Operating Systems Orientation

CS 241

January 25, 2012

Copyright ©: University of Illinois CS 241 Staff

Objectives

- Explain the main purpose of operating systems and describe milestones of OS evolution
- Explain fundamental machine concepts
  - Instruction processing
  - Memory hierarchy
  - Interrupts
  - I/O
- Explain fundamental OS concepts
  - System calls
  - Processes
  - Synchronization
  - Files
- Explain the POSIX standard (UNIX specification)

Copyright ©: University of Illinois CS 241 Staff

OS Structure

Application Software

Firefox

Second Life

Yahoo Chat

GMail

Standard Operating Sy… …erface

Operating Sy…

Machine Independent

Read/Write

St… Output

Device Control

File System

Communication

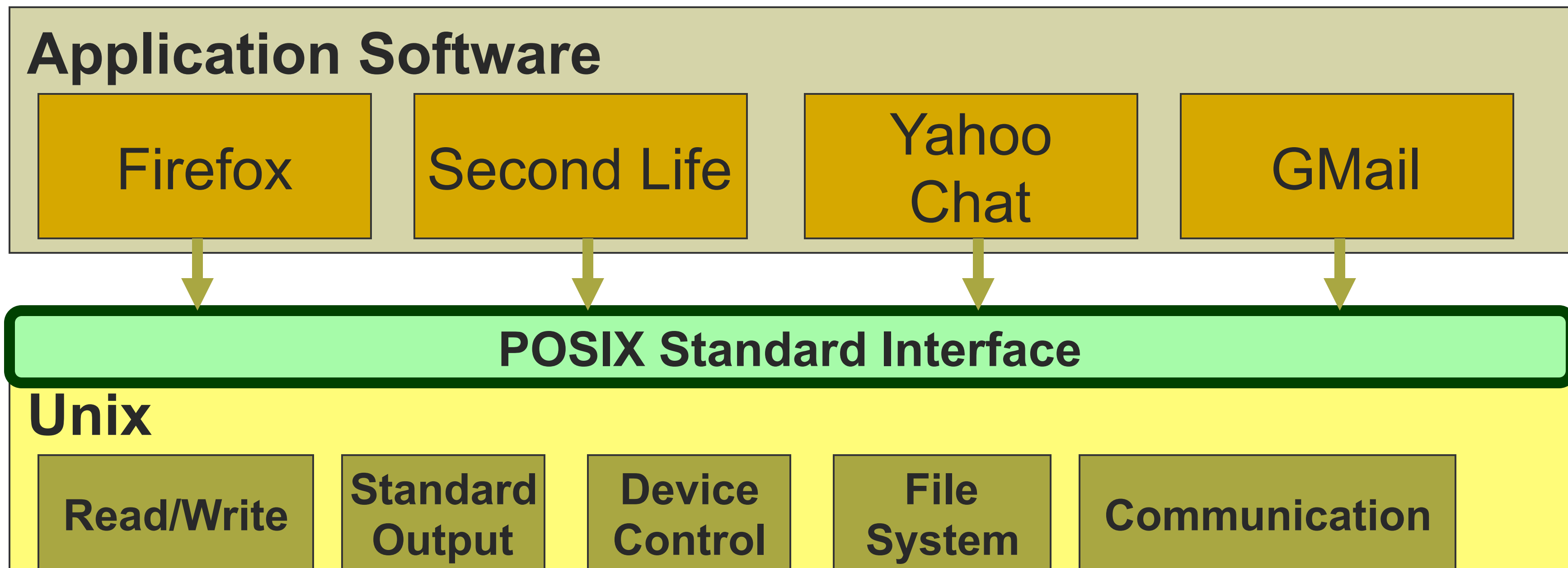
Machine Specific

Hardware

Network

Copyright ©: University of Illinois CS 241 Staff

POSIX
The UNIX Interface Standard



Copyright ©: University of Illinois CS 241 Staff

What is an Operating System?

- It is an *extended machine*
  - Hides the messy details that must be performed
  - Presents user with a virtualized and simplified abstraction of the machine, easier to use

- It is a *resource manager*
  - Each program gets time with the resource
  - Each program gets space on the resource

Copyright ©: University of Illinois CS 241 Staff

A Peek into Unix

Application

Libraries

User space/level

Portable OS Layer

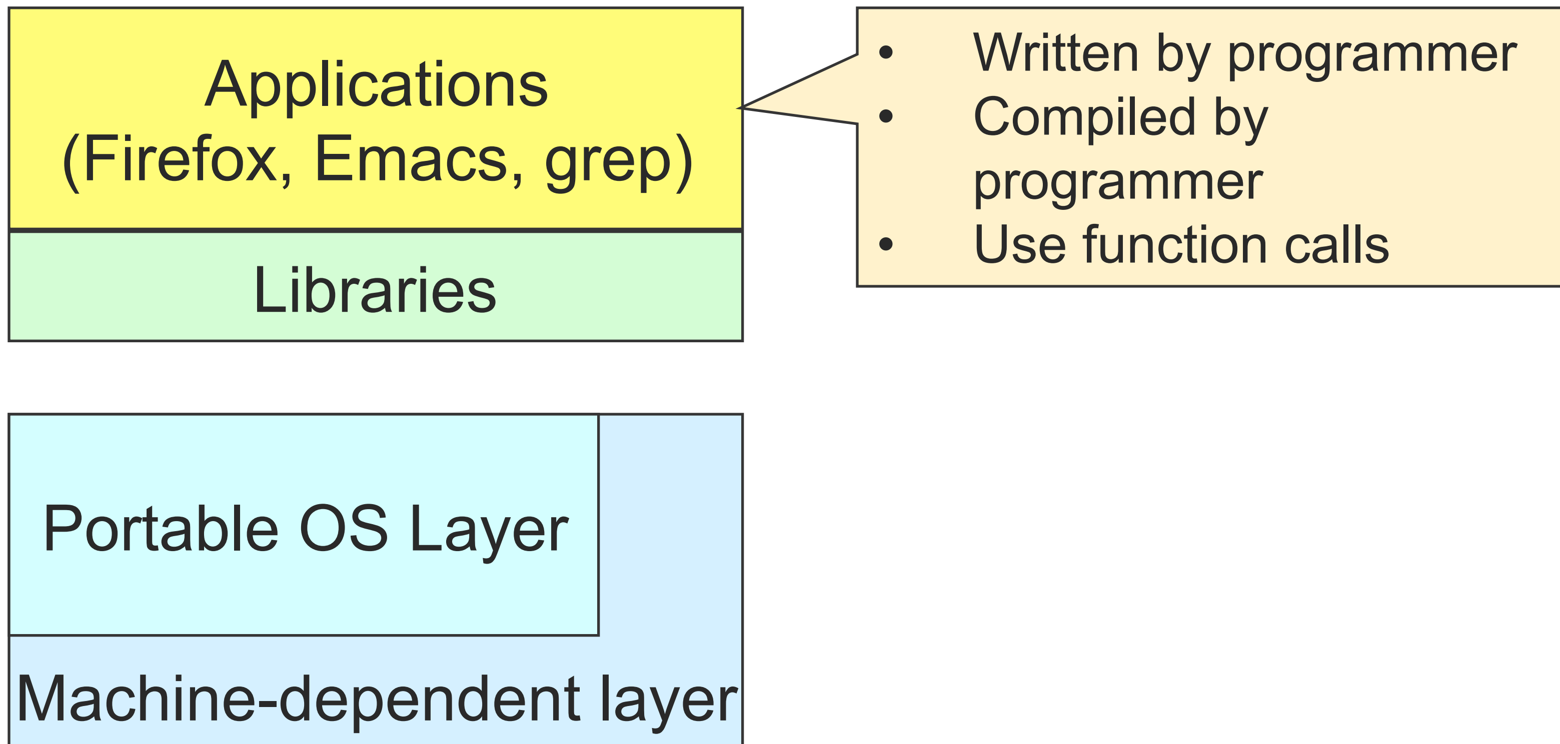
Machine-dependent layer

Kernel space/level

- **User/kernel modes are supported by hardware**
- **Some systems do not have clear user-kernel boundary**

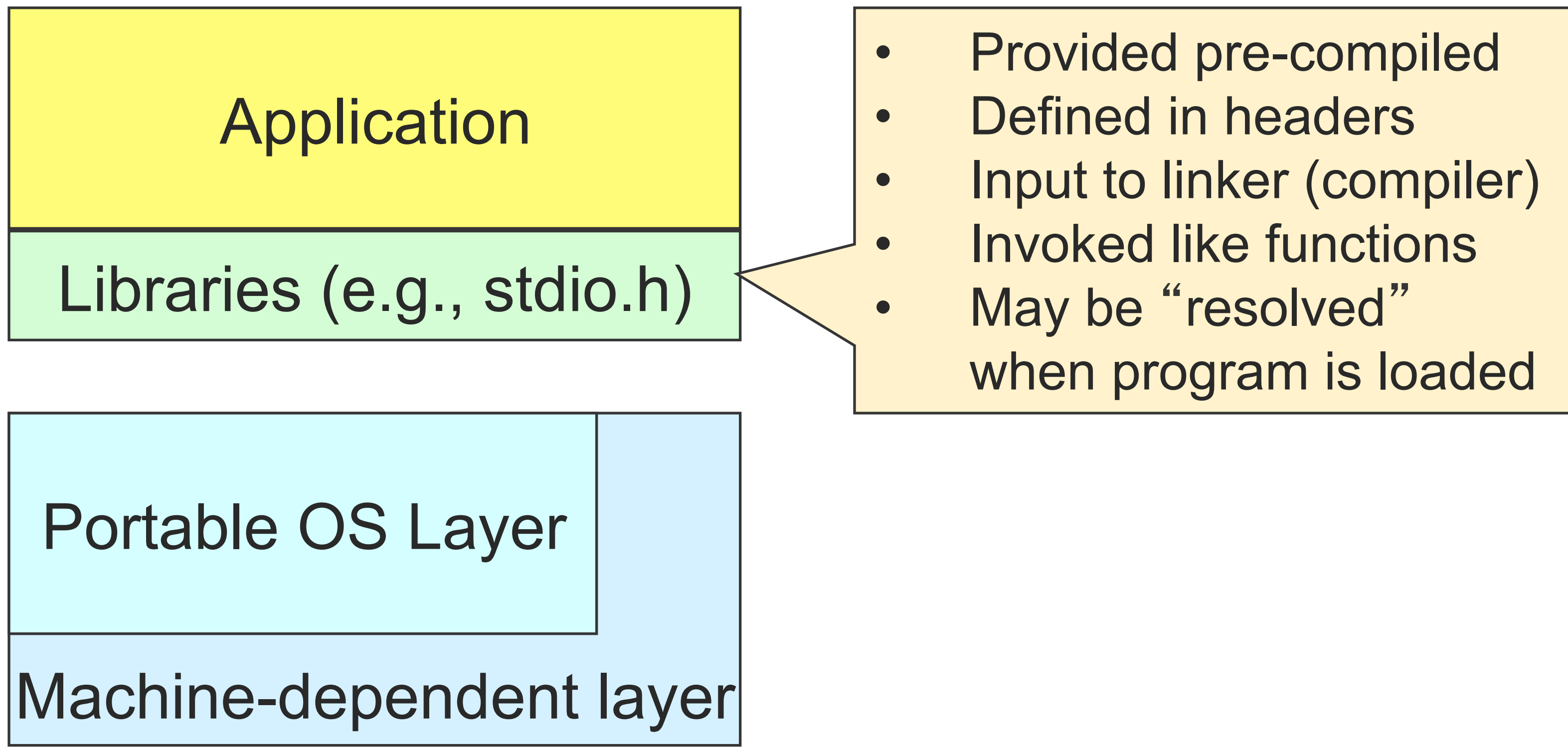
Copyright ©: University of Illinois CS 241 Staff

Application



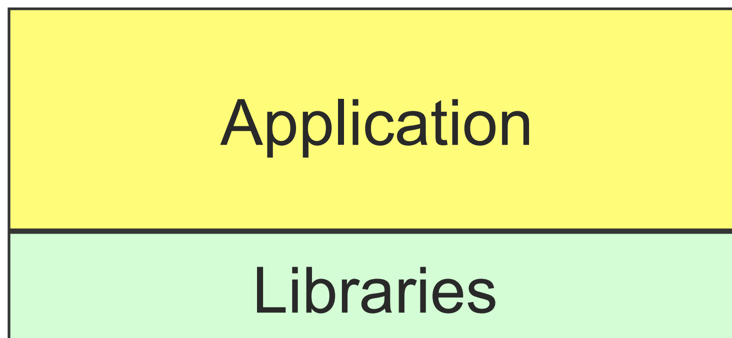
Copyright ©: University of Illinois CS 241 Staff

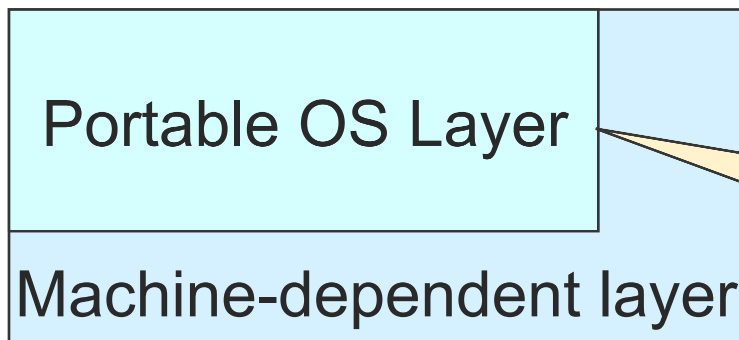
Unix: Libraries



Copyright ©: University of Illinois CS 241 Staff

Typical Unix OS Structure

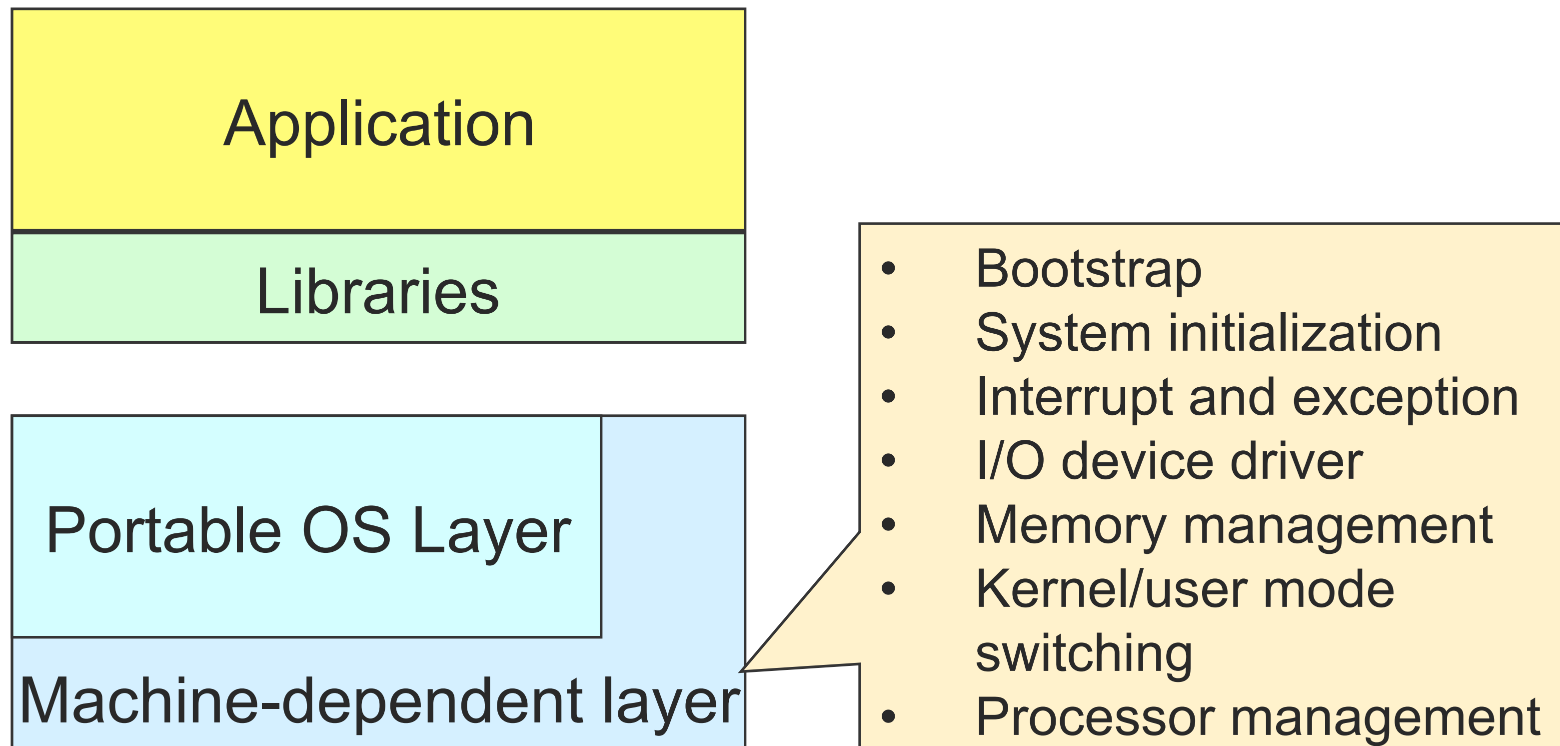




- System calls (read, open..)
- All “high-level” code

Copyright ©: University of Illinois CS 241 Staff

Typical Unix OS Structure



Copyright ©: University of Illinois CS 241 Staff

History of Computer Generations

- Pre-computing generation 1792 - 1871
  - Charles Babbage's "Analytical Engine"
  - Purely mechanical
  - Designed, but never actually built
    - Required high-precision gears/cogs that didn't exist yet
  - A man before his time
    - When this works, we'll need software!
    - First programming language
    - World's first programmer: Ada Lovelace

Copyright ©: University of Illinois CS 241 Staff

History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
  - Vacuum tubes, relays, plug boards
  - Seconds per operation!
    - Focus on numerical calculations
  - No programming language
    - Everything done using pure machine language or wiring electrical circuits!
  - No operating system
    - Sign up for your time slot!
  - Progress: Punch cards!

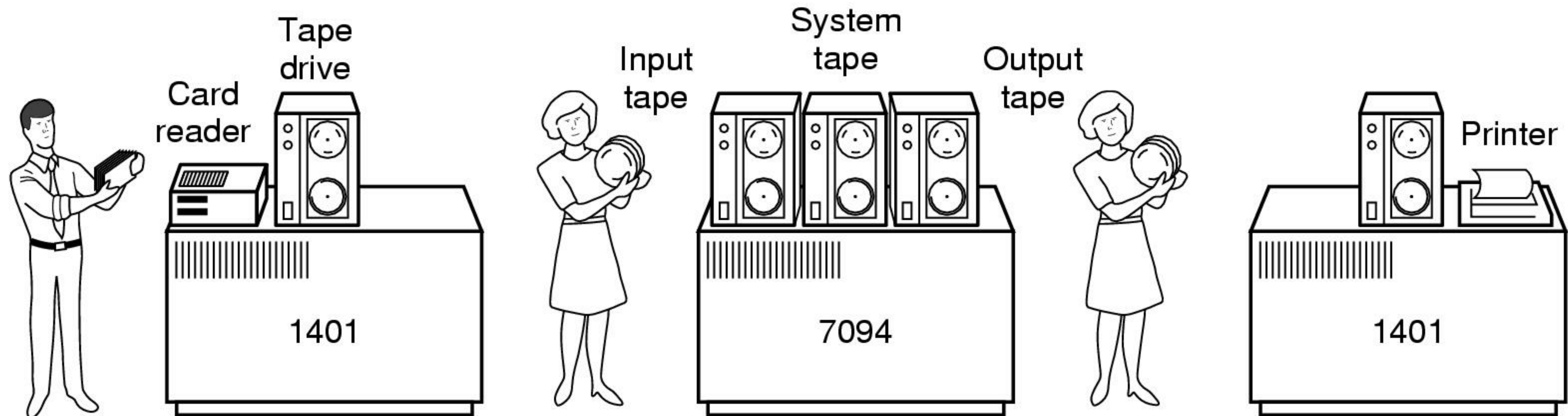
Copyright ©: University of Illinois CS 241 Staff

History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 - 1965
  - Transistors, mainframes
  - Large human component

Copyright ©: University of Illinois CS 241 Staff

History of Operating Systems

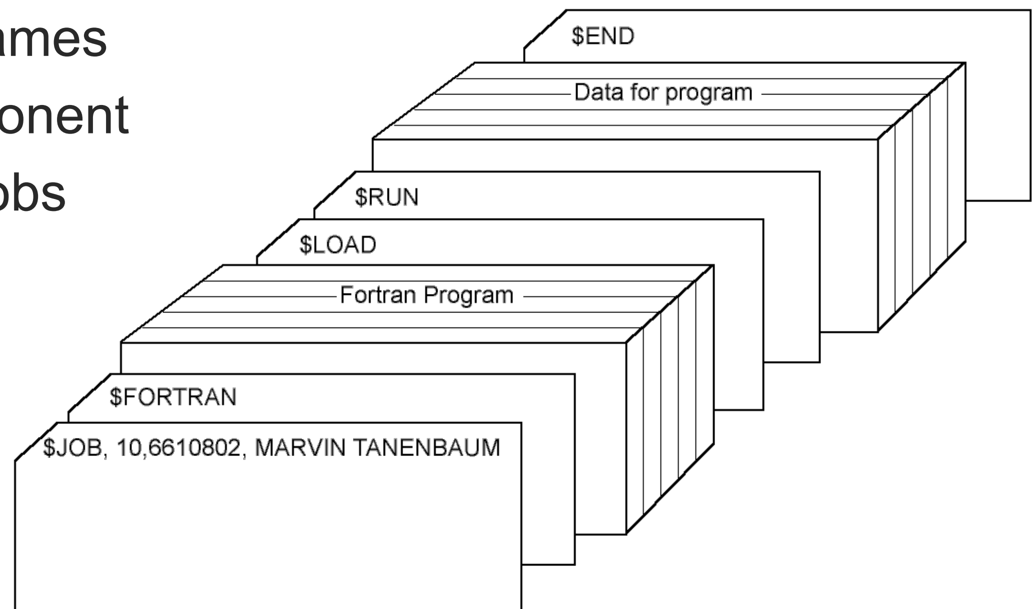


- Problem: a lot of time wasted by operators walking around machine room
- One solution: An early “batch system” by IBM
  - bring cards to 1401
  - read cards to tape
  - put tape on 7094 which does computing
  - put tape on 1401 which prints output

Copyright ©: University of Illinois CS 241 Staff

History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 - 1965
  - Transistors, mainframes
  - Large human component
  - Solution: Batched jobs



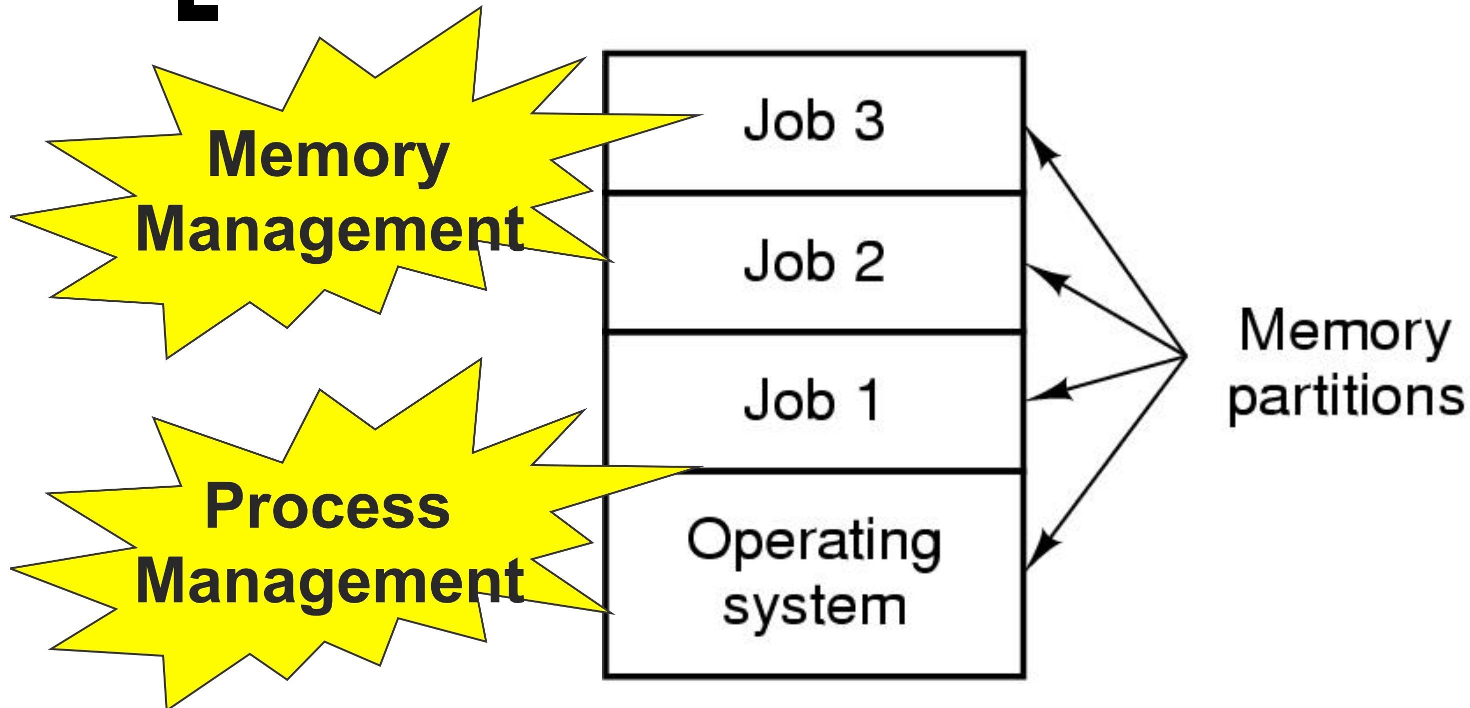
Copyright ©: University of Illinois CS 241 Staff

History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 – 1965
- Third generation 1965 – 1980
  - Integrated circuits and multiprogramming
  - IBM's New model: all software and OS must work on all platforms
    - A beast!
  - Progress: Multiprogramming
    - Keep the CPU busy

Copyright ©: University of Illinois CS 241 Staff

History of Operating Systems



- Multiprogramming/timesharing system
  - Three jobs in memory – 3rd generation

Copyright ©: University of Illinois CS 241 Staff

History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 – 1965
- Third generation 1965 – 1980
  - Integrated circuits and multiprogramming
  - IBM's New model: all software and OS must work on all platforms
  - Progress: Multiprogramming and timesharing
  - Progress: Spooling
    - Always have something ready to run
  - MULTICS + minicomputers == UNIX!

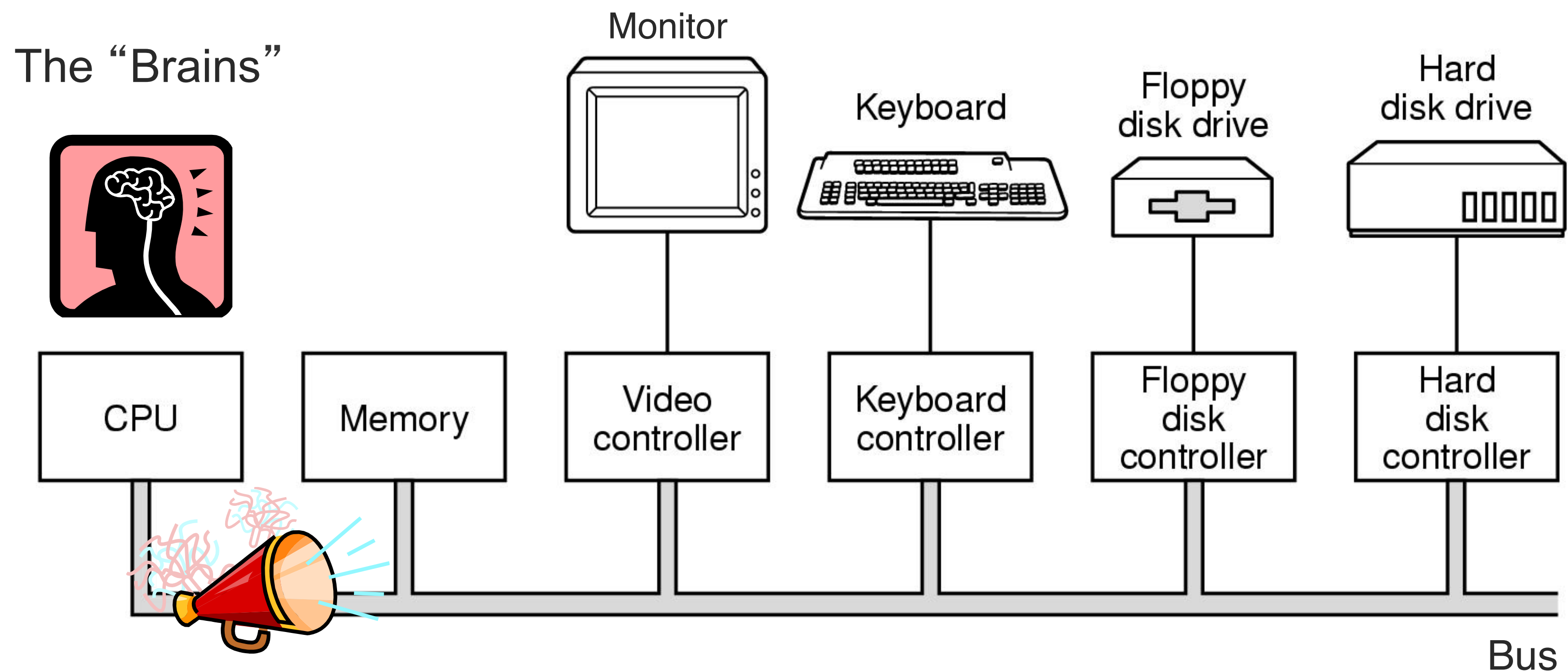
Copyright ©: University of Illinois CS 241 Staff

History of Computer Generations

- Pre-computing generation 1792 – 1871
- First generation 1945 – 1955
- Second generation 1955 – 1965
- Third generation 1965 – 1980
- Fourth generation 1980 – present
  - Personal computers
  - Multi-processors
  - Phones
  - …

Copyright ©: University of Illinois CS 241 Staff

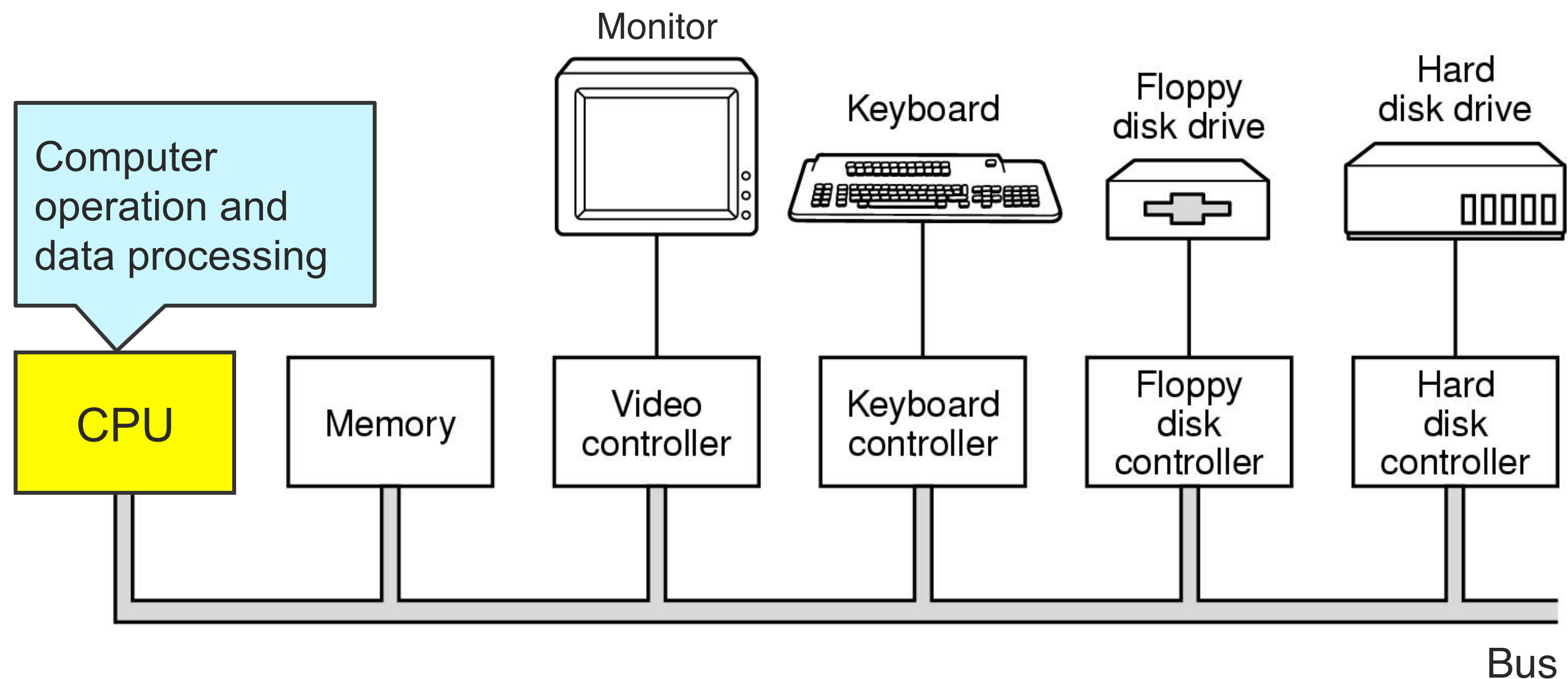
Computer Hardware Review



- Components of a simple personal computer

Copyright ©: University of Illinois CS 241 Staff

Computer Hardware Review



- Components of a simple personal computer

Computer Hardware Review

- Components of a simple personal computer

Early Pentium system

Cache bus
Local bus
Memory bus
Level 2 cache
CPU
PCI bridge
Main memory
PCI bus
SCSI
USB
ISA bridge
IDE disk
Graphics adaptor
Available PCI slot
Mon-itor
Mouse
Key-board
ISA bus
Modem
Sound card
Printer
Available ISA slot

Copyright ©: University of Illinois CS 241 Staff

CPU, From CS231

- Fetch instruction from code memory
- Fetch operands from data memory
- Perform operation (and store result)
- (Check interrupt line)
- Go to next instruction

- 'Conventional CPU'
  (Ignore pipeline, optimization complexities)

Copyright ©: University of Illinois CS 241 Staff

CPU Registers

- Fetch instruction from code memory
- Fetch operands from data memory
- Perform operation (and store result)
- Go to next instruction

- Note: CPU must maintain certain state
  - Current instructions to fetch (program counter)
  - Location of code memory segment
  - Location of data memory segment

Copyright ©: University of Illinois CS 241 Staff

CPU Register Examples

- Hold instruction operands
- Point to start of
  - Code segment (executable instructions)
  - Data segment (static/global variables)
  - Stack segment (execution stack data)
- Point to current position of
  - Instruction pointer
  - Stack pointer

Copyright ©: University of Illinois CS 241 Staff

CPU Register Examples

- Hold instruction operands
- Point to start of
  - Code segment
  - Data segment
  - Stack segment
- Point to current position of
  - Instruction pointer
  - Stack pointer
    - Why stack?

Sample Layout for program image in main memory

Address	Segment	Description
High address	Command-line arguments and environment variables	← `argc, argv`, environment
	stack ↓	← Activation record for function calls (return address, parameters, saved registers, automatic variables
	↑ heap	← Allocations from `malloc` family
	Uninitialized static data	
	Initialized static data	
Low address	Program text (code segment)	

Processes have three segments: text, data, stack

Copyright ©: University of Illinois CS 241 Staff

Memory Hierarchy

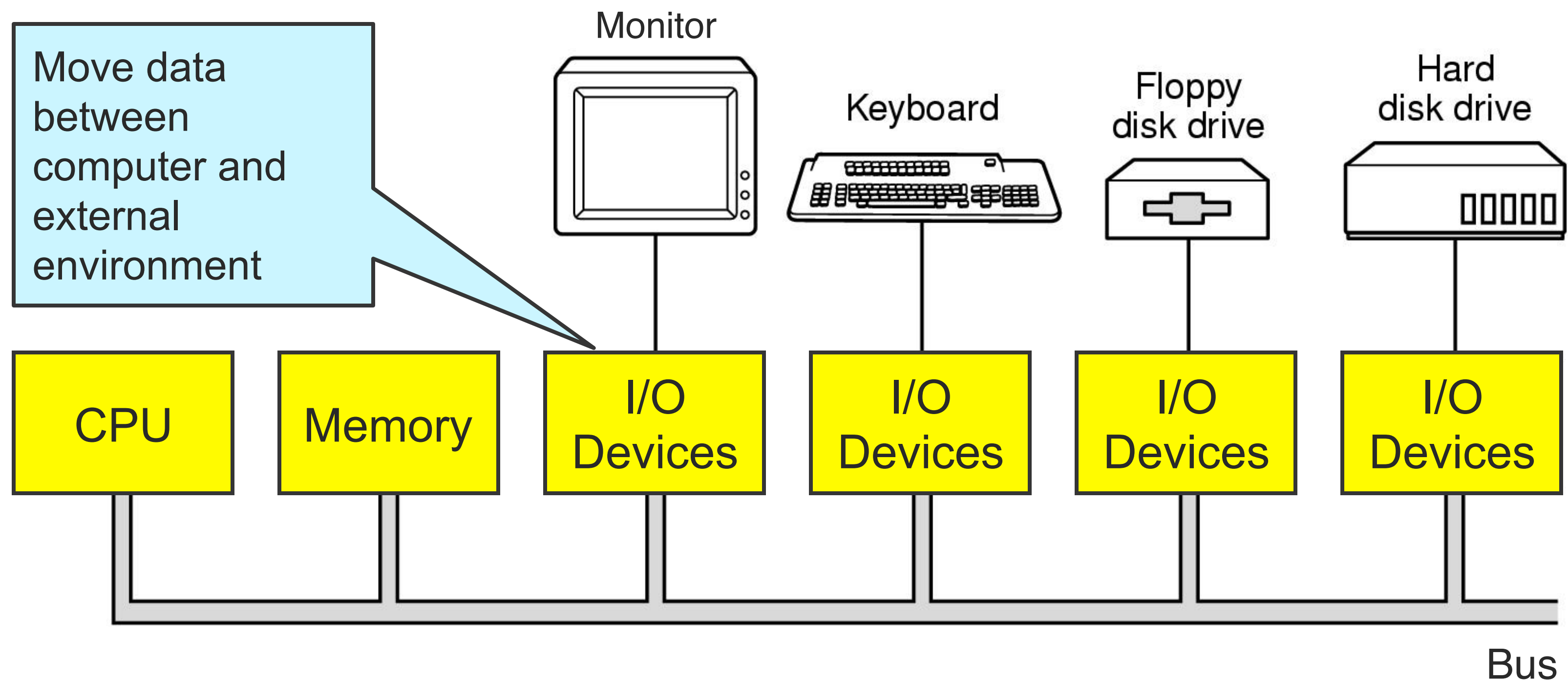
- Leverage locality of reference

1. Decreasing cost per bit
2. Increasing capacity
3. Increasing access time
4. Decreasing frequency of access

Typical access time		Typical capacity
1 nsec	Registers	1 KB
2 nsec	Cache	128 MB
10 nsec	Main memory	4 GB
10 msec	Magnetic disk	1TB
100 sec	Magnetic tape	10 TB

Copyright ©: University of Illinois CS 241 Staff

Computer Hardware Review



- Components of a simple personal computer

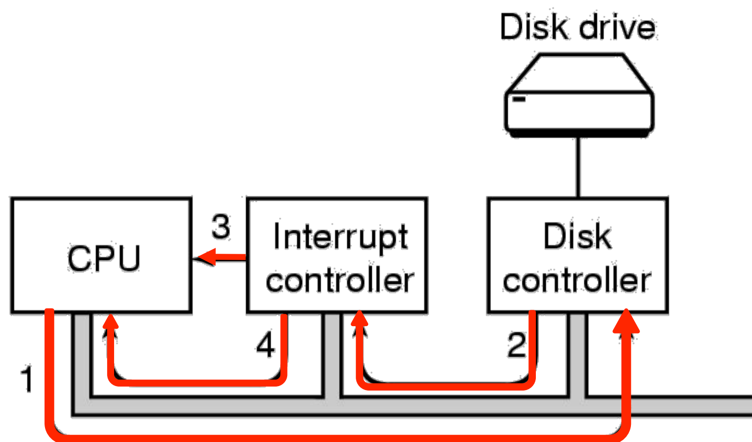
Copyright ©: University of Illinois CS 241 Staff

I/O Device Access

- System Calls
  - Application makes a system call reading from or writing to a device
  - Kernel blocks application
  - Kernel translates request to specific driver
  - Driver starts I/O
  - Polls device for completion
    - Or waits for **interrupt** from device
  - Kernel obtains results, un-blocks application

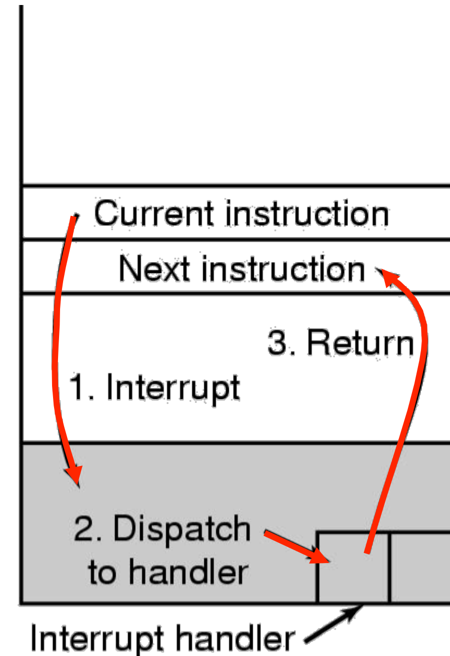
Copyright ©: University of Illinois CS 241 Staff

I/O Interrupt Mechanism



(a)

(b)



1. Application writes into device registers, Controller starts device
2. When done, device controller signals interrupt controller
3. Interrupt controller asserts pin on CPU
4. Interrupt controller puts I/O device number on bus to CPU

Copyright ©: University of Illinois CS 241 Staff

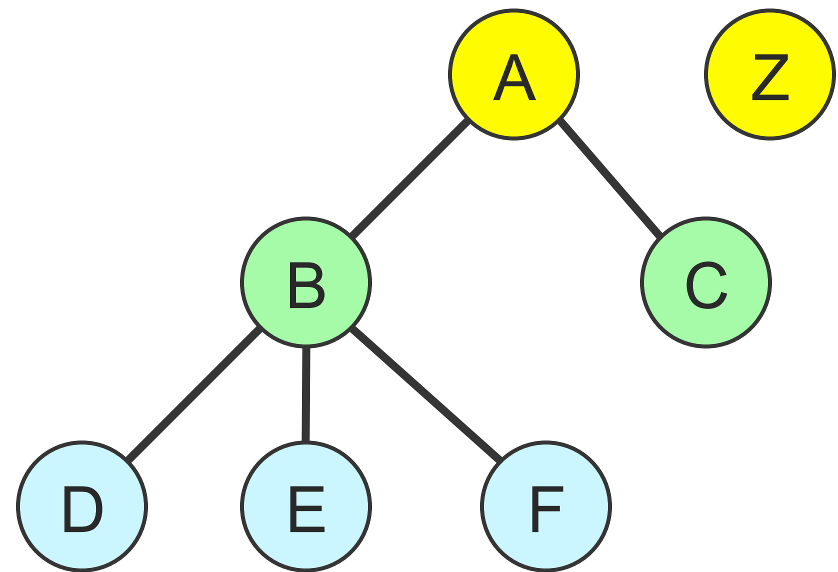
Operating System Concepts

- Shared resources
  - I have B KB of memory, but need 2B KB
  - I have N processes trying to access the disk at the same time
  - How would you control access to resources?
- Challenges
  - Who gets to use the resources?
  - How do you control fair use of the resources over time?
  - Deadlock

Copyright ©: University of Illinois CS 241 Staff

Operating System Concepts

- Process
  - An executable instance of a program
  - Only one process can use a (single-core) CPU at a time
- A process tree
  - A created two child processes, B and C
  - B created three child processes, D, E, and F



Copyright ©: University of Illinois CS 241 Staff

Operating System Concepts

- How would you switch CPU execution from one process to another?
- Solution: Context Switching
  - Store/restore state on CPU, so execution can be resumed from same point later in time
  - Triggers: multitasking, interrupt handling, user/kernel mode switching
  - Involves: Saving/loading registers and other state into a "process control block" (PCB)
  - PCBs stored in kernel memory

Copyright ©: University of Illinois CS 241 Staff

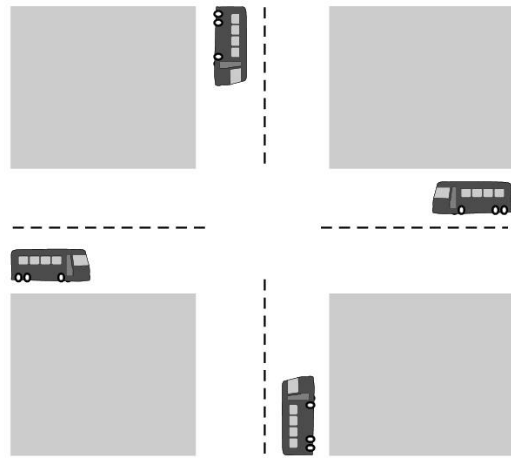
Operating System Concepts

- Context Switching
  - What are the costs involved?

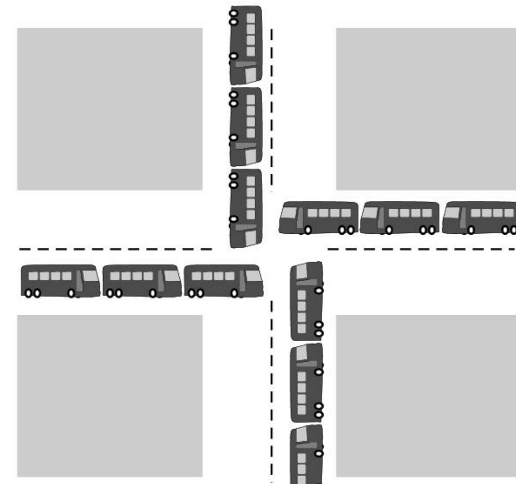
Item	Time	Scaled Time in Human Terms (2 billion times slower)
Processor cycle	0.5 ns (2 GHz)	1 s
Cache access	1 ns (1 GHz)	2 s
Memory access	15 ns	30 s
Context switch	5,000 ns (5 micros)	167 m
Disk access	7,000,000 ns (7 ms)	162 days
System quanta	100,000,000 (100 ms)	6.3 years

Copyright ©: University of Illinois CS 241 Staff

Operating System Concepts



(a) A potential deadlock

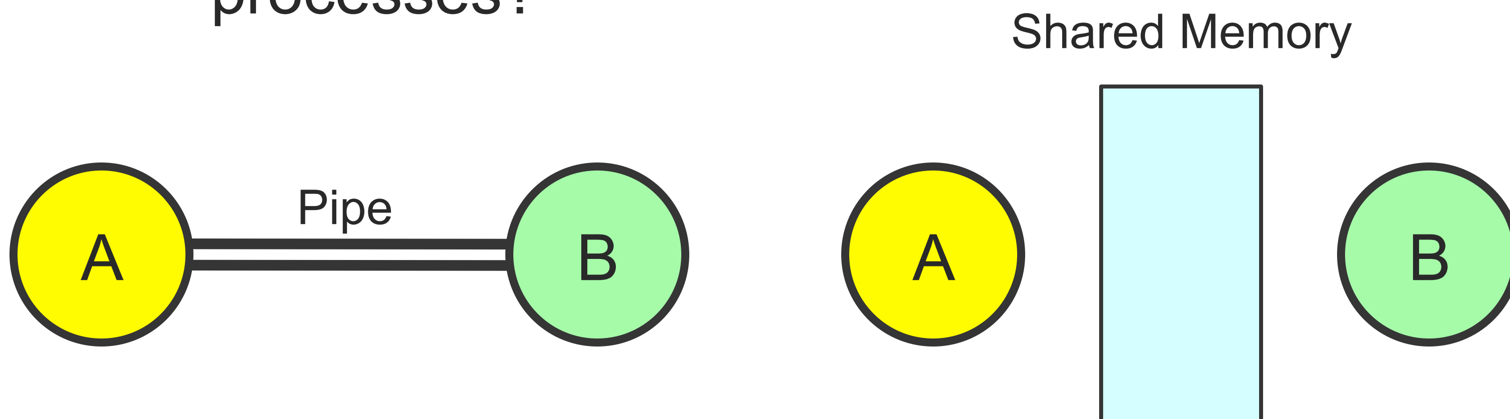


(b) An actual deadlock

- Another challenge: Deadlock
  - Set of actions waiting for each other to finish
- Example:
  - Process A has lock on file 1, wants to acquire lock on file 2
  - Process B has a lock on file 2, wants to acquire lock on file 1

Operating System Concepts

- Inter-process Communication
  - Now process A needs to exchange information with process B
  - How would you enable communication between processes?



Copyright ©: University of Illinois CS 241 Staff

Next up

- MP1 released
- Discussion sections tomorrow
- Friday: System calls

Copyright ©: University of Illinois CS 241 Staff

Summary

- Resource Manager
- Hardware independence
- Virtual Machine Interface
- POSIX
- Concurrency & Deadlock

Copyright ©: University of Illinois CS 241 Staff