



Operating Systems (Honor Track)

Lecture 1: Course Introduction

Yao Guo (郭耀)

Peking University

Fall 2024

Class Information

□ References:

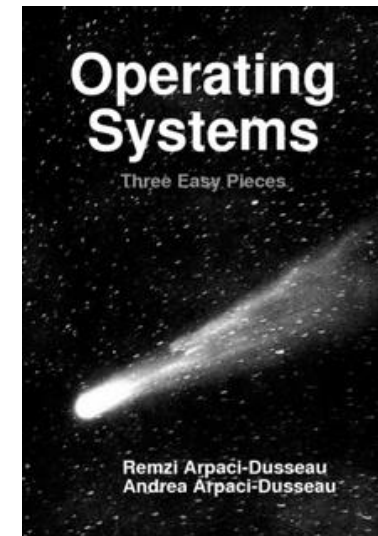
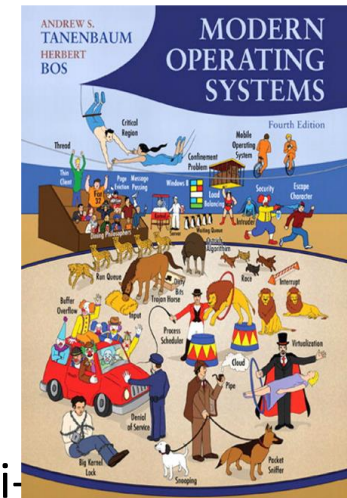
- Modern Operating Systems, 4th Edition (Tanenbaum & Hal)
- Operating System Concept (Abrahan Silberschatz)
- **Operating Systems: Three Easy Pieces** (Remzi Arpaci-Dusseau & Andrea Arpaci-Dusseau) **It's Free!**

□ Class Website

- <http://course.pku.edu.cn/>

□ TA

- 黄瑞哲 ruizhe.huang@stu.pku.edu.cn
- TBD





Class Requirements % Grading

- Part I: the basic topics (50%)
 - OS principles and techniques (lectures)
 - Midterm exam (20%): TBD (~Week 9)
 - Final exam (30%): Jan 6, 2025 (2~4pm)
- Part II: the advanced topics (20%)
 - Class participation (& presentation)
 - Paper reading (& presentation)
 - Come to class and join discussions!
- Part III: OS Lab (30%)
 - 5 projects (+ challenges for bonus)

OS Lab: JOS

- JOS is an x86-based OS designed by MIT for teaching
 - See website: <https://pdos.csail.mit.edu/6.828/2018/>
 - Note that we will follow the lab requirements of 2018
- To complete JOS, several labs are provided, in which we'll implement important OS components from scratch
 - Lab 1: bootloader
 - Lab 2: virtual memory management
 - Lab 3: user mode and system call
 - Lab 4: multitasking
 - Lab 5: file system
- Alternative option:
 - Implement new OS functionalities on a RISC-V/ARM IoT devices
 - Rust-based OS kernel construction (e.g., add modules for Asterinas)





OS Lab: Grading (30%)

- Complete the lab requirements: 60%
 - Complete the coding tasks
 - Submit the report
- Quiz : 40%
 - Two quizzes: 20% each
 - (May be combined to one quiz)
- Bonuses: 10%
 - One challenge for each lab
 - Choose from a given list of challenges
 - (Maximum Lab points: 30%)

A starting survey

1. In your opinion, what is the definition of an operating system (OS)?
2. Write down 10 concepts/words coming out of your mind when you think of an OS.
3. What do you want to learn in an OS course?



What is an OS?

- What does it do?

- Give me a few names of OSes?
 - For desktops?
 - For smartphones?
 - For embedded systems?
 - For robots?

- We use one or more OSes everyday!



Search Applications

Filter results ▸

Recently Used



Installed See fewer results ▾



Bluetooth

Reveal when moving the pointer to the defined hot spot.

Top left corner

Reveal sensitivity Low

Restore Default Behaviours



Recycle Bin



Jim Tanous



Documents



Pictures



PC settings



File Explorer



Snipping Tool



Calculator



Sticky Notes



Paint

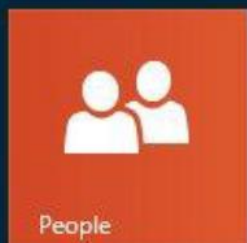
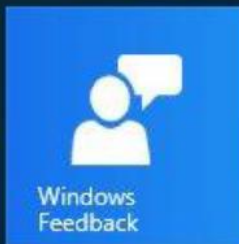
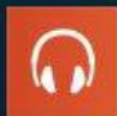
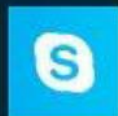


WinSnap



All Apps

Search everywhere



People



Monday

Calendar



Supreme Court declines to review same-sex marriage cases

News



FIFA 15: UT

Store



Mail





Today

Thurs
Oct

Weather

9:20 AM
Tomkins Co


9:20 AM
New York

9:20 AM
Albany


9:20 AM
Poland

Show More

World Clock


New York

Social

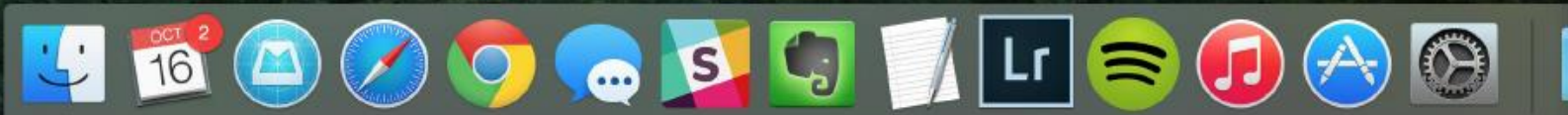


Calculator

C

7

4

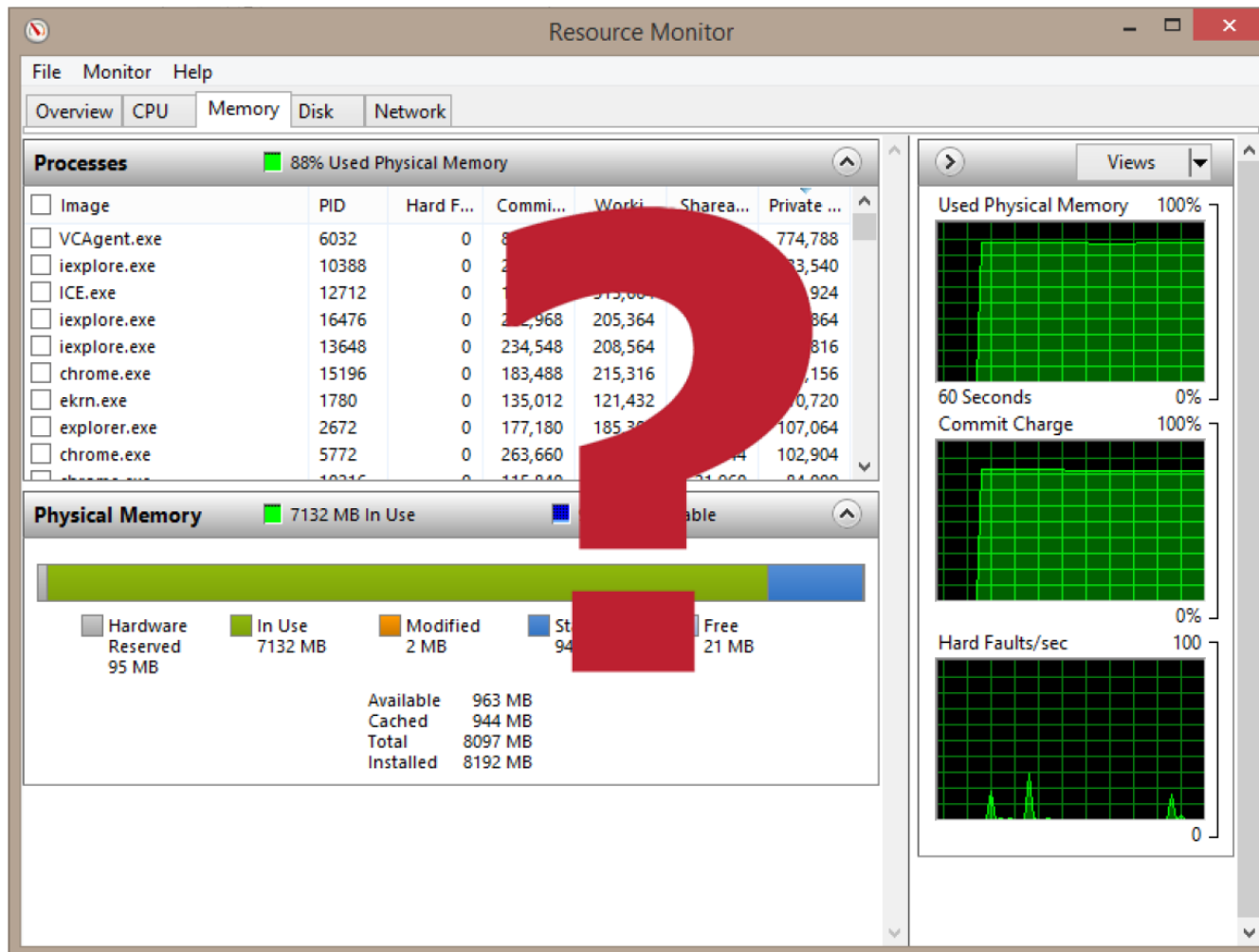


```

top - 20:48:08 up 275 days, 0:00:00, 0% load average: 0.06, 0.07, 0.05
Tasks: 171 total, 1 running, 170 sleeping, 0 zombie
Cpu(s): 0.1%us, 0.1%sy, 0.0%id, 99.8%wa, 0.0%hi, 0.0%si, 0.0%st
Mem: 16467276k total, 141596k used, 23016312k free, 171168k buffers
Swap: 0k total, 0k used, 0k free, 884340k cached

  PID USER      PR  NI  VIRT  RES  SHR  S  %CPU  TIME+  COMMAND
14677 voelker   20   0 55548 3232 2364 R    0.0  0:00.07 top
24637 voelker   20   0 86300 6364 1024 S    0.0  32:06.70 mosh-server
   1 root      20   0 57812 1636 584 S    0.0  1:26.73 init
   2 root      20   0      0     0  0 S    0.0  0:03.13 kthreadd
   3 root      RT   0      0     0  0 S    0.0  0:04.38 migration/0
   4 root      20   0      0     0  0 S    0.0  9:54.94 ksoftirqd/0
   5 root      RT   0      0     0  0 S    0.0  0:00.01 watchdog/0
   6 root      RT   0      0     0  0 S    0.0  0:04.39 migration/1
   7 root      20   0      0     0  0 S    0.0 11:22.89 ksoftirqd/1
   8 root      RT   0      0     0  0 S    0.0  0:00.01 watchdog/1
   9 root      RT   0      0     0  0 S    0.0  0:18.05 migration/2
  10 root      20   0      0     0  0 S    0.0  9:44.37 ksoftirqd/2
  11 root      RT   0      0     0  0 S    0.0  0:00.01 watchdog/2
  12 root      RT   0      0     0  0 S    0.0  0:18.06 migration/3
  13 root      20   0      0     0  0 S    0.0  9:01.67 ksoftirqd/3
  14 root      RT   0      0     0  0 S    0.0  0:00.01 watchdog/3
  15 root      20   0      0     0  0 S    0.0  2:30.99 events/0

```



Why do we need to learn OS (1/2)

- Do you know how to write a program to control a hardware device?
- Is it easy to show “Hello world!” in a display?

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    puts("Hello world!");
    return 0;
}
```



What will happen (1)?

- The user tells the OS to run the program (How?)
- The OS locates the program file, checks its type
- The OS checks the header of the program file, finds the locations of the code and data (file format?)
- The file system locates the first disk block of the file
- The OS creates a new process (to run the program)
- The OS maps the program file to the process
- The OS sets the CPU context, and sets the CPU program counter to the beginning of the program
- The first instruction of the program is to be run, and failed – Page Fault
- The OS allocates one page, reads some code, and continue to run the instructions
- More page faults occur, and more pages are allocated and filled



What will happen (2)?

- The program/process invokes a system call to write a string into a descriptor
- The OS checks if the address of the string is valid
- The OS locates the device by the descriptor
- The OS finds that this device is a pseudo-terminal controlled by a process
- The OS sends the string to the process
- The process tells the window system that it would like to display a string
- The window system verifies that this is a valid operation, and converts the string to pixels
- The window system writes the pixels into some video memory and communicates with the video adapter
- The video adapter converts the pixel information into some control/data signals to the monitor
- The monitor interprets the signals and fires the LEDs
- Finally, we see “Hello world!” on the screen



Buzz Words (e.g.)

Multiprogramming	PCB	Starvation	Page replacement
Time-sharing	Context switch	Preemptive	Temporal locality
Booting	Thread	Priority	Programmed I/O
Protection	Multithreading	Deadlock	DMA
Privileged instruction	Scheduling	Wait-for graph	Buffer
Exception	Synchronization	Virtual memory	RAID
Fault	Data race	Paging	Disk cache
System call	Mutual exclusion	Virtual address	File
Interrupt	Critical section	Swapping	Directory
Process	Atomic operation	Page table	Device driver
Execution state	Lock	TLB	Page fault
Address space	Semaphore	Shared address space	Spatial locality



Why do we need to learn OS (2/2)

- Do you know how to make your machine run two programs?
 - Why is a machine able to run two programs?
 - How/where to put them into memory? What if the memory is not large enough to hold them (or even one of them)?
 - How to avoid one program to interfere with another?
 - What if the two programs want to communicate?
 - What if two programs want to write different values to the same location?
 - ...
- More programs? More issues?



What will we learn in this course

- Architecture support for OS
- Processes and threads
- Memory management
- File systems
- I/O
-
- And some **interesting** & **cutting-edge** techniques...

And more...

Principles

- ☐ System concepts
- ☐ OS design
- ☐ Some theory
- ☐ Rationale

Goals

- ☐ Understanding OS decisions
- ☐ Basis for future learning
- ☐ Design something?

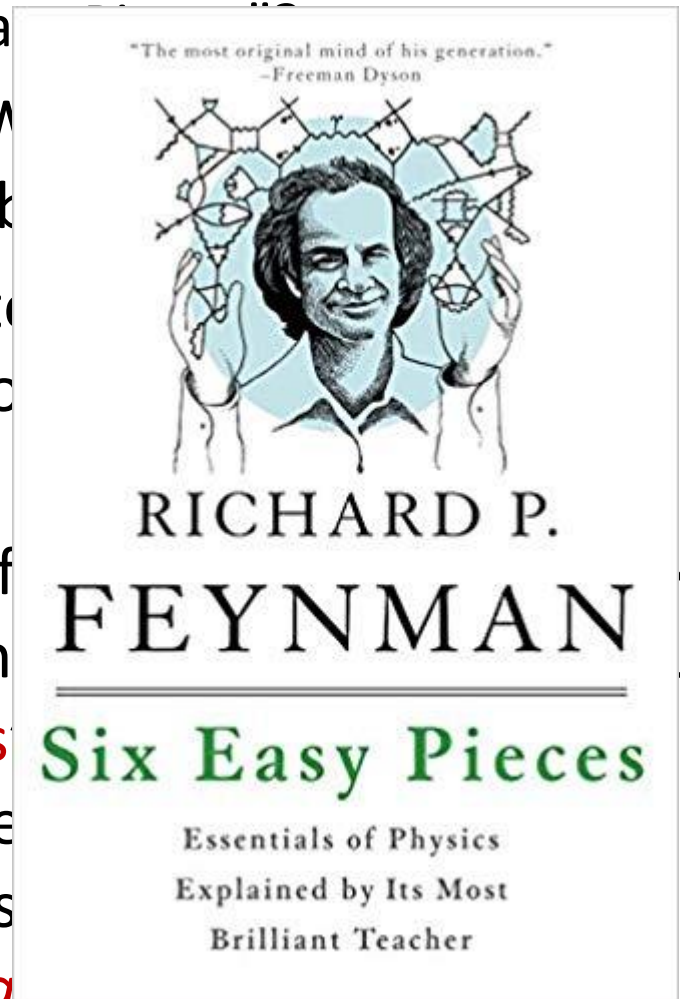
Operating Systems: Three Easy Pieces



Virtualization		Concurrency	Persistence
3 <u>Dialogue</u>	12 <u>Dialogue</u>	25 <u>Dialogue</u>	35 <u>Dialogue</u>
4 <u>Processes</u>	13 <u>Address Spaces</u> <u>code</u>	26 <u>Concurrency and Threads</u> <u>code</u>	36 <u>I/O Devices</u>
5 <u>Process API</u> <u>code</u>	14 <u>Memory API</u>	27 <u>Thread API</u>	37 <u>Hard Disk Drives</u>
6 <u>Direct Execution</u>	15 <u>Address Translation</u>	28 <u>Locks</u>	38 <u>Redundant Disk Arrays (RAID)</u>
7 <u>CPU Scheduling</u>	16 <u>Segmentation</u>	29 <u>Locked Data Structures</u>	39 <u>Files and Directories</u>
8 <u>Multi-level Feedback</u>	17 <u>Free Space Management</u>	30 <u>Condition Variables</u>	40 <u>File System Implementation</u>
9 <u>Lottery Scheduling</u> <u>code</u>	18 <u>Introduction to Paging</u>	31 <u>Semaphores</u>	41 <u>Fast File System (FFS)</u>
10 <u>Multi-CPU Scheduling</u>	19 <u>Translation Lookaside Buffers</u>	32 <u>Concurrency Bugs</u>	42 <u>FSCK and Journaling</u>
11 <u>Summary</u>	20 <u>Advanced Page Tables</u>	33 <u>Event-based Concurrency</u>	43 <u>Log-structured File System (LFS)</u>
	21 <u>Swapping: Mechanisms</u>	34 <u>Summary</u>	44 <u>Flash-based SSDs</u>
	22 <u>Swapping: Policies</u>		45 <u>Data Integrity and Protection</u>
	23 <u>Complete VM Systems</u>		46 <u>Summary</u>
	24 <u>Summary</u>		47 <u>Dialogue</u>
			48 <u>Distributed Systems</u>
			49 <u>Network File System (NFS)</u>
			50 <u>Andrew File System (AFS)</u>
			51 <u>Summary</u>

The opening dialog from OSTEP (1)

- **Student:** Why is it called “Three Easy Pieces”?
- **Professor:** That’s an easy one. We’re going to do these great lectures on Physics by Richard Feynman.
- **Student:** Oh! The guy who wrote *Mr. Feynman*, right? Great book, but not as hilarious like that book was?
- **Professor:** Um... well, no. Hopefully you’ll find his notes on Physics. Some of them are in a book called “**Six Easy Pieces**” on Quantum Physics; we’re going to do Three Easy Pieces on the topic of Operating Systems. This book is called *Operating Systems are about having to deal with physics.*





The opening dialog from OSTEP (2)

- **Student:** Well, I liked physics, so that is probably good. What are those pieces?
- **Professor:** They are the three key ideas we're going to learn about: **virtualization, concurrency, and persistence**. In learning about these ideas, we'll learn all about how an operating system works, including how it decides what program to run next on a CPU, how it handles memory overload in a virtual memory system, how virtual machine monitors work, how to manage information on disks, and even a little about how to build a distributed system that works when parts have failed. That sort of stuff.
- **Student:** I have no idea what you're talking about, really.
- **Professor:** Good! That means you are in the right class.



The opening dialog from OSTEP (3)

- **Student:** **what's the best way to learn this stuff?**
- **Professor:** Excellent question. **this out on their own**, or
- **Student:** Oh, I know! **I remember. I do and I understand.**
- **Professor:** (surprised) How can you say that?
- **Student:** It seemed to follow from Confucius, and an even better source for this question.
- **Student:** So we have to think? Well, I'm up for that. I mean, what else do I have to do anyhow? **It's not like I have much of a life outside of this book.**

不闻不若闻之，
闻之不若见之，
见之不若知之，
知之不若行之；
学至于行之而止矣。

——《荀子·儒效》

Andrew S. Tanenbaum: The Impact of MINIX





Any questions?

- ☐ Let me know what you expect to learn in this course.
- ☐ Next lecture: Introduction to OS