CS4500
Operating Systems

Week 1 – Part B

Course Content

Unit	Module	Week	Topic
1 – Introduction	1 & 2	1	Welcome to OS & Introduction
	2	2	OS Overview
2 – Processes and Threads	3	3	Processes
	4	4	Threads
	5	5	CPU Scheduling
	6	6	Multiprocessor Scheduling
	7	7	Interprocess Communication
	8	8	Pthread
3 – Deadlocks & Memory	9	9	Deadlock
	10	10	Memory Management
	--	11	Midterm Exam
4 – Additional Topics	11	11	Page Replacement
	12	13	File Systems
	13	14	IO Devices
	14	15	Security
	--	16	Final Exam

Why Study Operating Systems?

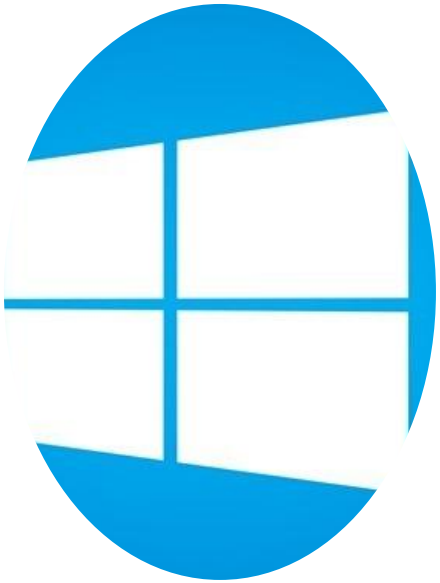
Why Study Operating Systems?

The most complex software

- > 27 million lines of code in Linux

The most fundamental software

- OSes are almost everywhere, e.g., supercomputer, PC, phone…



Why Study Operating Systems?

By studying OS, you will

- Learn how computers work
  - Gain a good understanding of OS and hardware
- Learn about system design
  - Simplicity, portability, performance, and trade-offs

Hardware abstraction

- Processes, threads, files …

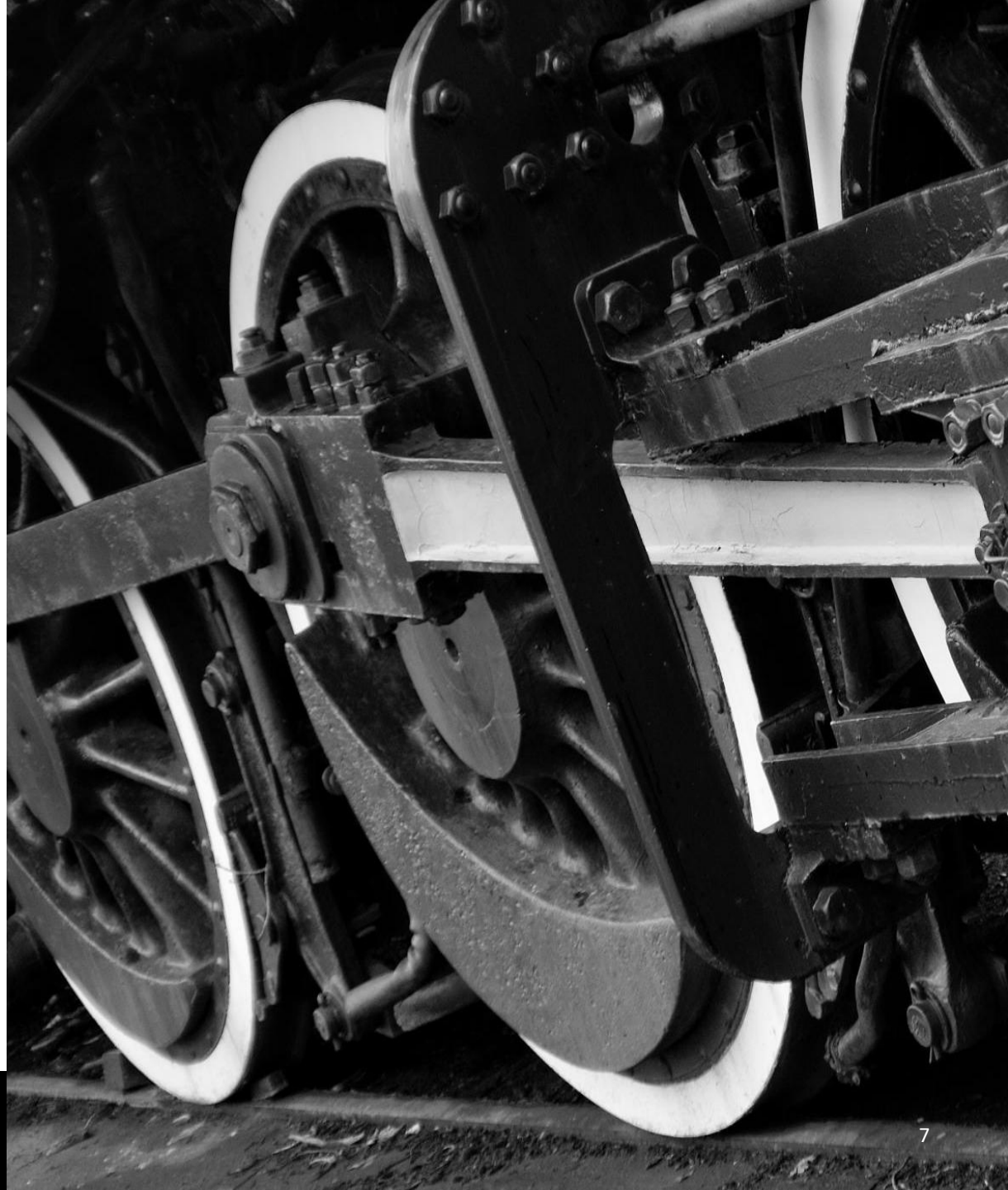
Resource management

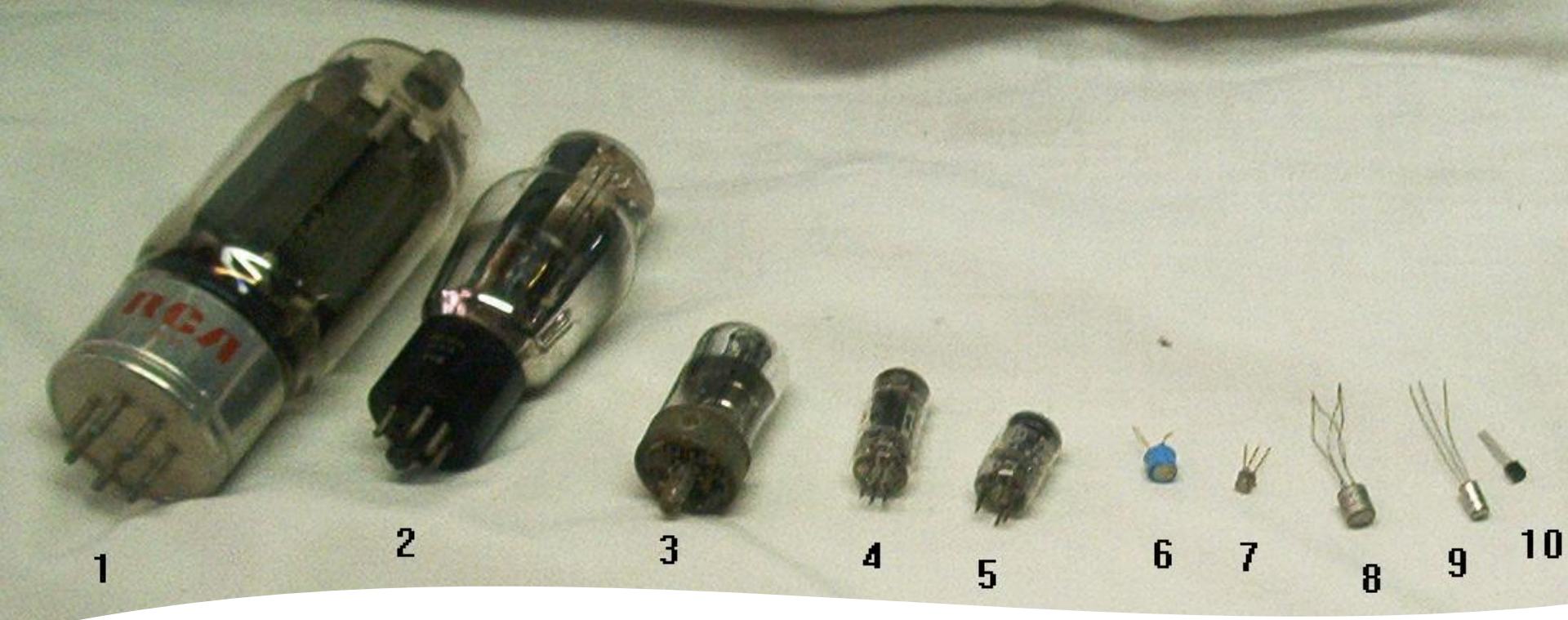
- CPU scheduling, memory management, file systems …

Coordination

- Multiple programs and users
- Fairness and efficiency

History





History of Operating Systems

First generation 1945 - 1955
- Vacuum tubes, plug boards

1945–1955

Second generation 1955 - 1965
- Transistors, batch systems

1955–1965

Third generation
- ICs and multiprogramming

1965–1980

Fourth generation 1980 – present
- Personal computers: LSI (large scale integration)

1980 – present

next 5-10 years
- Mobile devices
- Many-core computers

Present

Batch Processing (1955-1965)

Reduce wasted time --> adopted batch system

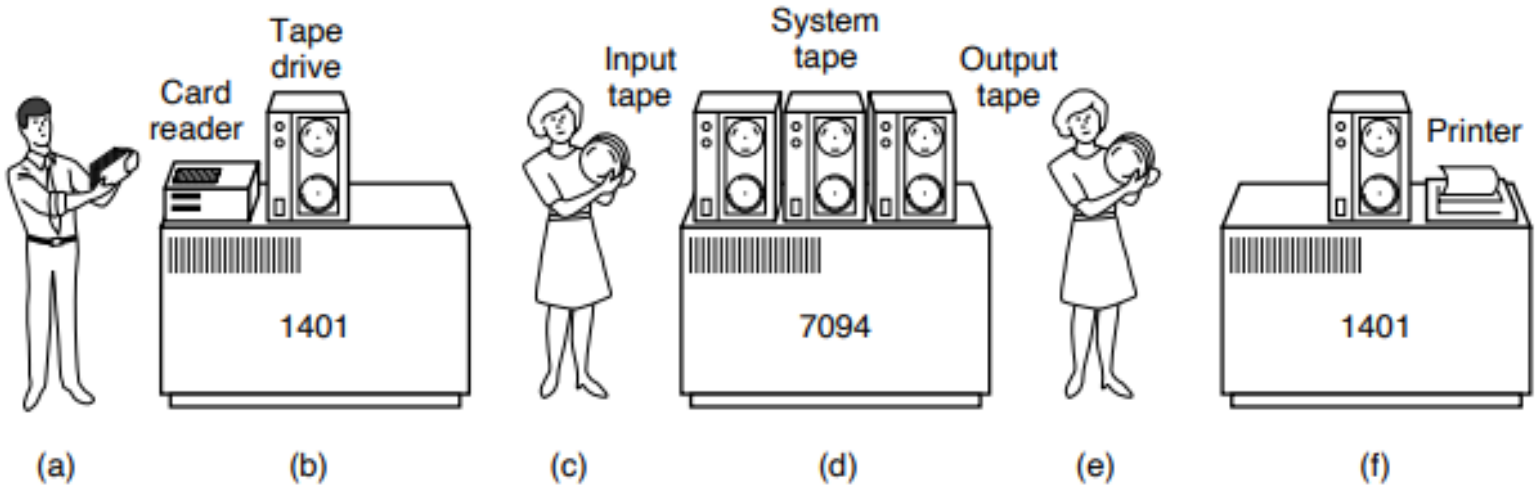
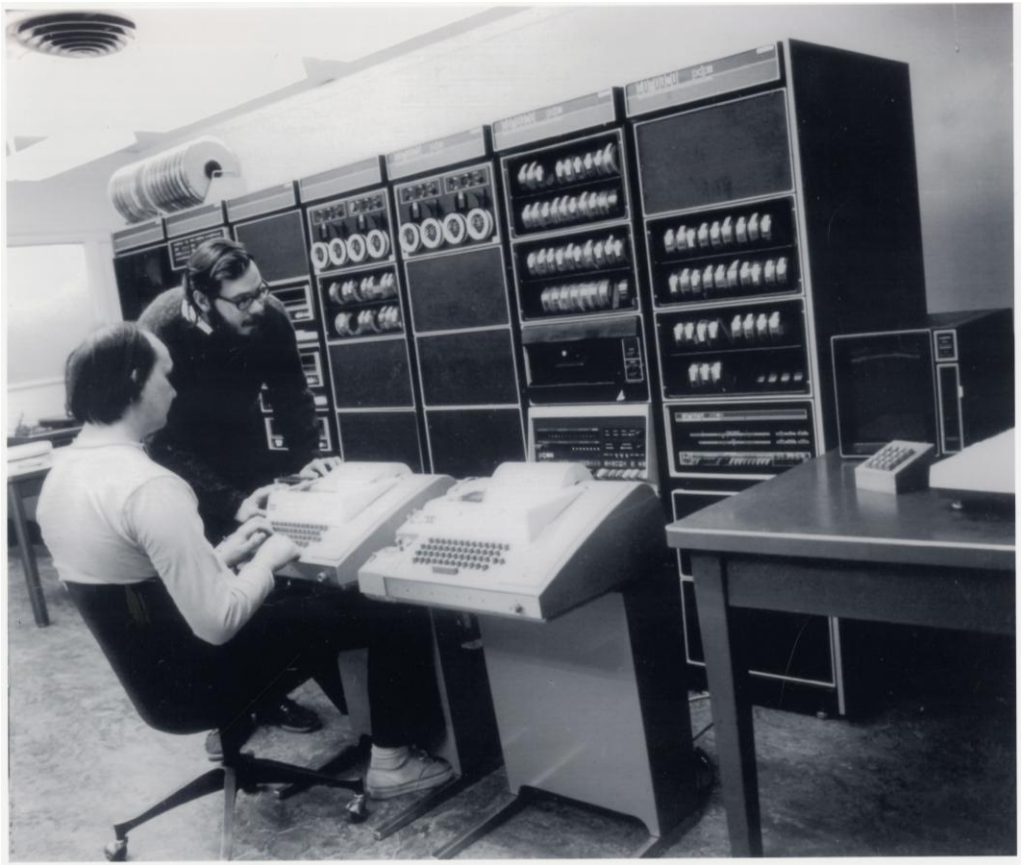


Figure 1-3. An early batch system. (a) Programmers bring cards to 1401. (b) 1401 reads batch of jobs onto tape. (c) Operator carries input tape to 7094. (d) 7094 does computing. (e) Operator carries output tape to 1401. (f) 1401 prints output.

UNIX – A Simpler Operating System



Courtesy of the Computer History Museum

```
/*
 * You are not expected to understand this.
 */
```

Comment by Ken Thompson in the source code of UNIXv6, 1975. aka the most famous comment ever. Interestingly, the code to which it pertained contained a bug, so perhaps the authors did not understand it either. :)

Linux (1990s). Source: *comp.os.minix*

“Hello everybody out there using minix -

I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) for 386(486) AT clones. This has been brewing since april, and is starting to get ready. I'd like any feedback on things people like/dislike in minix, as my OS resembles it somewhat (same physical layout of the file-system (due to practical reasons) among other things).

I've currently ported bash(1.08) and gcc(1.40), and things seem to work. This implies that I'll get something practical within a few months, and I'd like to know what features most people would want. Any suggestions are welcome, but I won't promise I'll implement them :-)

Linus (torvalds@kruuna.helsinki.fi

PS. Yes - it's free of any minix code, and it has a multi-threaded fs. It is NOT portable (uses 386 task switching etc), and it probably never will support anything other than AT-harddisks, as that's all I have :-(”.

Supplemental Information

Fun Timeline Website

https://www.g2.com/articles/history-of-computers

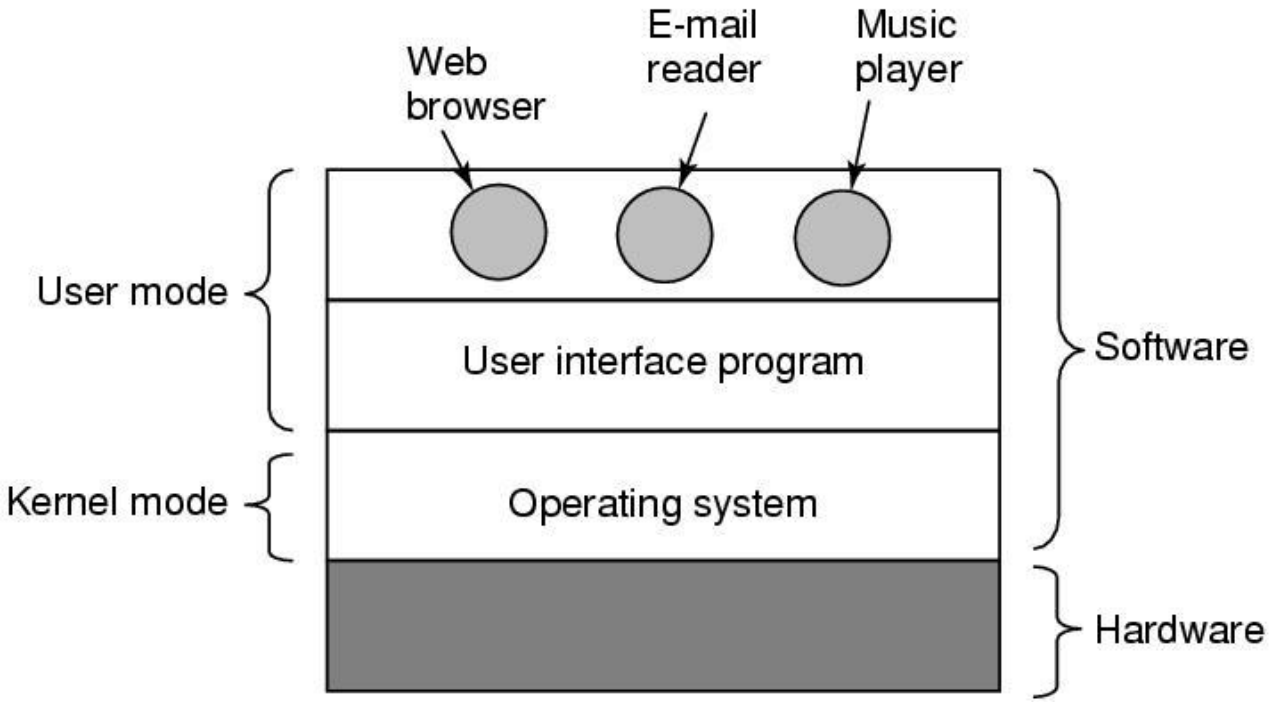
Triumph of the Nerds (Documentary)

https://www.youtube.com/watch?v=MNVbmzVCyLU

OS Overview

What is an Operating System?

- A computer system consists of
  - hardware
  - system programs
  - application programs

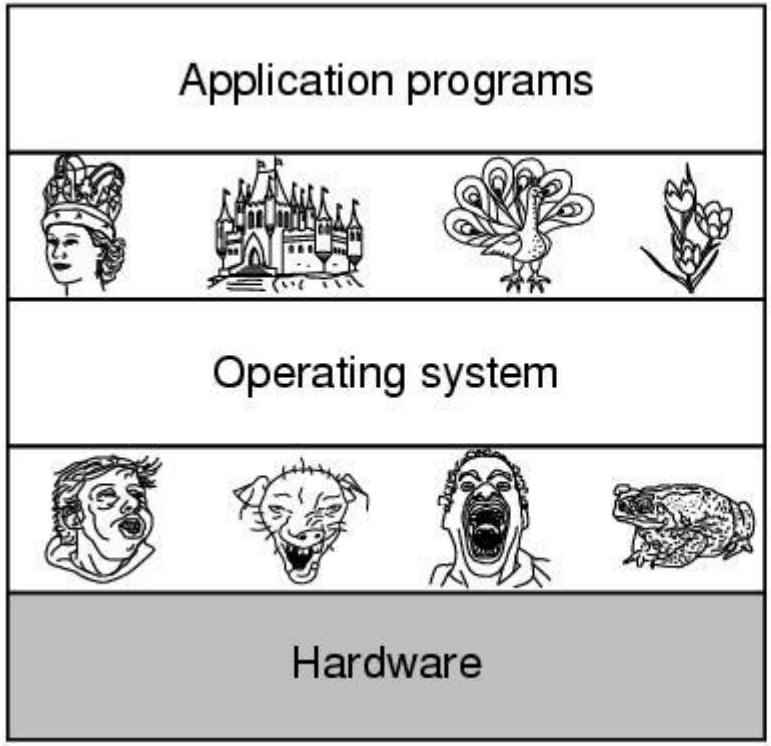


What does an Operating System do?

It is an extended (or virtual) machine

- Hides the messy details which must be performed
- Presents user with a virtual machine, easier to use
- Provides protection

The Operating System as an Extended Machine




```
fprintf(fd, "%d", data);
        ↓
write(fd, buffer, count);
```

← Beautiful interface

```
file->f_op->write(file, buf,
count, pos);
        ↓
        ⋮
```

← Ugly interface

```
load(block, length, device);
seek(device, track);
out(device, sector);
```

What does an Operating System do?

It is an extended (or virtual) machine	Hides the messy details which must be performed Presents user with a virtual machine, easier to use Provides protection
It is a resource manager	Each program gets time with the resource, e.g., CPU Each program gets space on the resource, e.g., MEM

Multiplexing

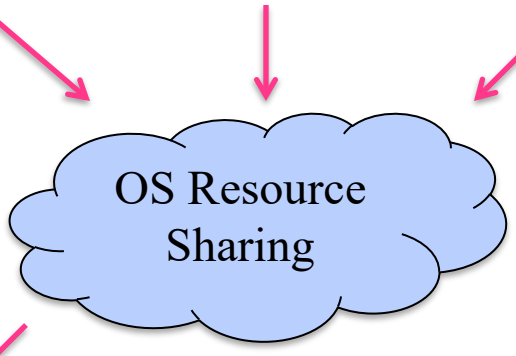
Resource management includes sharing resources in two different ways

- In time - different programs or users take turns using it
  - Ex: Printer
- In space - Instead of the customers taking turns, each one gets part of the resource
- Enough memory to hold multiple programs? More efficient to hold several programs in memory than give one of them the entire memory
  - Ex: disks and Flash drives

The Operating System as a Resource Manager

Program 1 … Program i … Program n

Extended Machine Interface (Resource Abstraction)

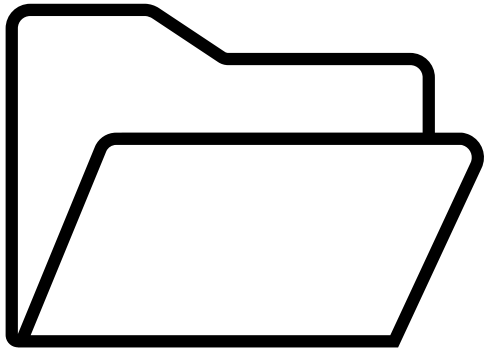


Time-multiplexed CPU resource

Space-multiplexed memory resource

Files

- Another layer of abstraction for using disks
- Using this abstraction, programs can create, write, and read files
  - No need to worry about the details of how the hardware works



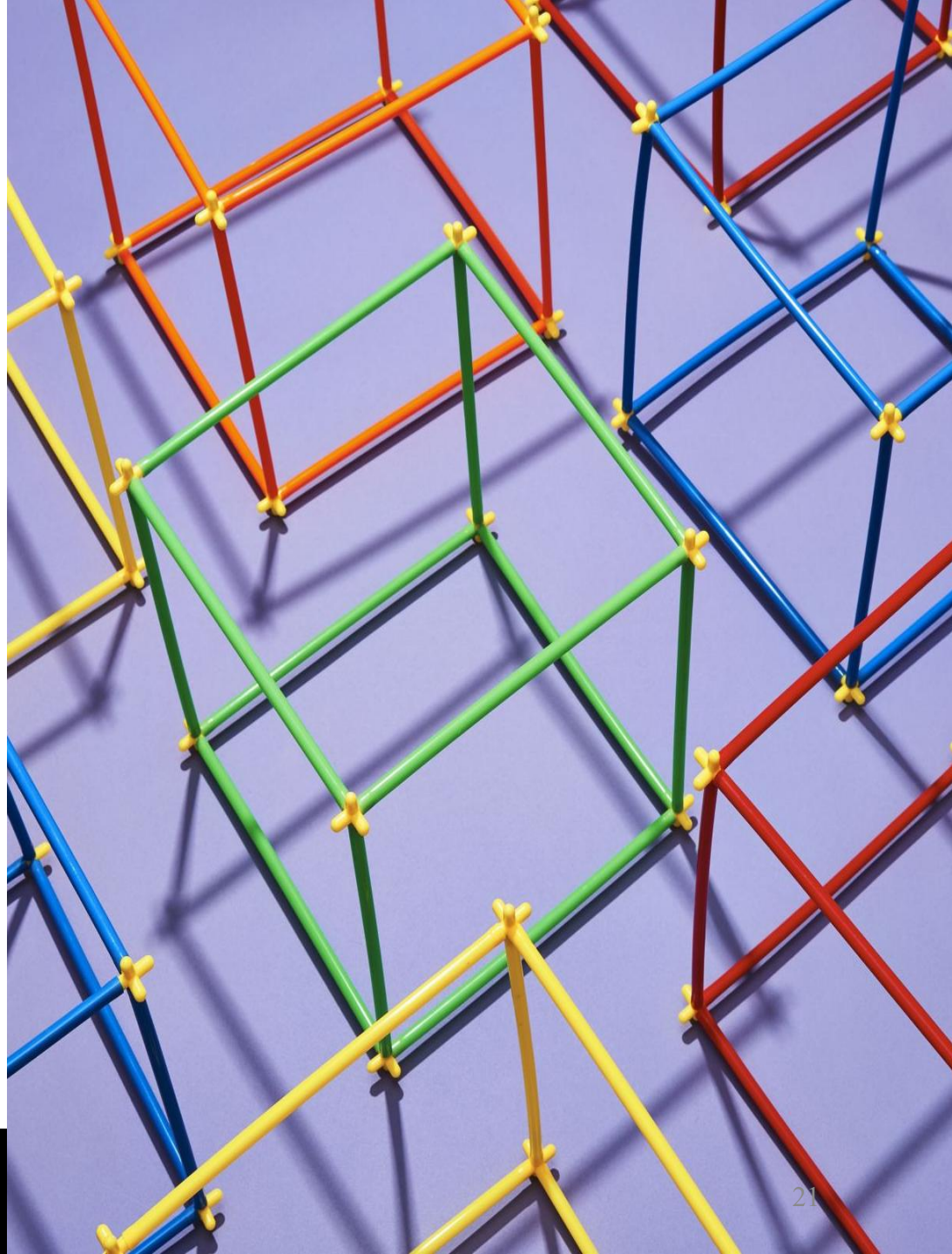
Why Resource Abstraction?

Resource abstraction

- Mask complexity
- Cover multiple devices
- Reliability

Resource sharing

- Efficiency
- Fairness
- Protection and security



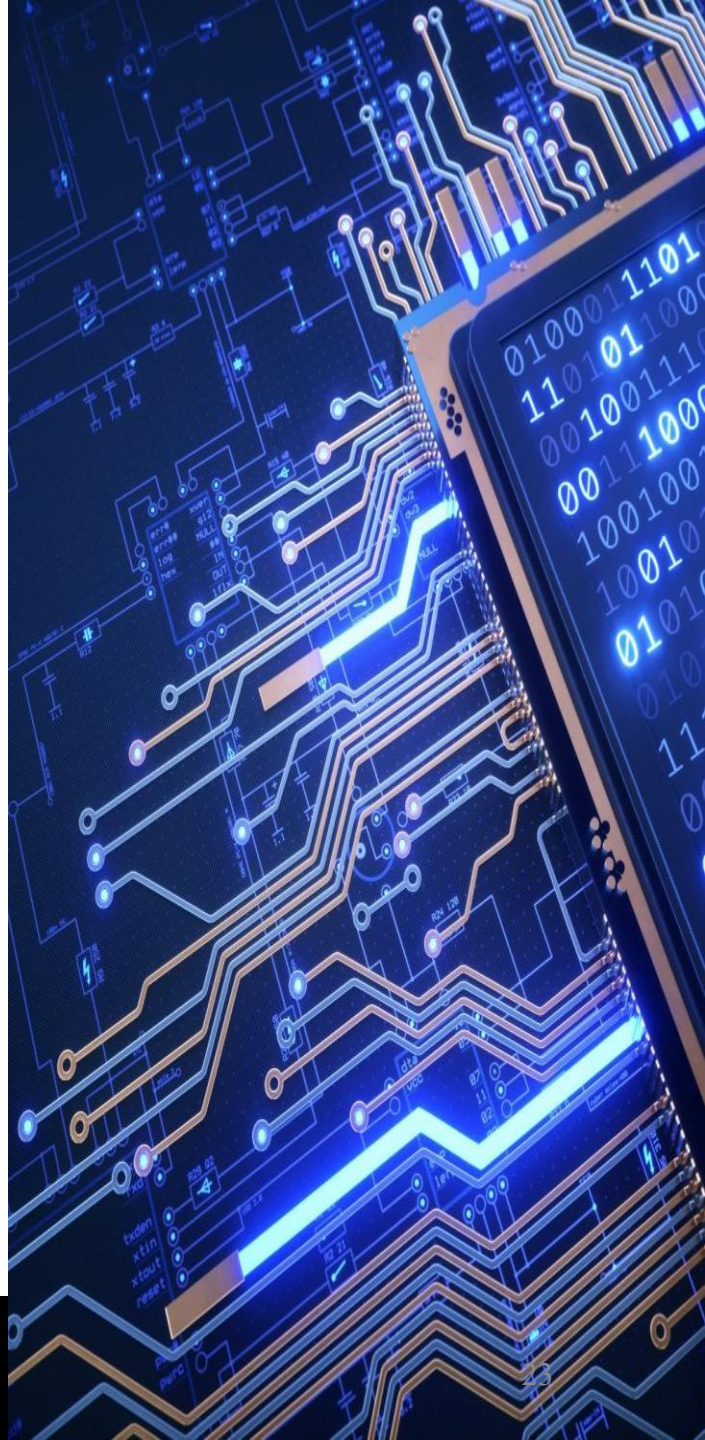
Creates two manageable tasks from many

Abstraction

1. Defines and implements the abstractions
2. Using these abstractions to solve the problem

How does an OS work?

- Computers have two modes of operation
  - User mode (application)
  - Kernel mode (OS kernel)
- Transition between user/kernel mode
  - Interrupt – HW device requests OS services (asynchronous/interrupt-driven)
  - Trap – user program requests OS services (synchronous/blocking)
  - Exception – error handling



Different Types of OS

Batch processing

- Processes jobs one by one

Time sharing OS

- Processes multiple jobs in “round robin”

Real-time OS

- Still time-sharing, but has deadlines for certain jobs

Distributed OS

- Multiple computers run a single copy of OS

Embedded OS

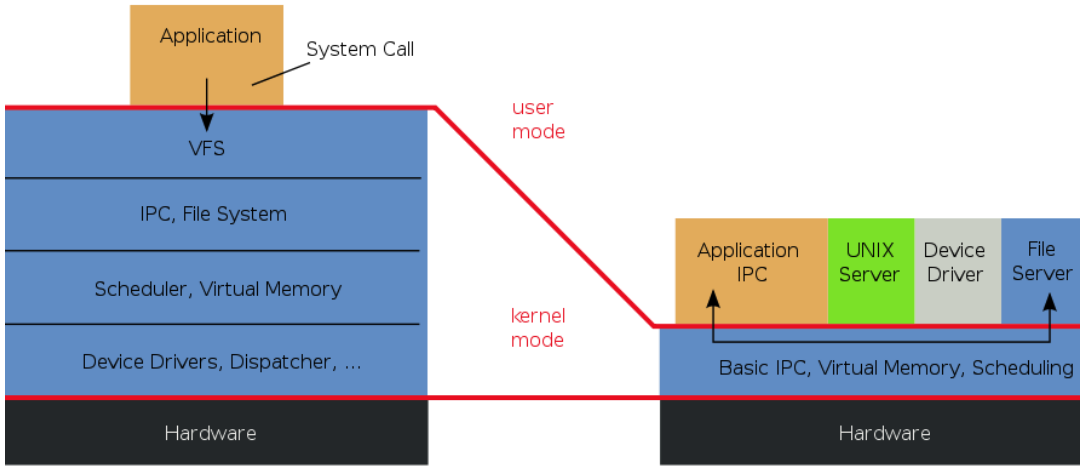
- Runs on (old) cell phones, PDAs, tailored and highly efficient

The Structure of OS

Monolithic Kernel based Operating System

Microkernel based Operating System

Hybrid Kernel



UNIX, Linux, Windows 98

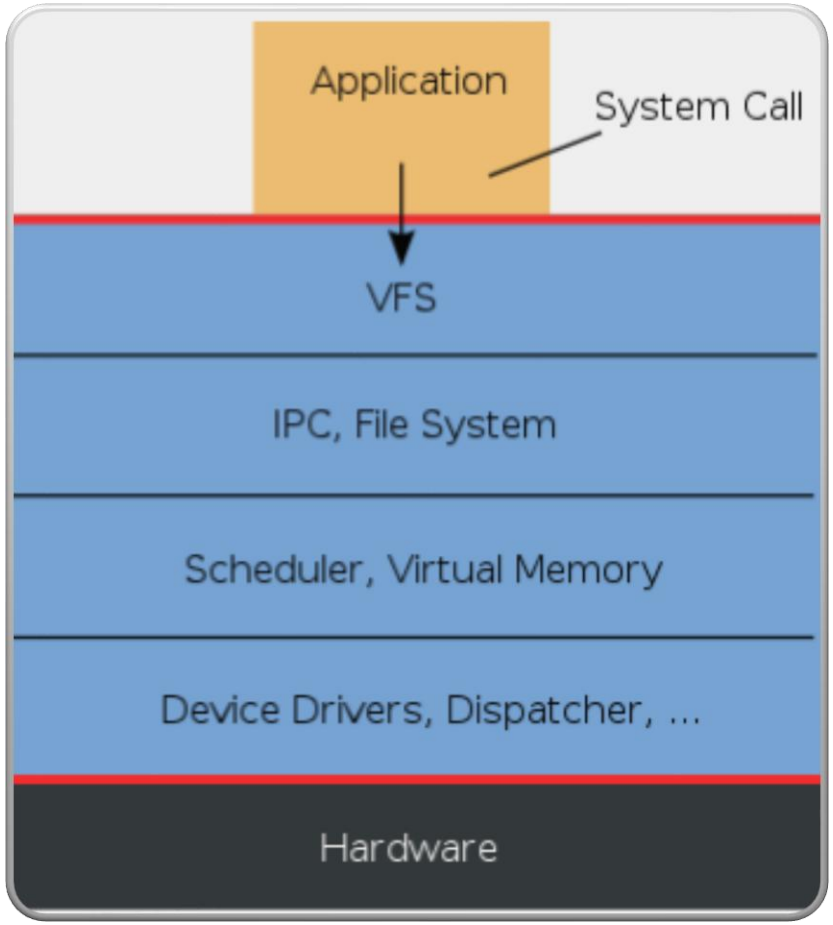
Mach

Windows NT, Mac OS X

Advantage v.s. disadvantage?

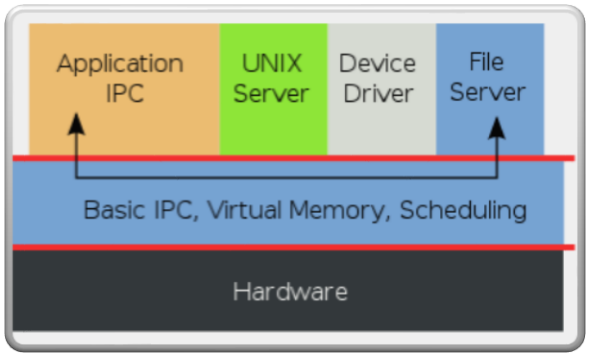
The Structure of OS

- A large process running entirely in a **single address space** (a single binary file)
- All kernel services execute in kernel address space
- Pros: fast, easy to implement
- Cons
  - Huge kernel, harder to maintain
  - No protection between kernel components
  - Complex dependencies among components, not easily extensible



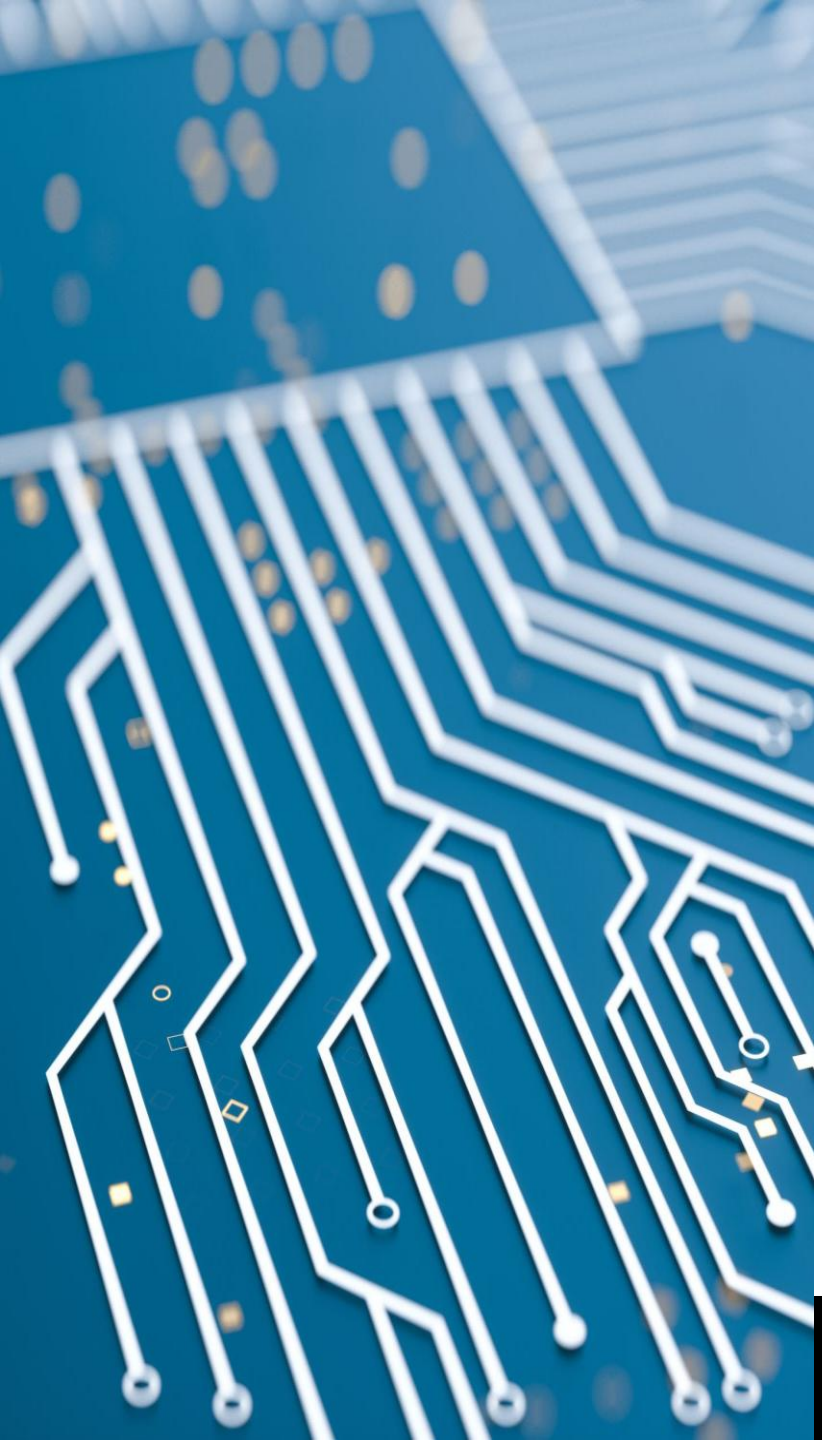
Monolithic kernel

The Structure of OS



Microkernel

- Kernel broken down to separate processes (aka servers)
- Servers kept separate and run in different address spaces
- Communication is done via message passing
  - Servers communicate through IPC (Inter-process Communication)
- Pros
  - Modular design, easily extensible
  - Easy to maintain
  - More reliable and secure
- Cons: Performance loss, complicated process management



Summary

- An OS is just a special program
  - Two functionalities: resource abstraction and sharing
  - Provides services to user programs
- Three ways to request OS services
  - Interrupt, trap, and exception
- Next class
  - Overview of computer hardware
  - Organization of operating systems

For Next Time!

- Review syllabus
- Determine the access method for the textbook
- Read Chapter 1
- Complete Chapter 1 HW

Course Content

Unit	Module	Week	Topic
1 – Introduction	1 & 2	1	Welcome to OS & Introduction
	2	2	OS Overview
2 – Processes and Threads	3	3	Processes
	4	4	Threads
	5	5	CPU Scheduling
	6	6	Multiprocessor Scheduling
	7	7	Interprocess Communication
	8	8	Pthread
3 – Deadlocks & Memory	9	9	Deadlock
	10	10	Memory Management
	--	11	Midterm Exam
4 – Additional Topics	11	11	Page Replacement
	12	13	File Systems
	13	14	IO Devices
	14	15	Security
	--	16	Final Exam