OPERATING SYSTEMS

COMS W1001
Introduction to Information Science

Boyi Xie

Announcement

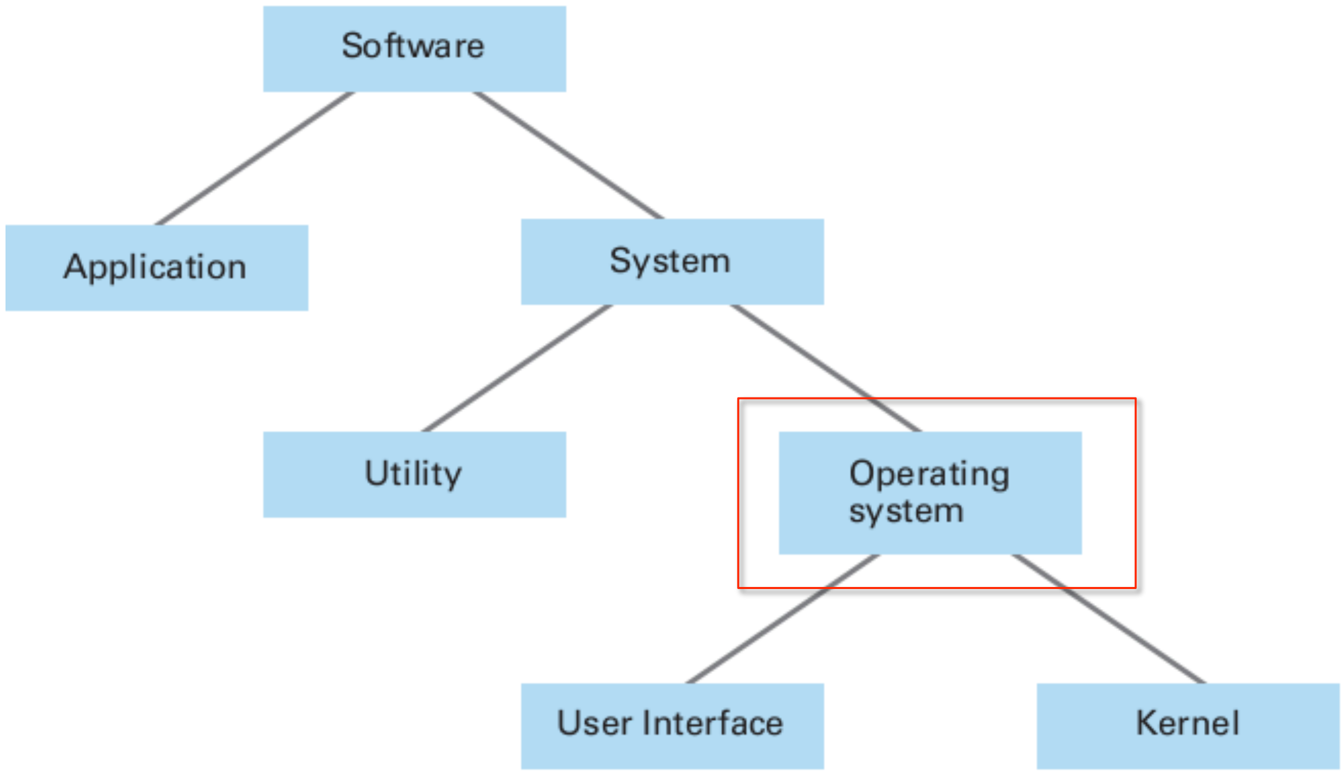
- Homework 1 is available
- Grace days
  - A total of 5 days for 5 HWs
  - If all grace days have been used, 50% of the points of that HW will be deducted for each late day
- Academic integrity

Von Neumann Architecture

- Named after the brilliant mathematician John Von Neumann, who first proposed it in 1946
- The Von Neumann architecture is a model for designing and building computers that is based on the following three characteristics:
  - A computer constructed from four major subsystems called **memory**, **input/output**, the **arithmetic/logic unit (ALU)**, and the **control unit**.
  - The **stored program concept**, in which the instructions to be executed by the computer are represented as binary values and stored in memory.
  - The **sequential execution of instructions**, in which one instruction at a time is fetched from memory to the control unit, where it is decoded and executed.

Operating System Basics

- OS in software classification

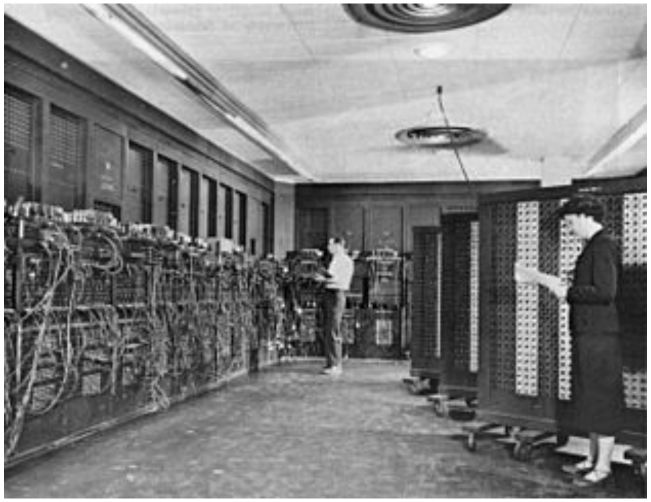


Today’s Topics

- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
  - Kernel and User Space
  - Processes
    - Process Administration
    - Coordinating the Machine’s Activities
    - Handling Competition Among Processes
  - Memory
  - File System
- Security

The History of Operating Systems

- When computers are born in the early days (e.g. 1940s)
  - Program execution requires significant preparation
    - Mounting magnetic tapes; placing punched cards in card readers; setting switches
  - Execution of each program (job) was handled as an isolated activity
  - Sign-up sheet for machine access
- OS for simplifying program setup and for streamlining the transition between jobs
  - Separation of users and equipment
  - Computer operator to operating the machine

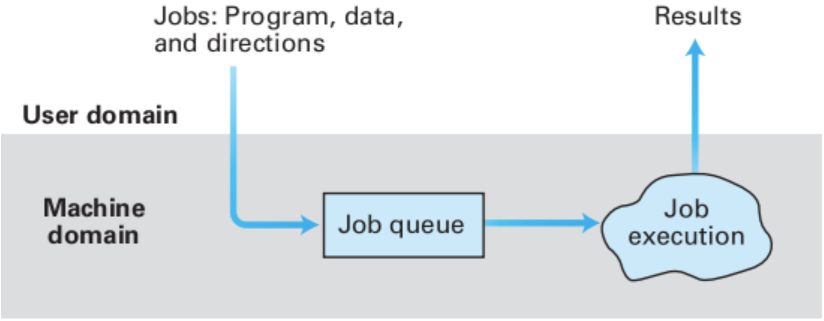


ENIAC

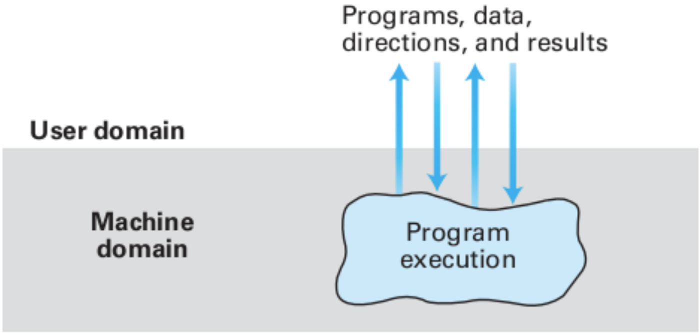
The History of Operating Systems

- Batch processing
  - Operator loads all materials
  - OS reads and executes them one at a time
  - Job queue – FIFO

Drawbacks: users have no interaction with jobs after submission



- Interactive processing
  - Terminals for user-computer interaction
  - Forced to execute tasks under a deadline – real-time processing



The History of Operating Systems

- Time-sharing
  - Provide service to multiple users at the same time
  - Implementation: to apply the technique of multiprogramming

- Multiprogramming technique
  - Time is divided into intervals and then the execution of each job is restricted to only one interval at a time
  - For single-user systems: multitasking – one user, multiple tasks
  - For multiuser systems

The History of Operating Systems

- Computer operators gave way to system administrators

- Operating systems have grown into complex systems that coordinate time-sharing, maintain programs and data files, and respond directly to requests from the users

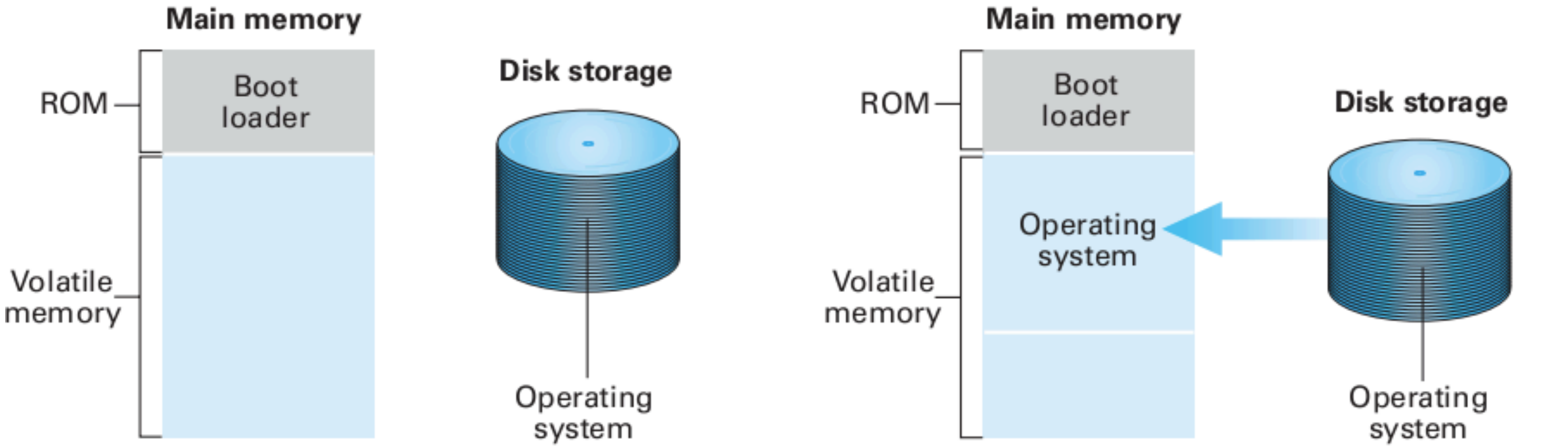
- Multiprocessor
  - Load balancing – dynamically allocating tasks to the various processors so that all processors are used efficiently
  - Scaling – breaking tasks into a number of subtasks compatible with the number of processors available

Today’s Topics

- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
    - Kernel and User Space
    - Processes
        - Process Administration
        - Coordinating the Machine’s Activities
        - Handling Competition Among Processes
    - Memory
    - File System
- Security

Getting Operating System Started

- Bootstrap
  - A program initially executed when the machine is turned on
  - Direct the CPU to transfer the OS from a predetermined location in mass storage into the volatile are of main memory
  - This above procedure is called boot strapping (booting)
  - Bootstrap resides in read-only memory (ROM)



Step 1: Machine starts by executing the boot loader program already in memory. Operating system is stored in mass storage.

Step 2: Boot loader program directs the transfer of the operating system into main memory and then transfers control to it.

Today's Topics

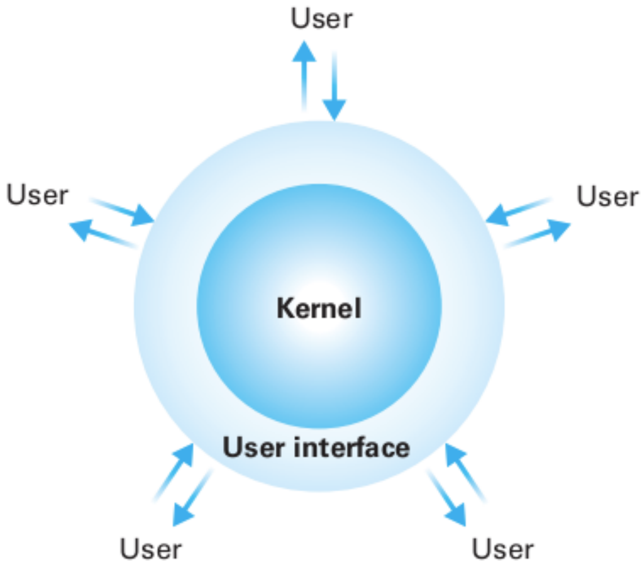
- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
    - Kernel and User Space
    - Processes
        - Process Administration
        - Coordinating the Machine's Activities
        - Handling Competition Among Processes
    - Memory
    - File System
- Security

Today’s Topics

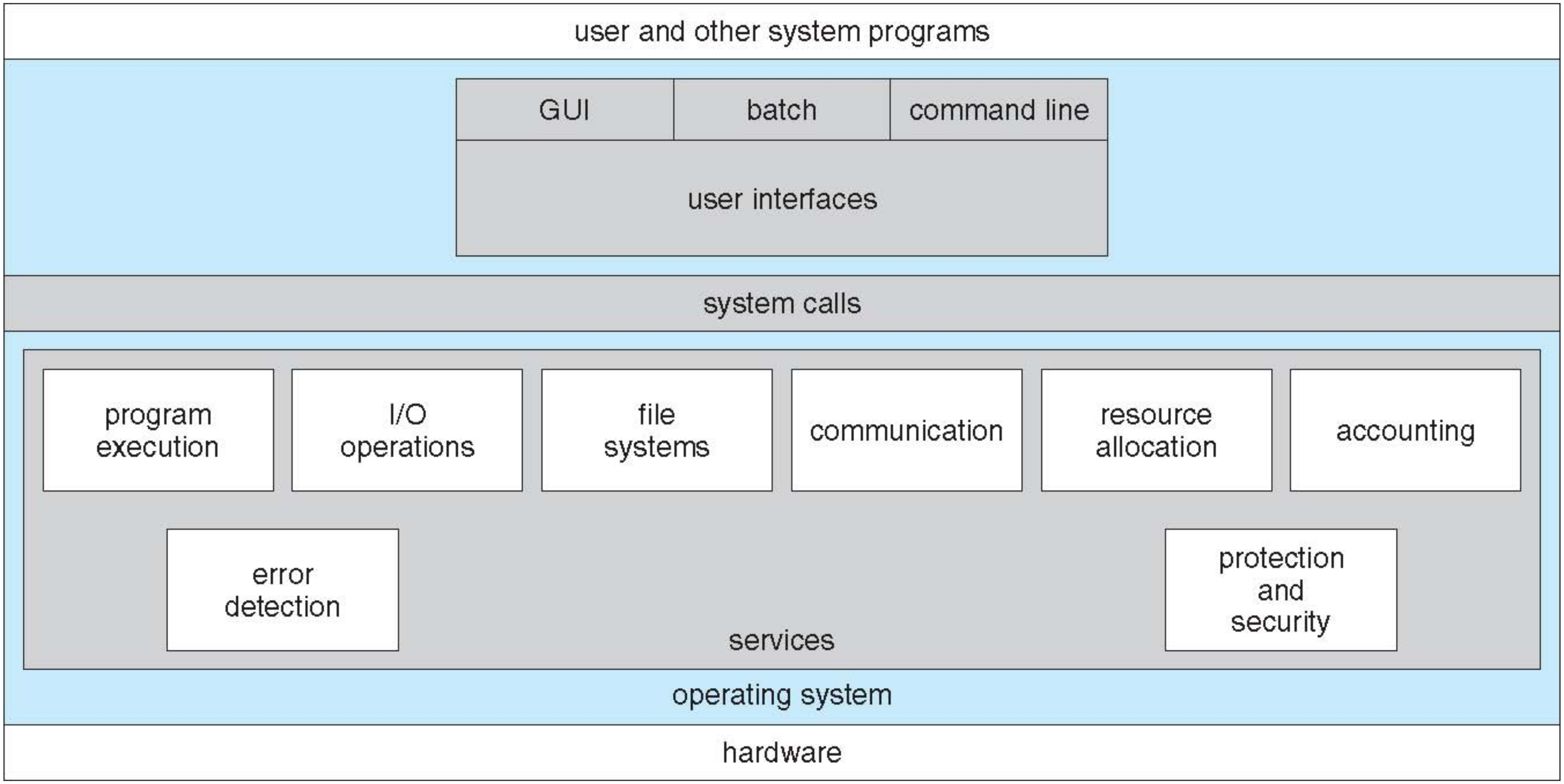
- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
  - Kernel and User Space
  - Processes
    - Process Administration
    - Coordinating the Machine’s Activities
    - Handling Competition Among Processes
  - Memory
  - File System
- Security

Components of an Operating System

- Shell
  - GUI (graphical user interface)
  - WIMP (windows, icons, menus, pointers)
  - Window manager
- Kernel
  - File manager
  - Device drivers
  - Memory manager
  - Scheduler – scheduling activities for execution
  - Dispatcher – allocation of time



User and Kernel Space



Today’s Topics

- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
  - Kernel and User Space
  - Processes
    - Process Administration
    - Coordinating the Machine’s Activities
    - Handling Competition Among Processes
  - Memory
  - File System
- Security

Coordinating the Machine’s Activities

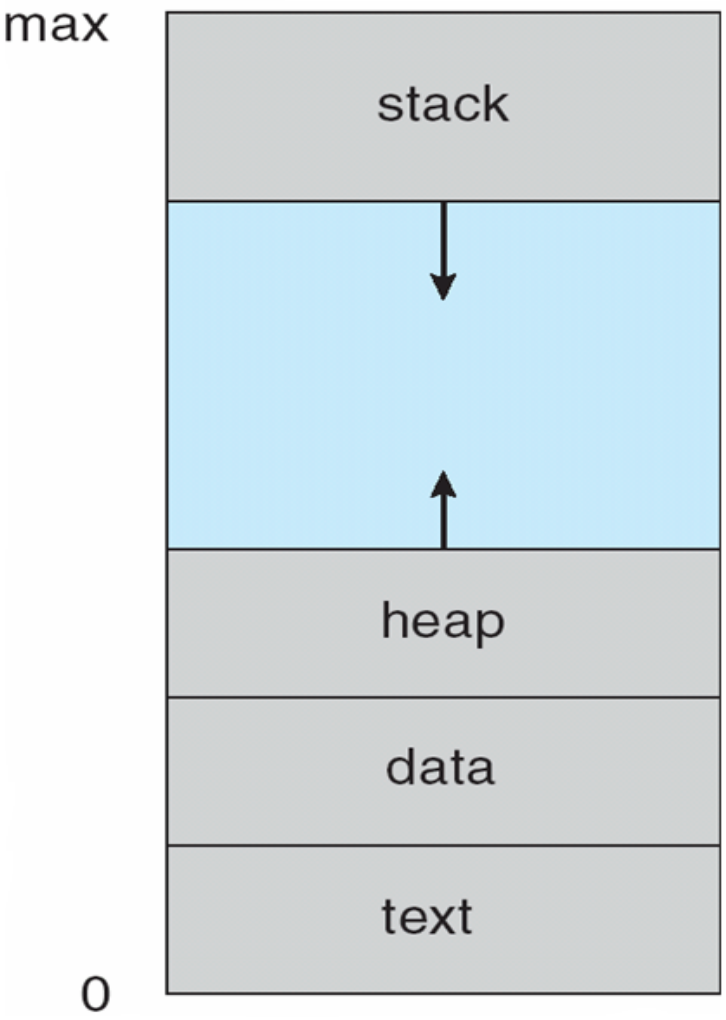
- Program
  - A static set of directions

- Process
  - A program in execution – a dynamic activity
  - Process state
    - Current status of the activity
    - Includes the position in the program being executed (program counter)
    - A snapshot of the machine at a particular time

- Thread
  - A lightwieght process (LWP) that shares with other threads of the same process its code section, data section, and other resources

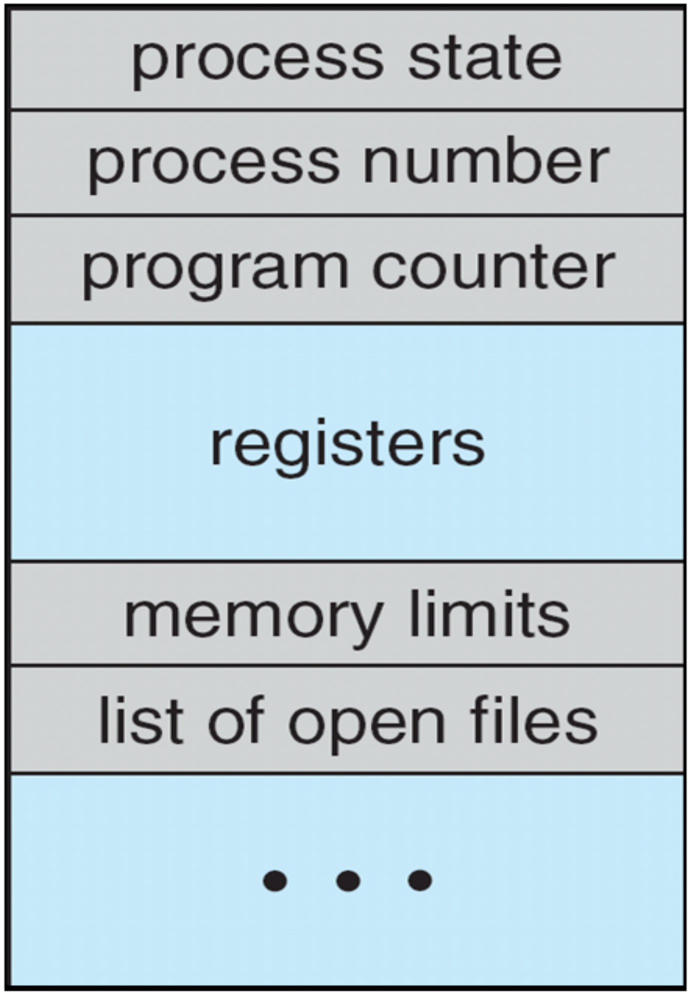
- It is a task for OS to mange processes so that they won’t compete for the computer’s resources

Process in Memory



- The program code, also called **text section**
- Current activity including **program counter**, processor registers
- **Stack** containing temporary data
  - Function parameters, return addresses, local variables
- **Data section** containing global variables
- **Heap** containing memory dynamically allocated during run time

Process Control Block (PCB)



process state

process number

program counter

registers

memory limits

list of open files

• • •

Process Administration

- Coordinating the execution of processes are handled by the scheduler and dispatcher within the kernel

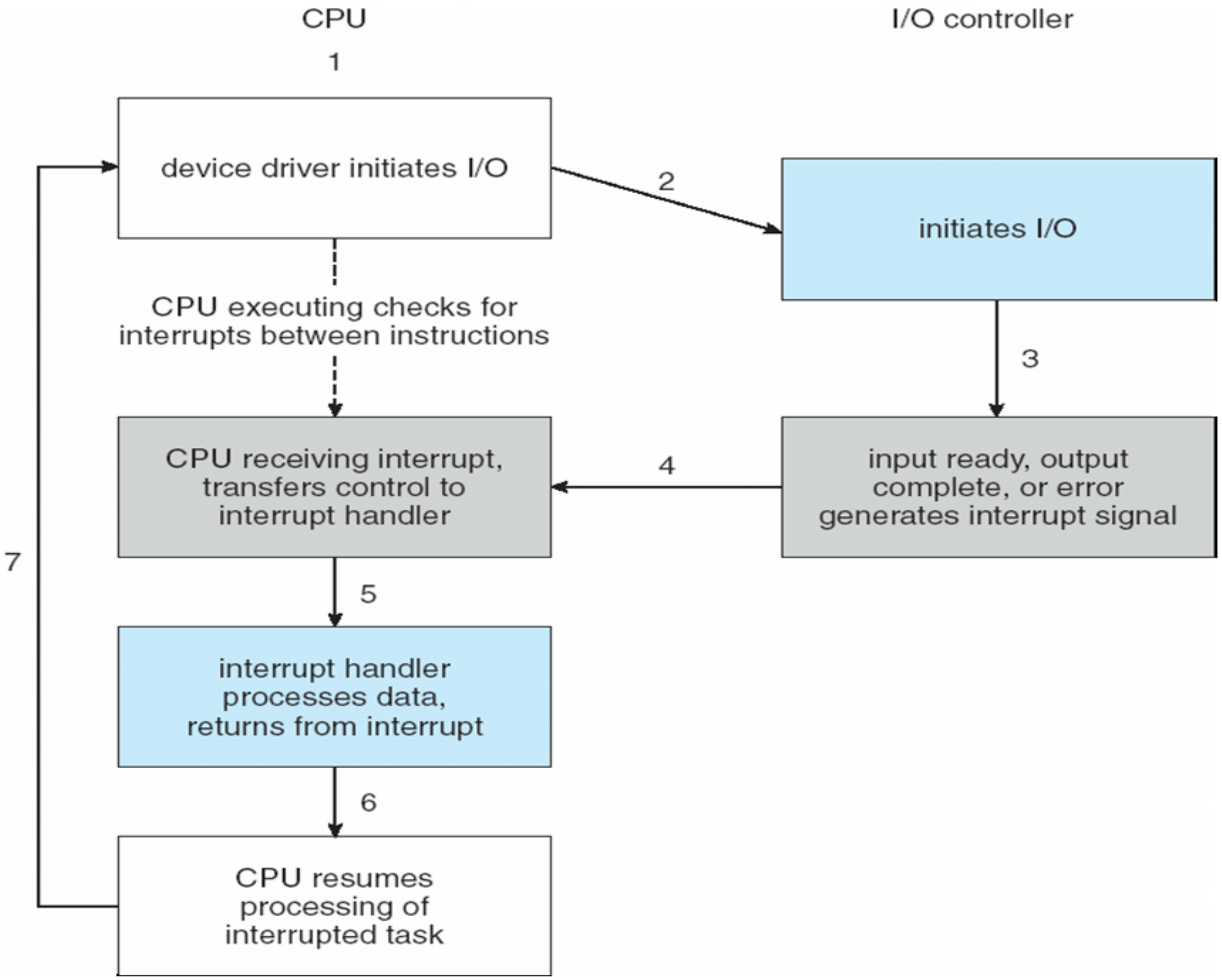
- Scheduler maintains process table
  - Each time the execution of a program is requested, the scheduler creates a new entry for that process in the process table
  - Entry contains
    - The memory area assigned
    - The priority of the process
    - Whether the process is ready or waiting

- Dispatcher oversees the execution of the processes
  - Divide time into short segments (time slices)
  - Process switch (context switch)
  - Generate interrupt to indicate the end of a slice

Process Administration

- Interrupt Handler
  - Stored at a predetermined location in main memory
  - When CPU receives an interrupt signal
    - Completes current machine cycle
    - Saves its position in the current process
    - Begins executing a program
- The effect of the interrupt signal is to preempt the current process and transfer control back to the dispatcher
- At this point, the dispatcher selects a ready process from the process table (as determined by the scheduler), restarts the timer circuit, and allows the selected process to begin its time slice

Interrupt-Driven I/O Cycle


CPU
I/O controller
1
device driver initiates I/O
2
initiates I/O
CPU executing checks for interrupts between instructions
3
CPU receiving interrupt, transfers control to interrupt handler
4
input ready, output complete, or error generates interrupt signal
7
5
interrupt handler processes data, returns from interrupt
6
CPU resumes processing of interrupted task

Process Administration

- Re-create the environment
  - Value of the program counter
  - Content of the registers and pertinent memory cells
  - CPUs designed for multiprogramming systems
    - Incorporate the task of saving this information as part of the CPU’s reaction to the interrupt signal
    - Have machine-language instructions for reloading a previously saved state

- Multiprogramming
  - Increase the overall efficiency of a machine

Process Switch

- Multiprogramming between process A and process B

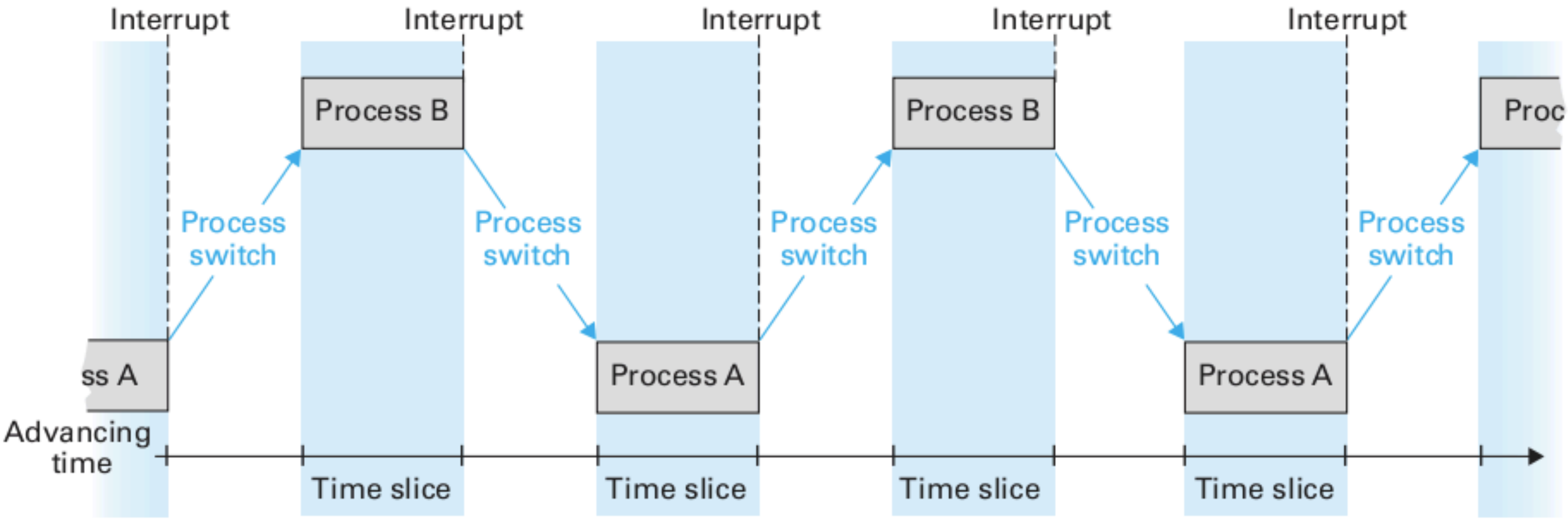
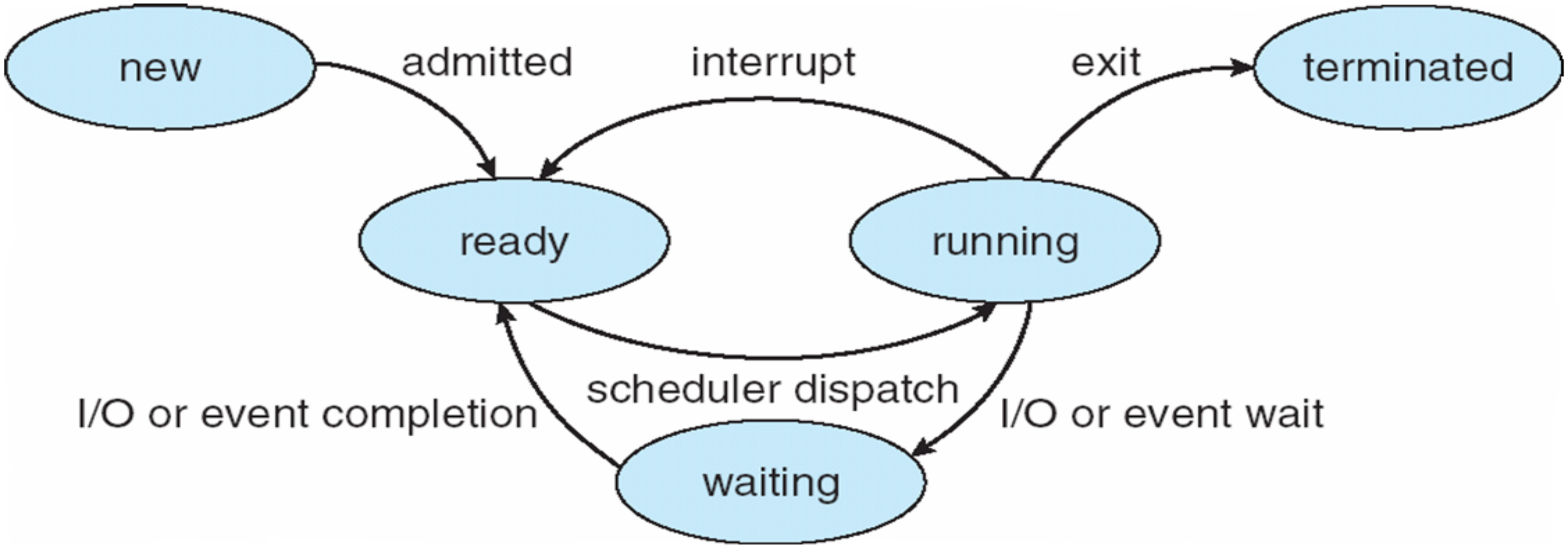


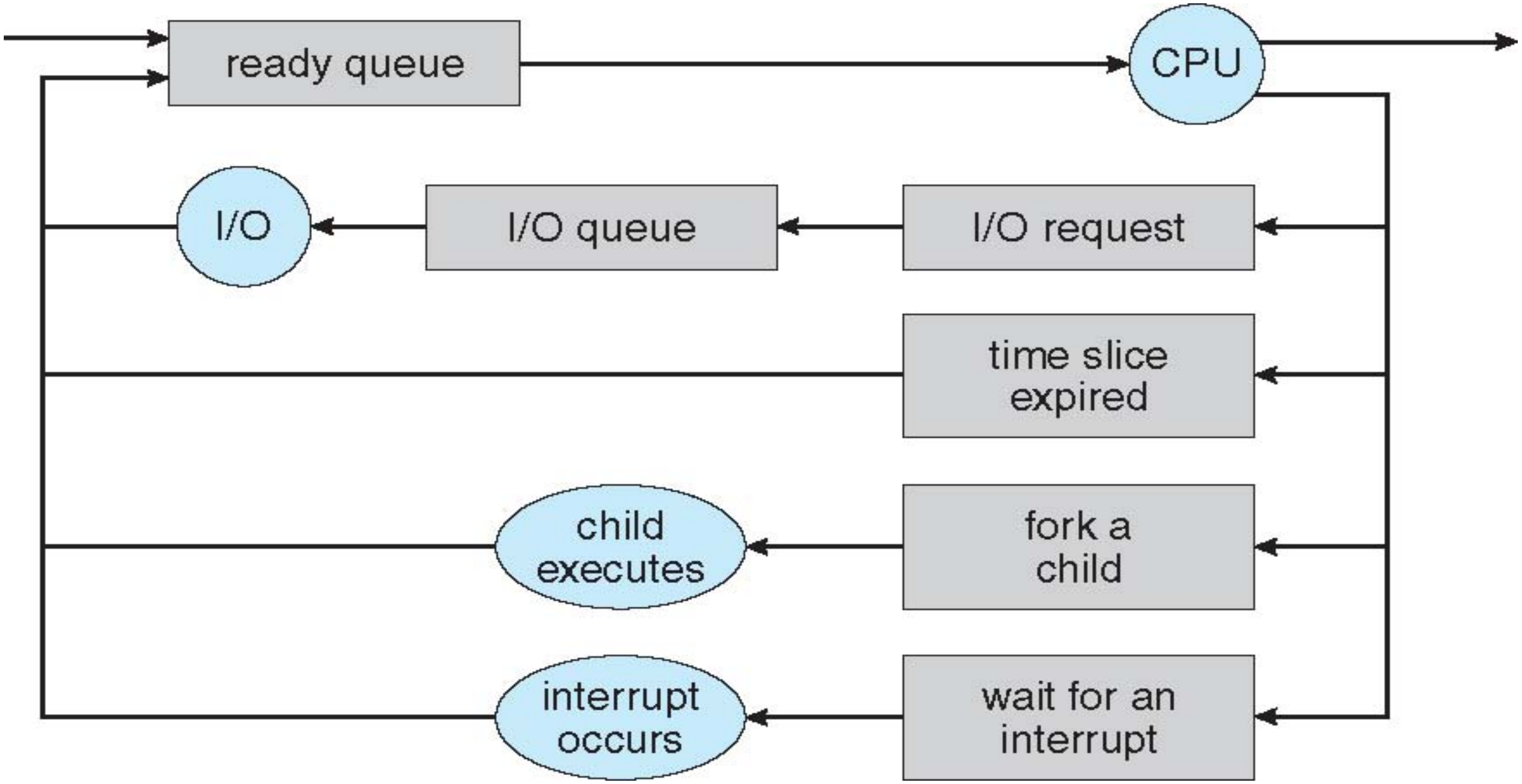
Diagram of Process State



As a process executes, it changes *state*

new: The process is being created
running: Instructions are being executed
waiting: The process is waiting for some event to occur
ready: The process is waiting to be assigned to a processor
terminated: The process has finished execution

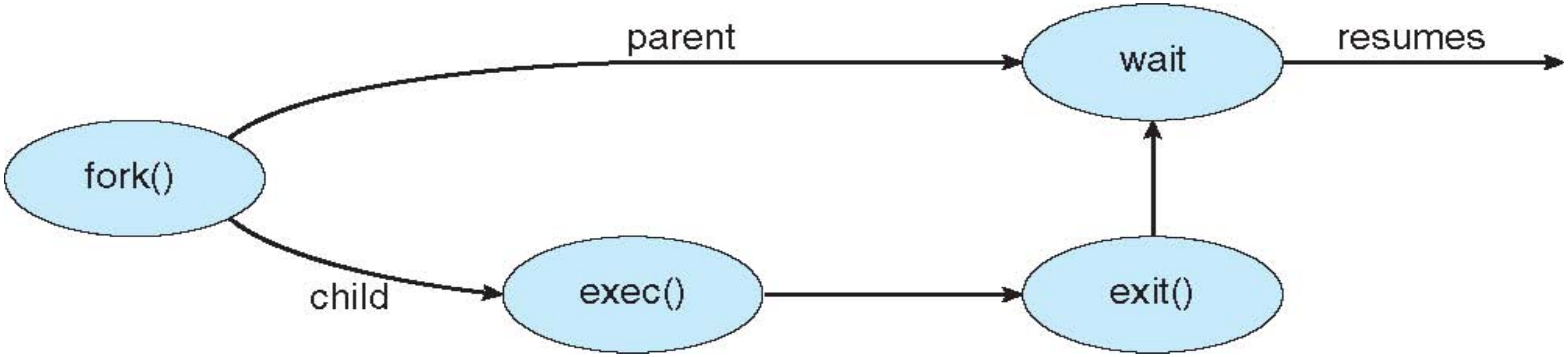
Representation of Process Scheduling

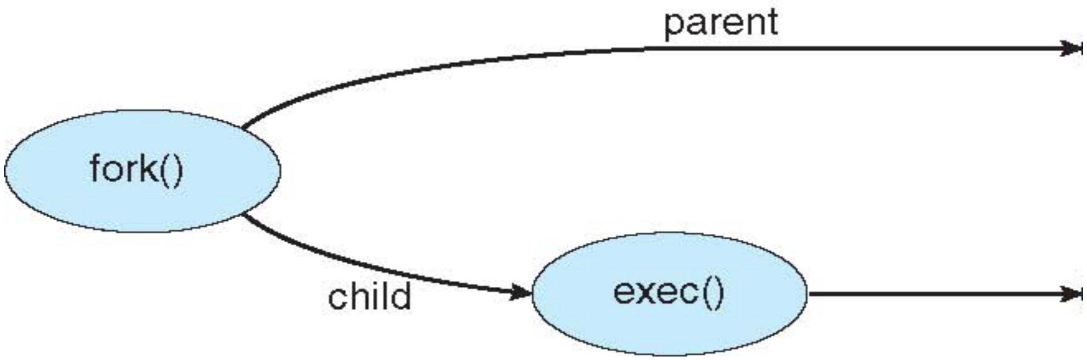


Process Creation

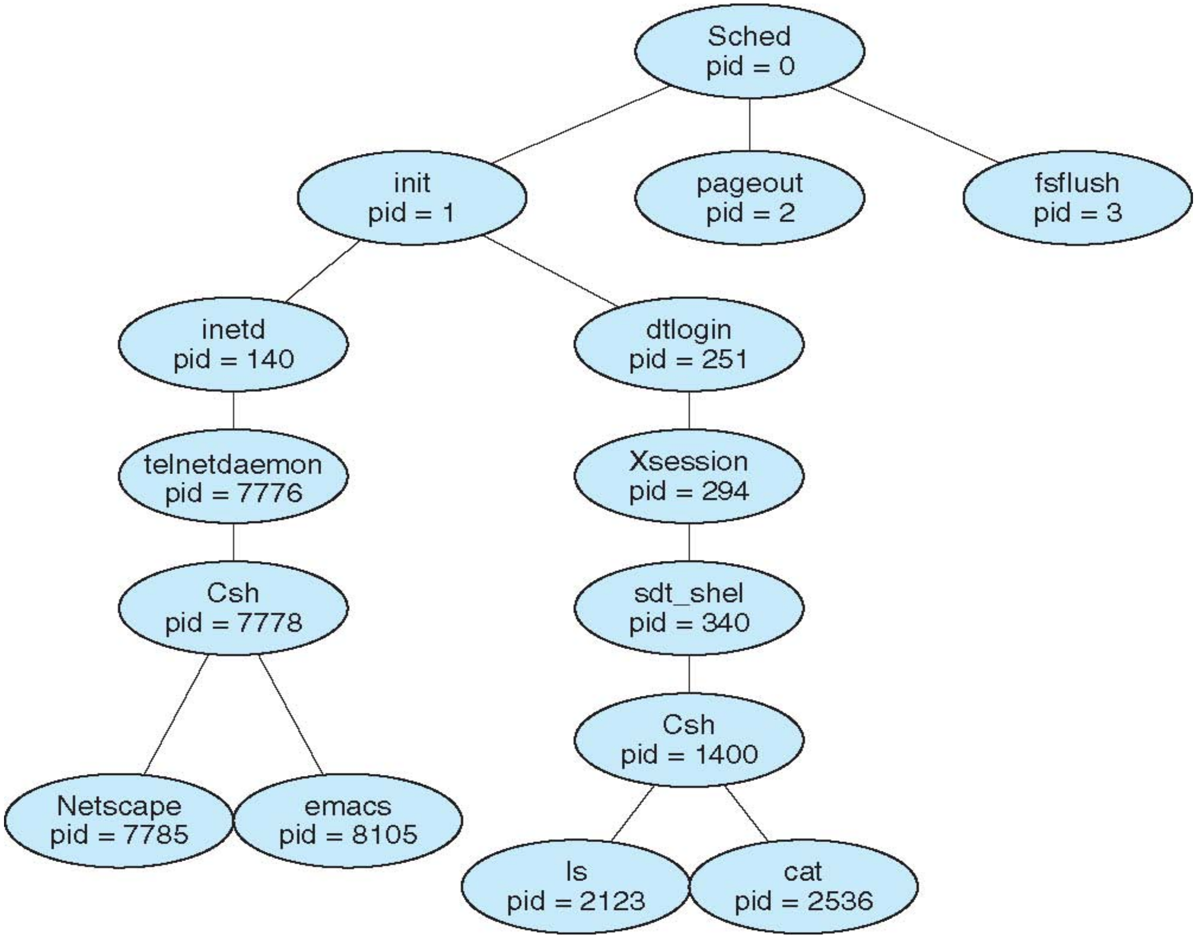
- **Parent** process create **children** processes, which, in turn create other processes, forming a tree of processes
- Generally, process identified and managed via **a process identifier** (**pid**)
- Execution
  - Parent and children execute concurrently
  - Parent waits until children terminate
- UNIX examples
  - **fork** system call creates new process
  - **exec** system call used after a **fork** to replace the process' memory space with a new program

Process Creation

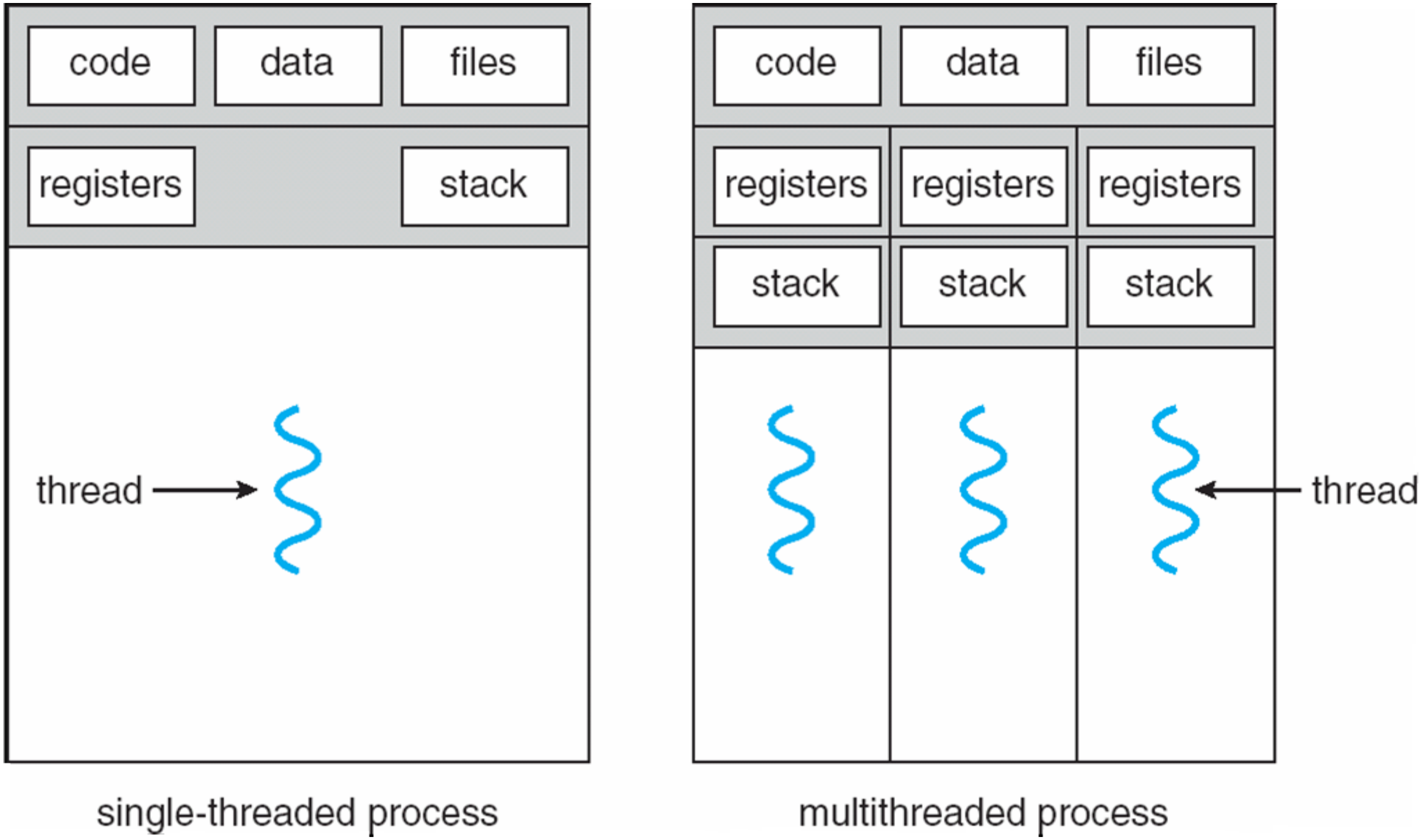




A Sample Tree of Processes



Single-threaded & Multithreaded Process



CPU Scheduling

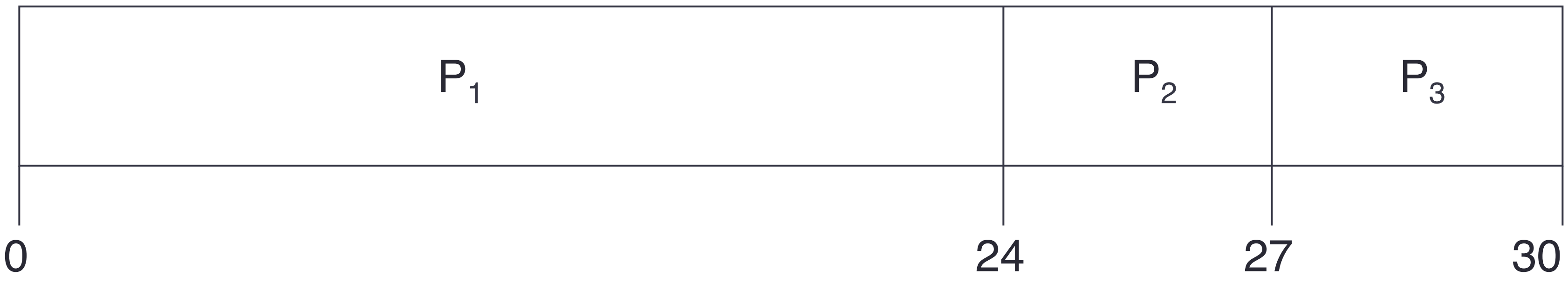
- Objectives
  - To introduce CPU scheduling, which is the basis for multiprogrammed operating systems
  - To describe various CPU-scheduling algorithms
  - To discuss evaluation criteria for selecting a CPU-scheduling algorithm for a particular system

- Criteria
  - **CPU utilization** – keep the CPU as busy as possible
  - **Throughput** – # of processes that complete their execution per time unit
  - **Turnaround time** – amount of time to execute a particular process
  - **Waiting time** – amount of time a process has been waiting in the ready queue
  - **Response time** – amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

First Come First Serve (FCFS)

Process	Burst Time
P_1	24
P_2	3
P_3	3

- Suppose that the processes arrive in the order: P_1 , P_2 , P_3
  The Gantt Chart for the schedule is:

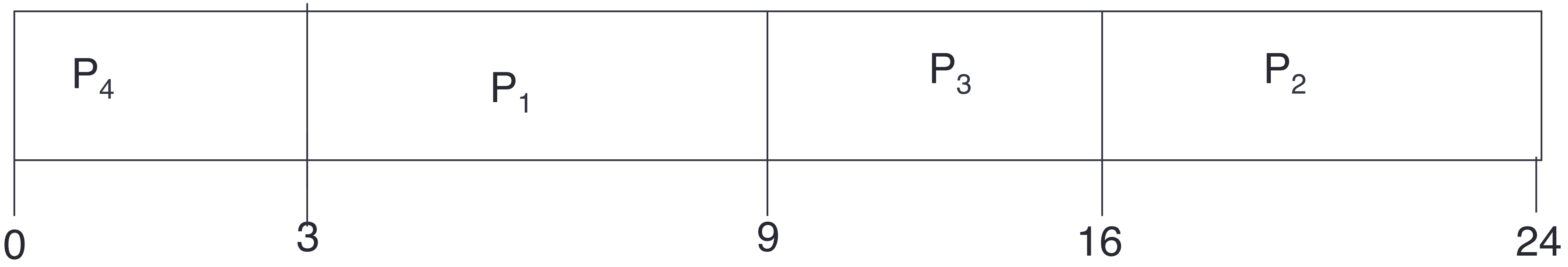


- Waiting time for $P_1 = 0$; $P_2 = 24$; $P_3 = 27$
- Average waiting time: (0 + 24 + 27)/3 = 17

Shortest Job First (SJF)

Process	Burst Time
P_1	6
P_2	8
P_3	7
P_4	3

- SJF scheduling chart

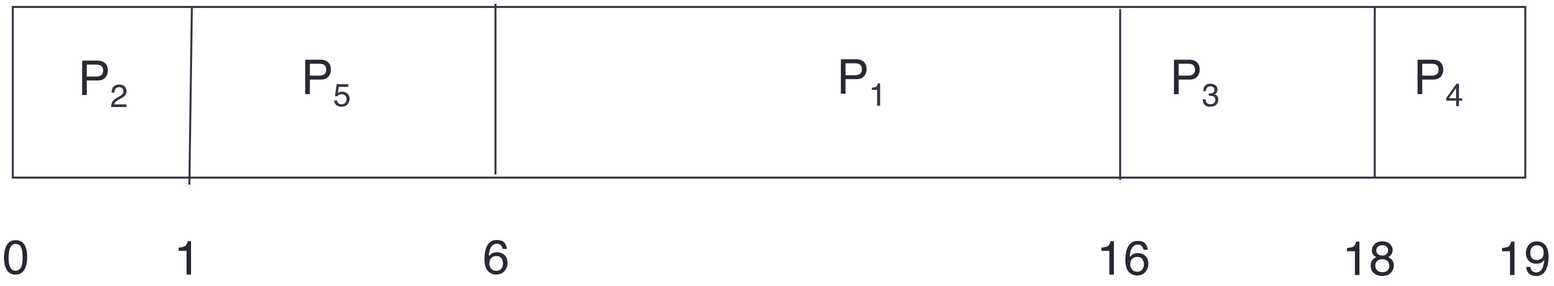


- Average waiting time = (3 + 16 + 9 + 0) / 4 = 7

Priority Scheduling

Process	Burst Time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

- Priority scheduling Gantt Chart



P_2 P_5 P_1 P_3 P_4

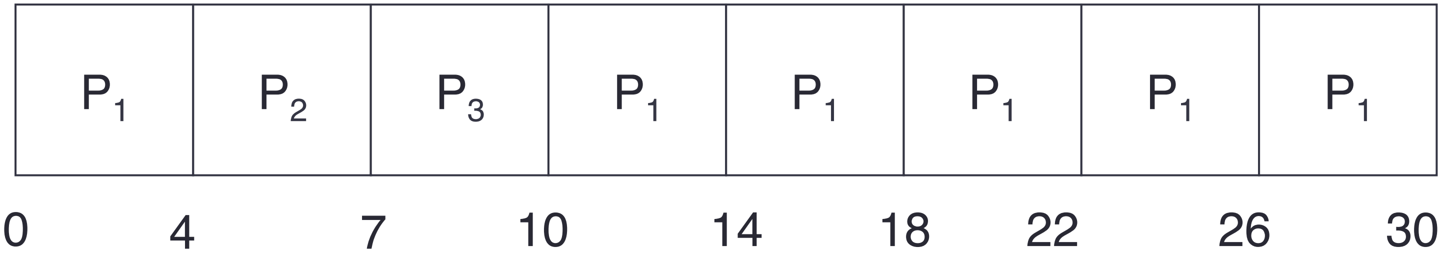
0 1 6 16 18 19

- Average waiting time = 8.2 msec

Round Robin (RR)

Process	Burst Time
P_1	24
P_2	3
P_3	3

- The Gantt chart (quantum=4) is:



P_1 P_2 P_3 P_1 P_1 P_1 P_1 P_1

0 4 7 10 14 18 22 26 30

- Typically, higher average turnaround than SJF, but better *response*
- q should be large compared to context switch time
- q usually 10ms to 100ms, context switch < 10 usec

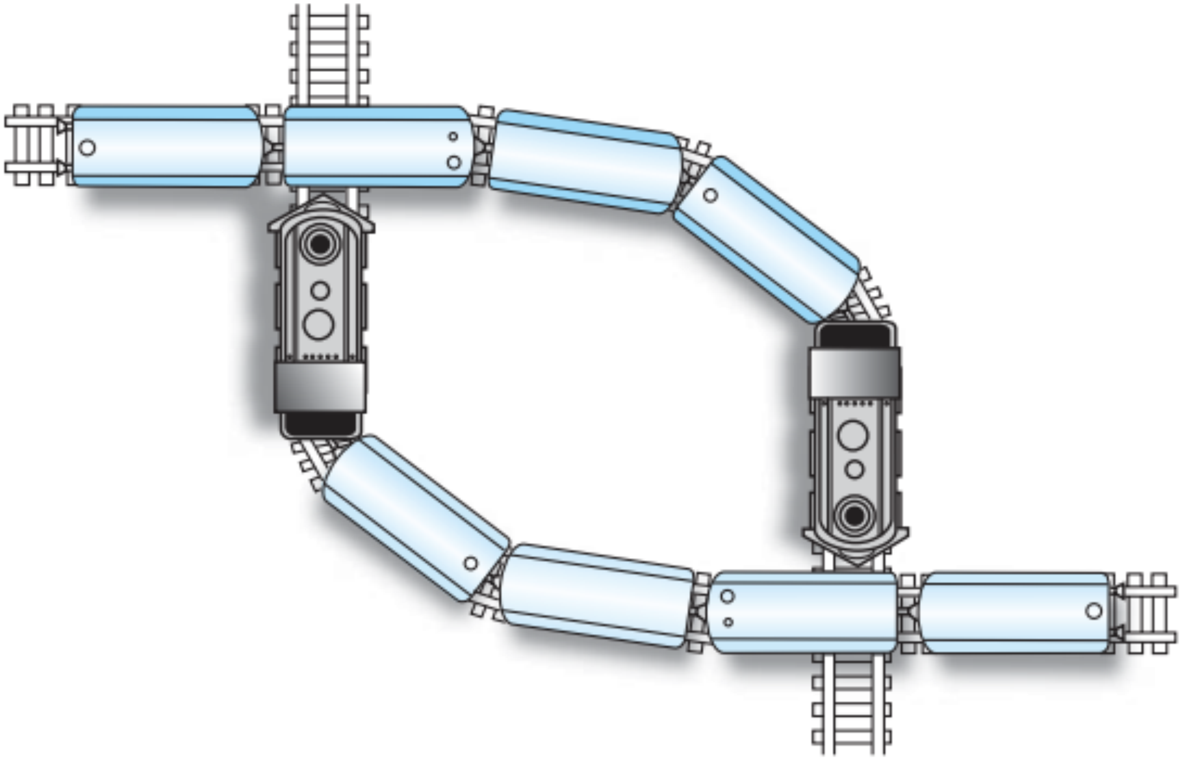
Handling Competition Among Processes

- The Critical-Section Problem
  - Consider a system consisting of n processes. Each process has a segment of code, called a **critical section**, in which the process may be changing common variables, updating a table, writing a file, and so on.
  - Printer allocation example

- Solution
  - Interrupt disable and interrupt enable
  - **Semaphore**, for the test-and-set instruction – a single instruction

Handling Competition Among Processes

- Deadlock
  - A condition in which two or more processes are blocked from progressing because each is waiting for a resource that is allocated to another



Handling Competition Among Processes

- Deadlock cannot occur unless all three of the following conditions are satisfied
  1. There is competition for nonshareable resources
  2. The resources are requested on a partial basis; that is, having received some resources, a process will return later to request more
  3. Once a resource has been allocated, it cannot be forcibly retrieved

- Solution
  - Attacking #3 – deadlock detection and correction: forcibly retrieving resources, e.g. kill
  - Attacking #1 & #2 – deadlock avoidance: converting nonshareable resources into shareable ones (e.g. spooling); request all resources at one time

Today's Topics

- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
    - Kernel and User Space
    - Processes
        - Process Administration
        - Coordinating the Machine's Activities
        - Handling Competition Among Processes
    - Memory
    - File System
- Security

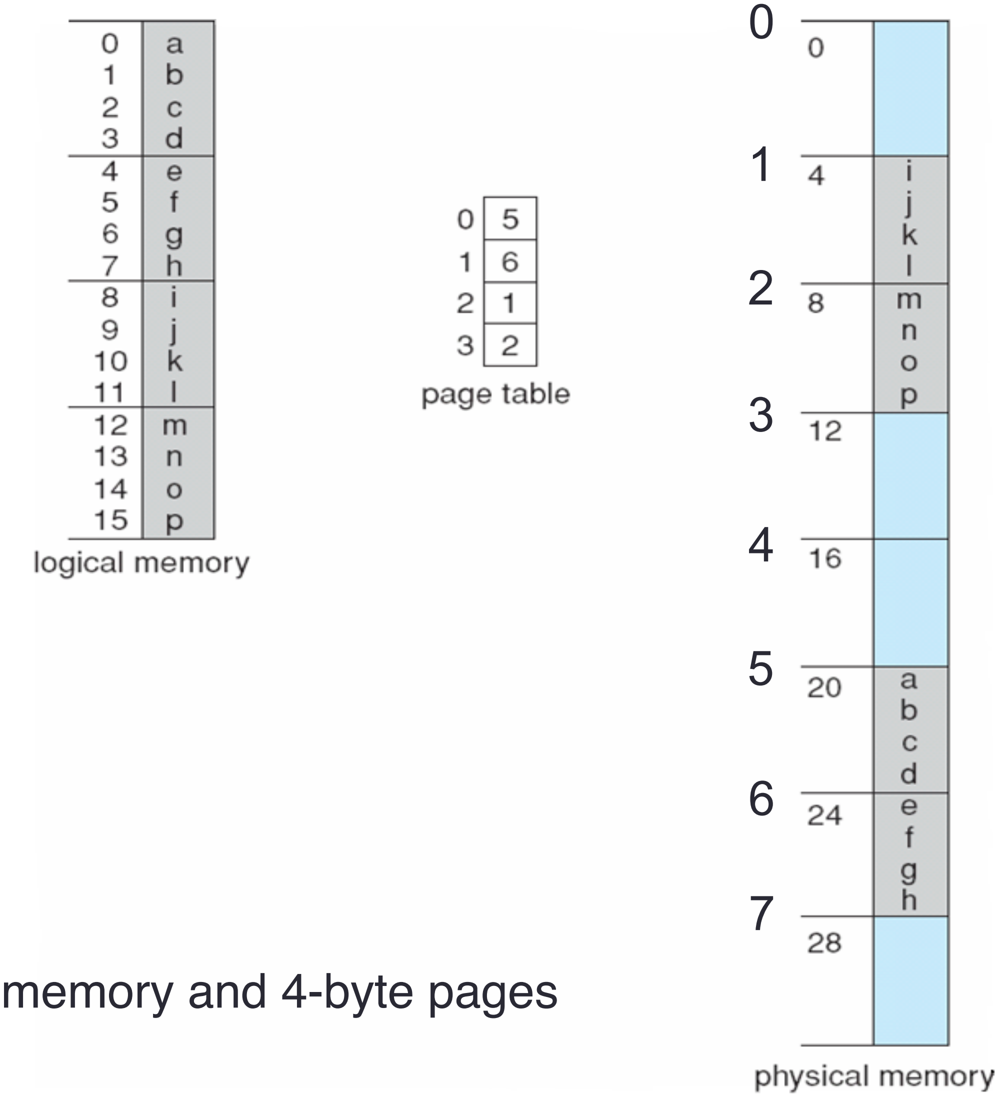
Memory

- Address
  - **Physical address** – address seen by the memory unit
  - **Logical address** – generated by the CPU; also referred to as **virtual address**

- Memory
  - **Physical memory** – memory of the machine
  - **Virtual memory** – Logical address space that can be much larger than physical address space

- Paging
  - A technique that the memory manager create the illusion of additional memory space by rotating programs and data back and forth between main memory and mass storage

Paging Example



0	a
1	b
2	c
3	d
4	e
5	f
6	g
7	h
8	i
9	j
10	k
11	l
12	m
13	n
14	o
15	p

logical memory

0	5
1	6
2	1
3	2

page table

0	0	
1	4	i j k l
2	8	m n o p
3	12	
4	16	
5	20	a b c d
6	24	e f g h
7	28	

physical memory

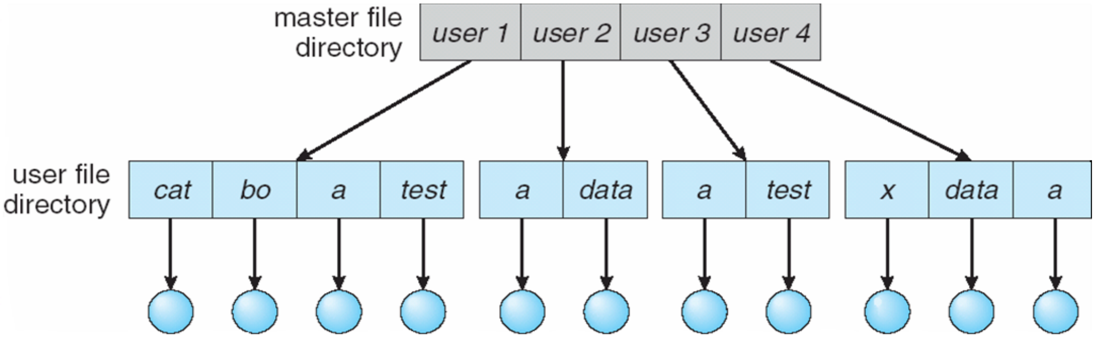
n=2 and *m*=4   32-byte memory and 4-byte pages

Today’s Topics

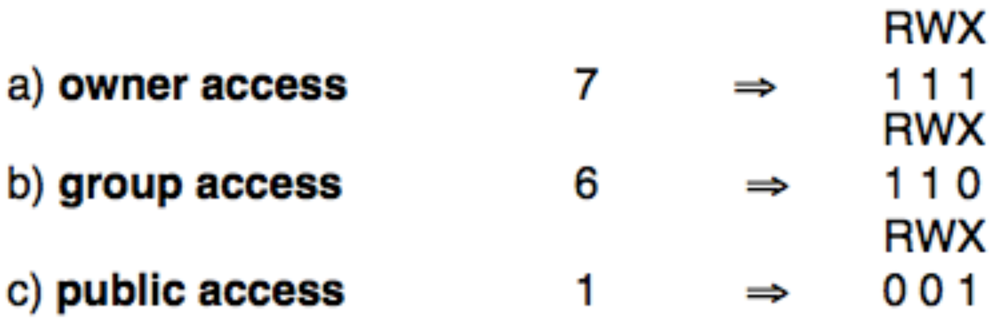
- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
    - Kernel and User Space
    - Processes
        - Process Administration
        - Coordinating the Machine’s Activities
        - Handling Competition Among Processes
    - Memory
    - File System
- Security

File System

- Filename, path, directory
- Directory structure (tree structure and general graph structure)



- File System Mounting
- Access Control
  - Mode of access: read, write, execute
  - Three classes of users



			RWX
a) **owner access**	7	⇒	1 1 1
			RWX
b) **group access**	6	⇒	1 1 0
			RWX
c) **public access**	1	⇒	0 0 1

Today’s Topics

- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
  - Kernel and User Space
  - Processes
    - Process Administration
    - Coordinating the Machine’s Activities
    - Handling Competition Among Processes
  - Memory
  - File System
- Security

Security

- To Protect against the Attacks from the Outside
  - Login
  - Super user
  - Detect destructive behavior – auditing software
  - Detect the presence of sniffing software
  - Prevention
    - Be careful
    - Change password
    - Adopt and enforce policies

- To Protect against the Attacks from Within
  - Privilege levels: privileged mode and nonprivileged mode
  - privileged instructions: instructions only available in privileged mode

Today’s Topics

- The History of Operating Systems
- Getting Operating System Started
- Aspects of an Operating System
  - Kernel and User Space
  - Processes
    - Process Administration
    - Coordinating the Machine’s Activities
    - Handling Competition Among Processes
  - Memory
  - File System
- Security

For the Next Class

- Survey, if you haven't done it (use one or two sentences)
  1. Tell me about you: name, school, major, year
  2. Why do you want to take this course?
  3. What do you expect to learn?
  4. How do you think this course can be relevant to your current major, future study, or career?
  5. Do you have any existing knowledge in computer science, e.g. programming language, web design, database, etc?
- Email to xie@cs.columbia.edu

- Read Chapter 4 HTML by Snyder
- Read Chapter 3 Networking and the Internet by Brookshear
- How to set up a personal website using CUNIX: http://cuit.columbia.edu/web-publishing/creating-personal-websites
- CUNIX tutorial: http://www.columbia.edu/~lgw23/cs1004/

References & Photo Credits

- Brookshear, J. Glenn (2011-04-13). Computer Science: An Overview (11th Edition). Prentice Hall. Kindle Edition.
- Avi Silberschatz, Peter Baer Galvin, Greg Gagne. Operating System Concepts Essentials. Slides from: http://os-book.com
- Avi Silberschatz, Peter Baer Galvin, Greg Gagne. Operating System Concepts. Slides from: http://os-book.com