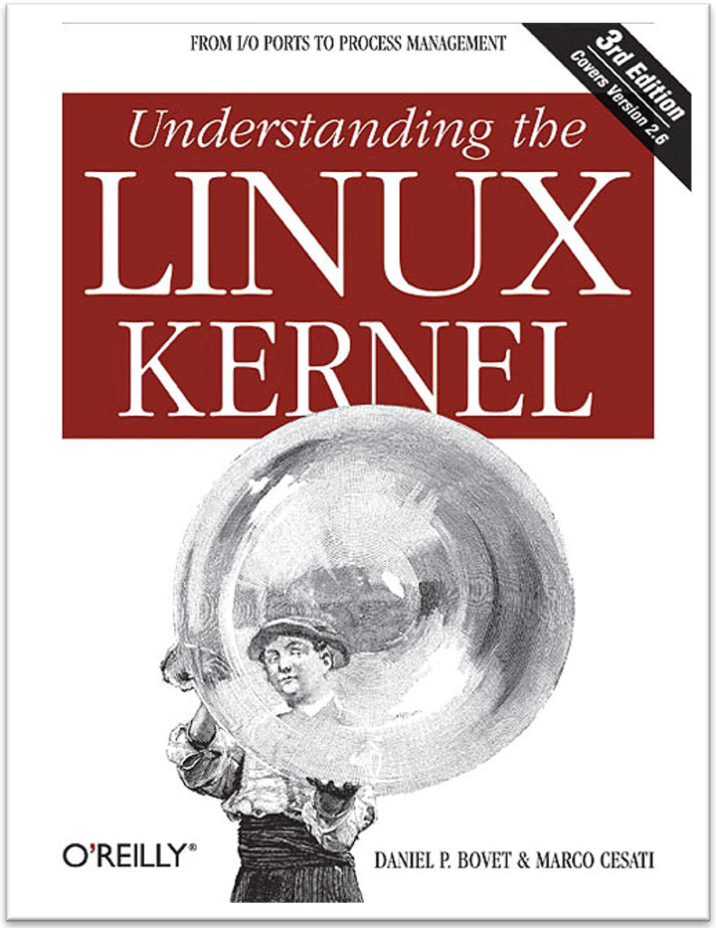
Operating Systems Concepts

By *Silberschatz, Galvin* and *Gagne*

Textbook

Other Recommended Textbooks


Operating Systems
Principles & Practice
SECOND EDITION
Thomas Anderson
Michael Dahlin


FROM I/O PORTS TO PROCESS MANAGEMENT
3rd Edition
Covers Version 2.6
Understanding the LINUX KERNEL
O'REILLY®
DANIEL P. BOVET & MARCO CESATI

Important Links

- **Course Website** (check it often)
  - https://www.cs.jhu.edu/~huang/cs318/fall20/
  - Course syllabus and schedule
  - Lecture slides
  - Homework handouts
  - Project descriptions and references
- **Discussion Forum**
  - https://piazza.com/jhu/fall2020/cs318418618 Access Code(0x7C00)
  - project, lecture, exam questions
- **Staff mail list:**
  - cs318-staff@cs.jhu.edu
  - administrative requests, sensitive questions

Homework

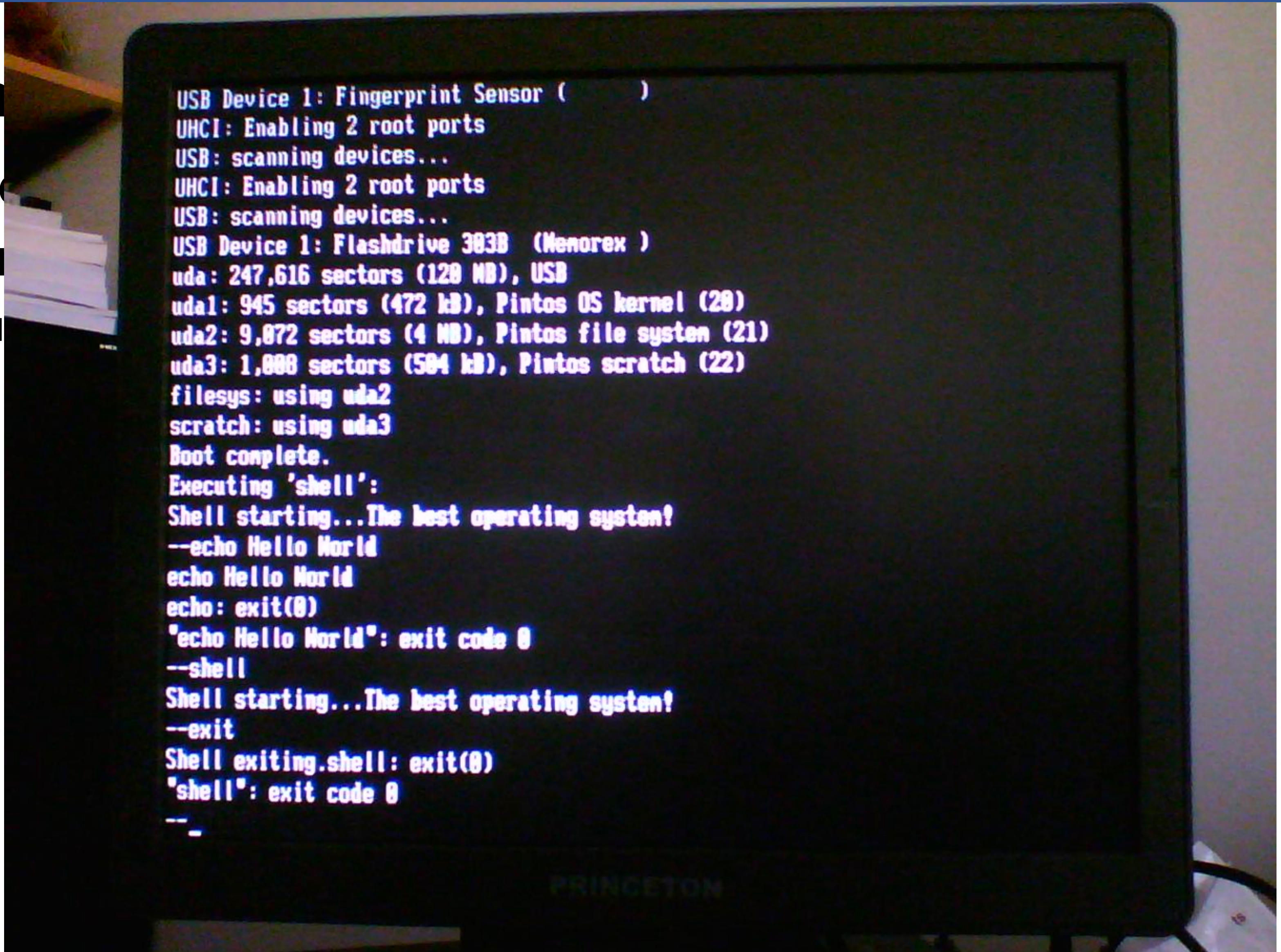
- **Five homework assignments throughout the semester**
  - help you check understanding about the lectures
  - prepare you for the exams
- **The homework assignments will *not* be graded**
  - solutions released ~a week later
  - amount learned from doing homework is proportional to effort
  - your choice on how much effort

Project Assignments

- **Implement parts of Pintos operating system**
  - Developed in 2005 for Stanford's CS 140 OS class
  - Written in C, built for x86 hardware
    - can run on a real machine!

Project Assignments

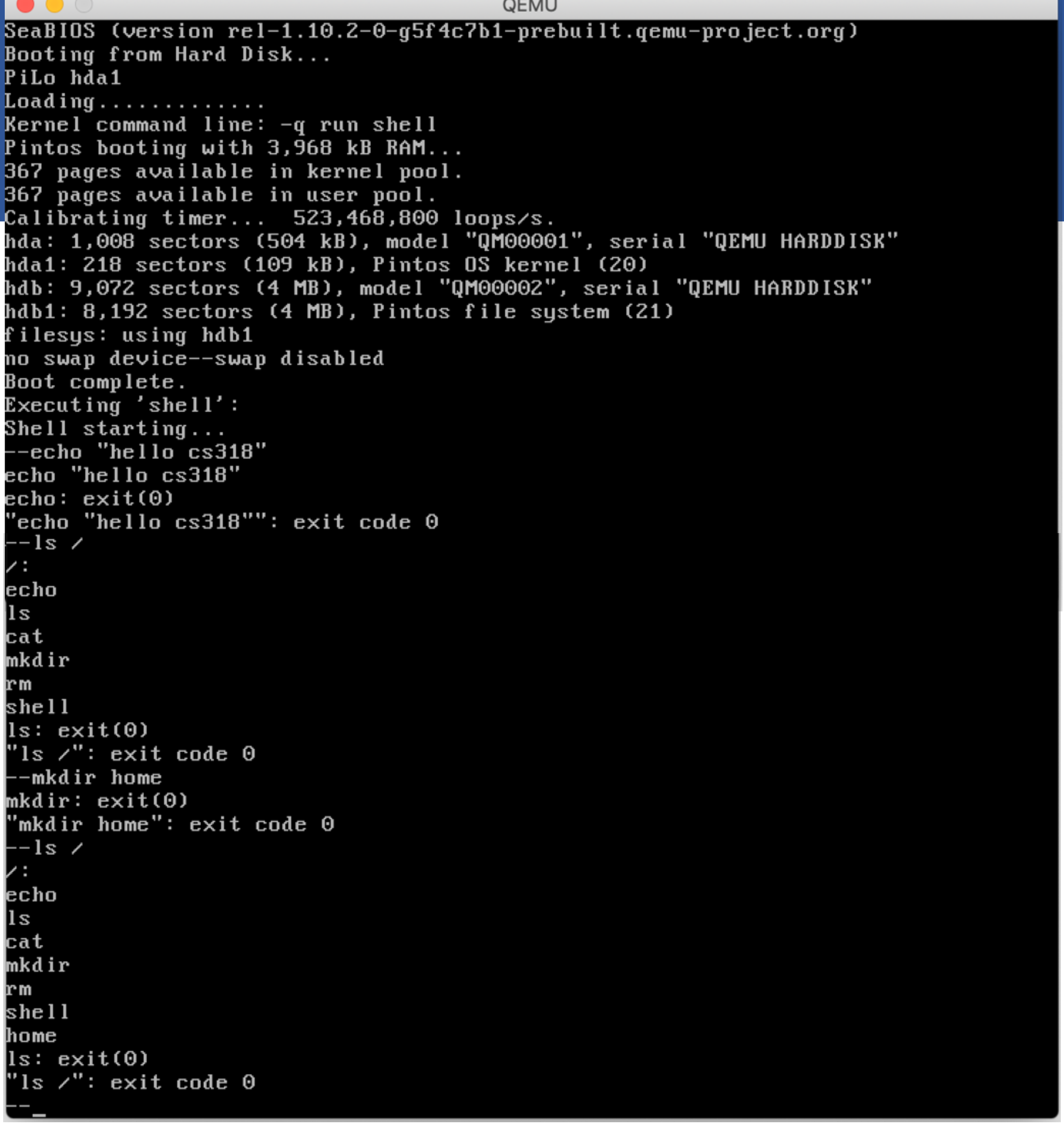
- **Implemen**
  - Develope
  - Written i
    - can ru



```
USB Device 1: Fingerprint Sensor (       )
UHCI: Enabling 2 root ports
USB: scanning devices...
UHCI: Enabling 2 root ports
USB: scanning devices...
USB Device 1: Flashdrive 303B  (Memorex )
uda: 247,616 sectors (120 MB), USB
uda1: 945 sectors (472 kB), Pintos OS kernel (20)
uda2: 9,072 sectors (4 MB), Pintos file system (21)
uda3: 1,008 sectors (504 kB), Pintos scratch (22)
filesys: using uda2
scratch: using uda3
Boot complete.
Executing 'shell':
Shell starting...The best operating system!
--echo Hello World
echo Hello World
echo: exit(0)
"echo Hello World": exit code 0
--shell
Shell starting...The best operating system!
--exit
Shell exiting.shell: exit(0)
"shell": exit code 0
--_
```

Project Assignments

- **Implement parts of Pintos operating system**
  - Developed in 2005 for Stanford's CS 140 OS class
  - Written in C, built for x86 hardware
    - can run on a real machine!
  - Use hardware emulator (QEMU/Bochs) during development



```
SeaBIOS (version rel-1.10.2-0-g5f4c7b1-prebuilt.qemu-project.org)
Booting from Hard Disk...
PiLo hda1
Loading.............
Kernel command line: -q run shell
Pintos booting with 3,968 kB RAM...
367 pages available in kernel pool.
367 pages available in user pool.
Calibrating timer...  523,468,800 loops/s.
hda: 1,008 sectors (504 kB), model "QM00001", serial "QEMU HARDDISK"
hda1: 218 sectors (109 kB), Pintos OS kernel (20)
hdb: 9,072 sectors (4 MB), model "QM00002", serial "QEMU HARDDISK"
hdb1: 8,192 sectors (4 MB), Pintos file system (21)
filesys: using hdb1
no swap device--swap disabled
Boot complete.
Executing 'shell':
Shell starting...
--echo "hello cs318"
echo "hello cs318"
echo: exit(0)
"echo "hello cs318"": exit code 0
--ls /
/:
echo
ls
cat
mkdir
rm
shell
ls: exit(0)
"ls /": exit code 0
--mkdir home
mkdir: exit(0)
"mkdir home": exit code 0
--ls /
/:
echo
ls
cat
mkdir
rm
shell
home
ls: exit(0)
"ls /": exit code 0
--_
```

Project Assignments (2)

- **One setup lab (lab 0)**
  - due next Thursday (done individually)
- **Four substantial labs:**
  - Threads, User processes, Virtual memory, File system
- **Implement projects in groups of up to 3 people**
  - Start picking your partners today
- **Warning: each project requires significant time to complete**
  - Don’t wait until the last minute to start!!

Project Assignments (3)

- **Automated tests**
  - All tests are given so you immediately know how well your solution performs
  - You either pass a test case or fail, there is *no* partial credit
- **Design document**
  - Answer important questions related to your design for a lab
- **Coding style**
  - Can your group member and TAs understand your code easily?

Project Design and Style

- **Must turn in a design document along with code**
  - Large software systems not just about producing working code
  - We supply you with templates for each project's design doc
- **TAs will manually inspect code**
  - e.g., must actually implement the design
  - must handle corner cases (e.g., handle `malloc` failure)
  - will deduct points for error-prone code
- **Code must be easy to read**
  - Indent code, keep lines and functions short
  - Use a consistent coding style
  - Comment important structure members, globals, functions

Project Lab Environment

- **The CS department ugrad and grad lab machines**
  - Running Linux on x86
  - The toolchain already setup
- **You may also use your own machine**
  - We have written detailed instructions for setting up the environment
    - https://cs.jhu.edu/~huang/cs318/fall20/project/setup.html
  - Unix and Mac OS preferred. Windows needs VMs
  - Pre-built VM image can be downloaded here

Exams

- **Midterm**
  - Covers first half of class + questions related to projects
  - Tuesday, October 20th
- **Final**
  - Covers second half of class + selected materials from first part
    - I will be explicit about the material covered
  - Also include project questions
  - TBA

Exam Format

- **Online with the LockDown Browser**
  - This course uses LockDown Browser and a webcam for online exams
  - Make sure you have a webcam that's built into your computer or one that plugs in with a USB cable.
- **Download link and instructions will be provided later on Piazza**

Grading

- **Midterm: 15%**
- **Final Exam: 25%**
- **Project: 60%**
  - Lab 4 is optional for 318-section student
    - Will receive a max 6% bonus points if choosing to do it
  - For each project
    - 60% based on passing test cases
    - 40% based on design document and style

Late Policies

- **Late submissions receive penalties as follows**
  - 1 day late, 15% deduction
  - 2 days late, 30% deduction
  - 3 days late, 60% deduction
  - after 4 days, no credit
- **Each team will have a total of 6-day grace period**
  - can spread into 4 projects
  - for interview, attending conference, errands, etc., no questions asked
  - use it wisely, strongly suggest to reserve it for later labs (lab3, 4)

Collaboration and Cheating Policies

- **Collaboration**
  - Explaining a concept to someone in another group
  - Discussing algorithms/testing strategies with other groups
  - Helping debug someone else’s code (in another group)
- **Do not look at other people’s solutions**
  - Including solutions online
    - This means copying code from GitHub will get you into big trouble
  - We will run comprehensive tools to check for potential cheating.
- **Do not publish your own solutions**
  - online (e.g., on GitHub) or share with other teams
- **Cite any code that inspired your code**
  - If you cite what you used, it won’t be treated as cheating
    - in worst case, we deduct points if it undermines the assignment

Do Not Cheat

- It *will* be caught
- The consequence is very high
- Truth: you always get better outcome by not cheating

How *Not* to Pass CS 318?

- **Do not come to lecture**
  - The slides are online and the material is in the book anyway
  - Lecture walks you through difficult materials and tells you the context
- **Do not do the homework**
  - It's not part of the grade
  - Concepts seem straightforward...until you apply them
  - Excellent practice for the exams, and project

How *Not* to Pass CS 318?

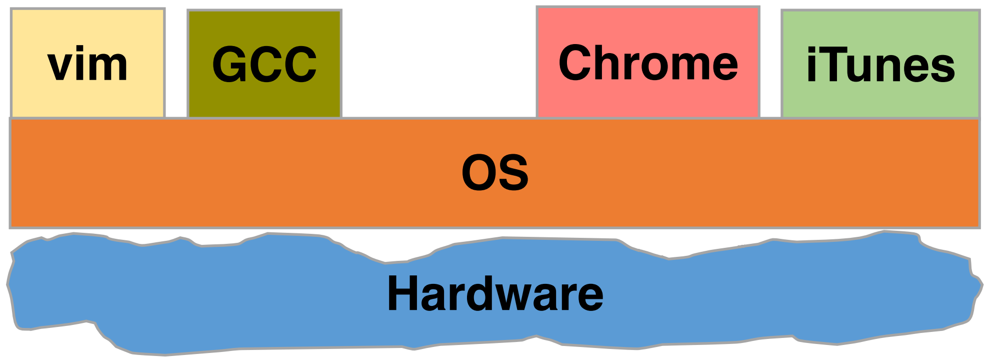
- **Do not ask questions in lecture, office hours or online**
  - It's scary, I don't want to embarrass myself
  - Asking questions is the best way to clarify lecture material
  - Office hours and email will help with homework, projects
- **Wait until the last couple of days to start a project**
  - We'll have to do the crunch anyways, why do it early?
  - The projects cannot be done in the last few days
  - Repeat: **The projects cannot be done in the last few days**
  - (p.s. The projects cannot be done in the last few days)

Questions

- **Before we start, any questions?**

What Is An Operating System?

- **Layer between applications and hardware**



- **All the code that you didn't have to write to implement your app**

OS and Hardware

- **Manage hardware resources**
  - Computation (CPUs)
  - Volatile storage (memory) and persistent storage (disk, etc.)
  - Communication (network, modem, etc.)
  - Input/output devices (keyboard, display, printer, camera, etc.)
- **Provides abstractions to hide details of hardware from applications**
  - Processes, threads
  - Virtual memory
  - File systems
  - …

OS and Hardware (2)

- **Mediate accesses from different applications**
  - Who has access at what point for how much/long
- **Benefits to applications**
  - Simpler (no tweaking device registers)
  - Device independent (all network cards look the same)
  - Portable (across Win95/98/ME/NT/2000/XP/Vista/7/8/10)

OS and Applications

- **Virtual machine interface**
  - The OS defines a logical, well-defined environment
  - Each program thinks it owns the computer
- **Provides protection**
  - Prevents one process/user from clobbering another
- **Provides sharing**
  - Concurrent execution of multiple programs (time slicing)
  - Communication among multiple programs (pipes, cut & paste)
  - Shared implementations of common facilities, e.g., file system

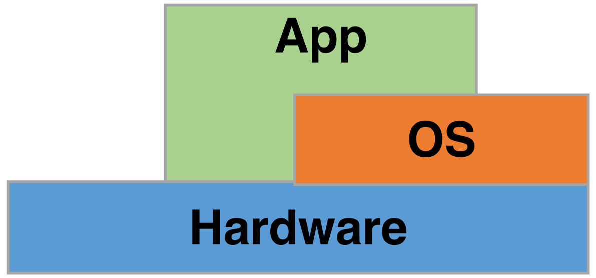
Questions to Ponder

- **What is part of an OS? What is not?**
  - Is the windowing system part of an OS?
  - Is the Web browser part of an OS?
  - This very question leads to different OS designs
- **How different are popular OSes today?**

Walk-through of OS basics

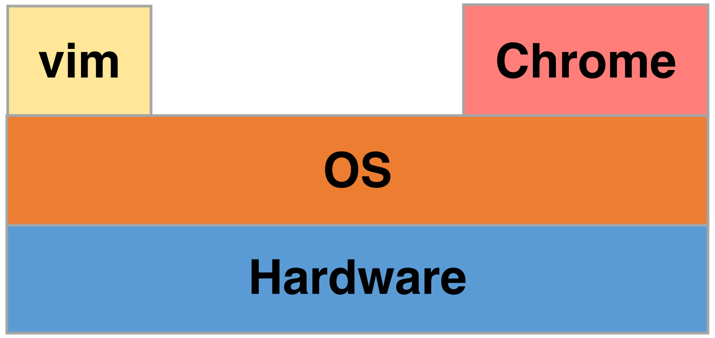
A Primitive Operating System

- **Just a library of standard services**



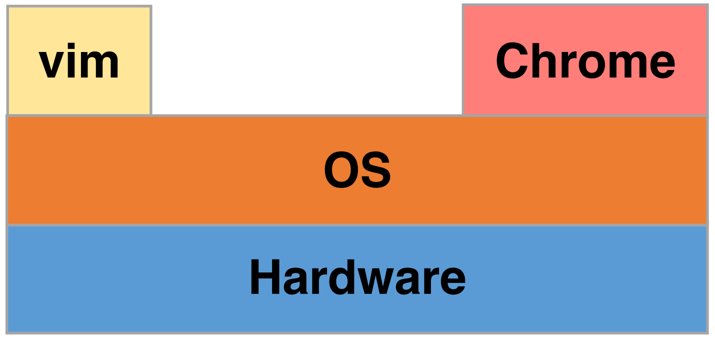
- **Simplifying assumptions**
  - System runs one program at a time
  - No bad users or programs
- **Problems: poor utilization**
  - ...of hardware (e.g., CPU idle while waiting for disk)
  - ...of human user (must wait for each program to finish)

Multitasking



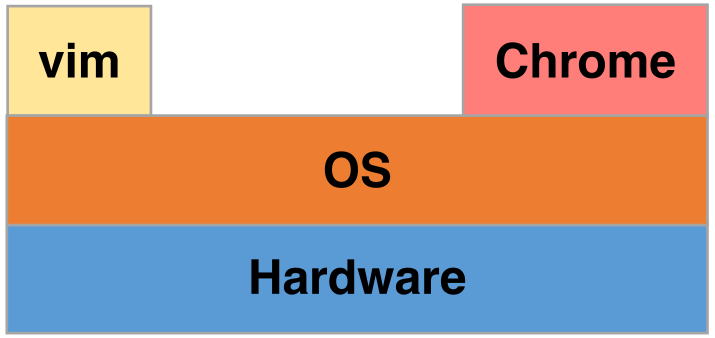
- **Idea: more than one process can be running at once**
  - When one process blocks (waiting for disk, network, user input, etc.) run another process
- **Mechanism: context-switch**
  - When one process resumes, it can continue from last execution point

Multitasking



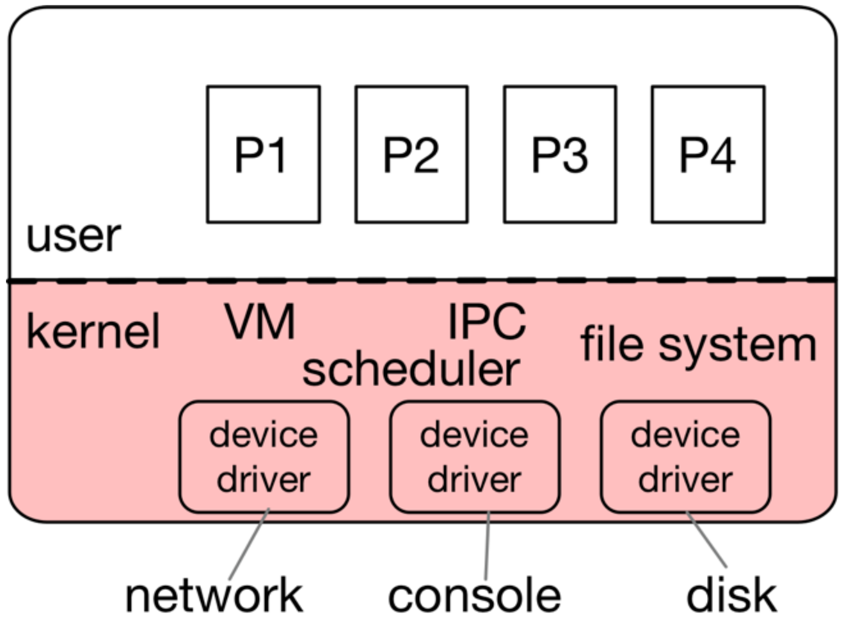
- **Idea: more than one process can be running at once**
- **Mechanism: context-switch**
- **Problems: ill-behaved process**
  - go into infinite loop and never relinquish CPU
  - scribble over other processes' memory to make them fail

Multitasking



- **Problems: ill-behaved process**
  - go into infinite loop and never relinquish CPU
  - scribble over other processes' memory to make them fail
- **Solutions:**
  - **scheduling**: fair sharing, take CPU away from looping process
  - **virtual memory**: protect process's memory from one another

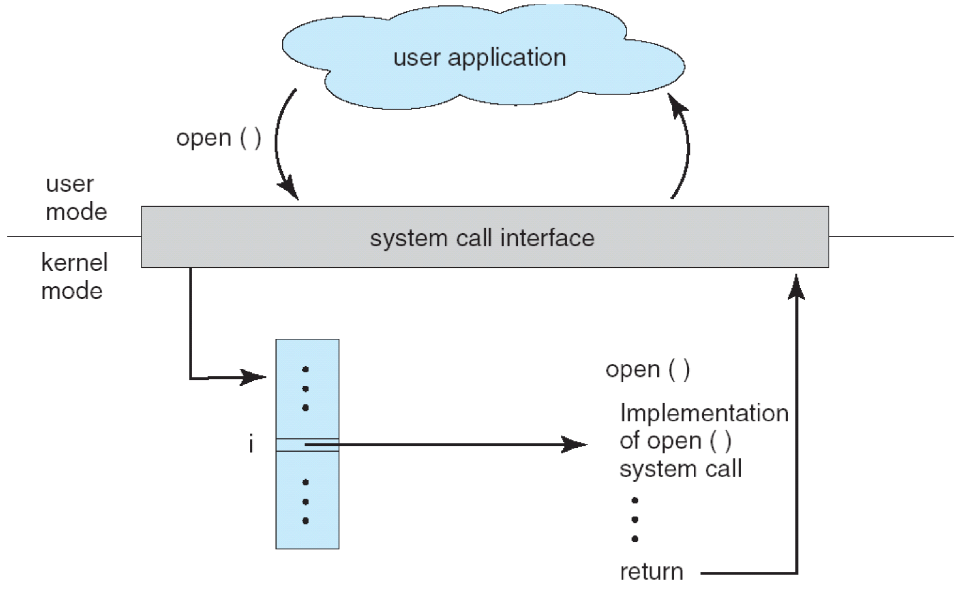
Typical OS Structure



- **Most software runs as user-level processes (P[1-4])**
- **OS kernel runs in privileged mode (shaded)**

System Calls

```c
#include <fcntl.h>
#include <unistd.h>
int main()
{
  int fd = open("cs318.txt", O_WRONLY | O_CREAT | O_TRUNC, 0644);
  if (fd < 0) {
    write(2, "Failed to open cs318.txt\n", 25);
    _exit(1);
  }
  write(fd, "Hello, OS!\n", 11);
  close(fd);
  return 0;
}
```

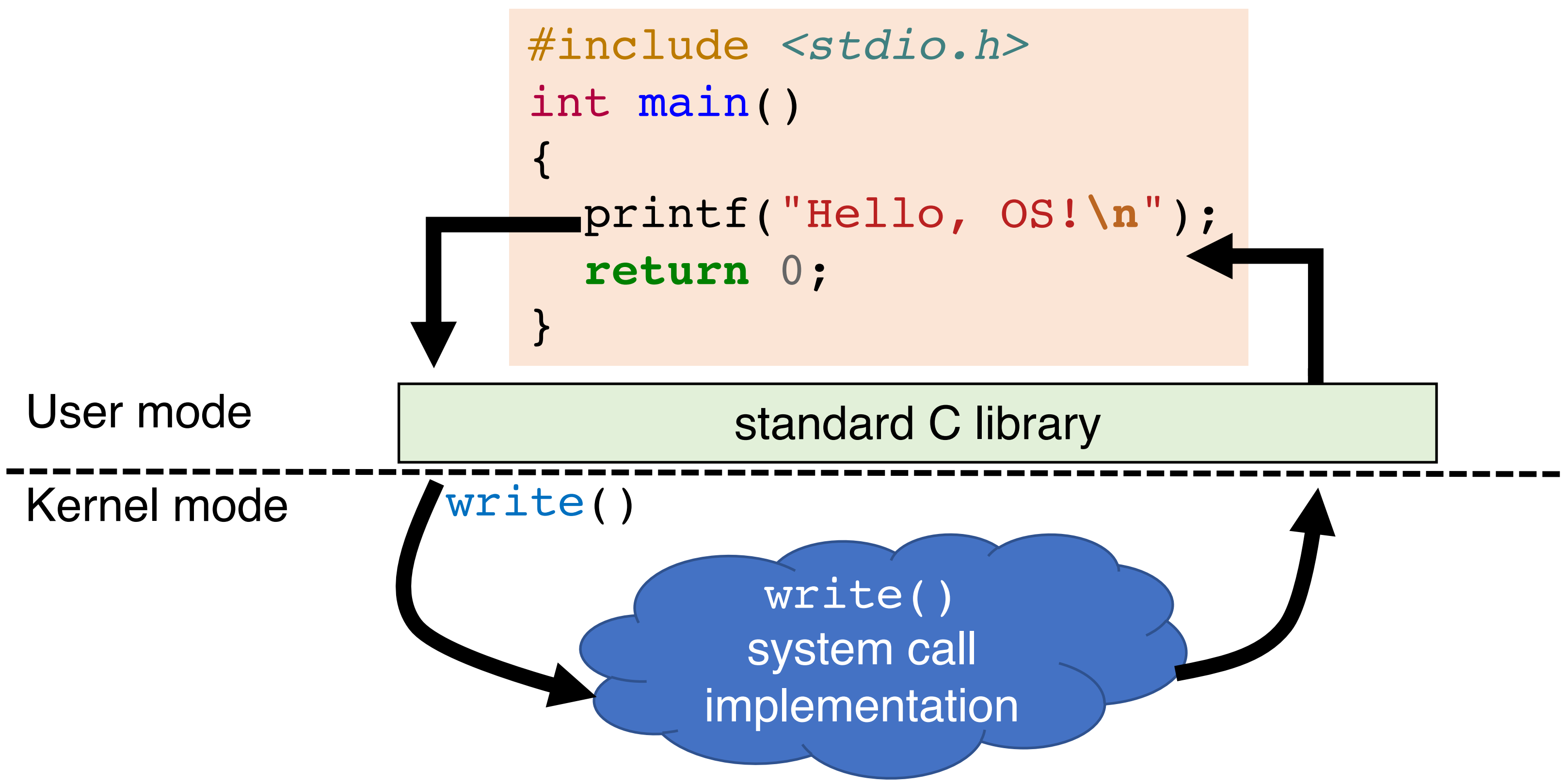


- **Applications can invoke kernel through system calls**
  - Special instruction transfers control to kernel
  - ...which dispatches to one of few hundred syscall handlers

System Calls (continued)

- **The *only* way for an application to invoke OS services**
- **Goal: Do things application can't do in unprivileged mode**
  - Like a library call, but into more privileged kernel code
- **Kernel supplies well-defined system call interface**
  - Applications set up syscall arguments and trap to kernel
  - Kernel performs operation and returns result
- **Higher-level functions built on syscall interface**
  - `printf`, `scanf`, `fgets`, *etc.* all user-level code

System Calls (continued)



```
#include <stdio.h>
int main()
{
  printf("Hello, OS!\n");
  return 0;
}
```

- **Standard library implemented in terms of syscalls**

For Next Class...

- Browse the course web
  - https://cs.jhu.edu/~huang/cs318/fall20
- Sign up on Piazza
- Read Chapters 1 and 2
- Setup Pintos and read its documentation
  - Work on Lab 0
- Looking for project partners

For Next Class...

- Browse the course web
  - https://cs.jhu.
- Sign up on Pi
- Read Chapte
- Setup Pintos
  - Work on Lab
- Looking for p