Operating Systems II

Basic Review

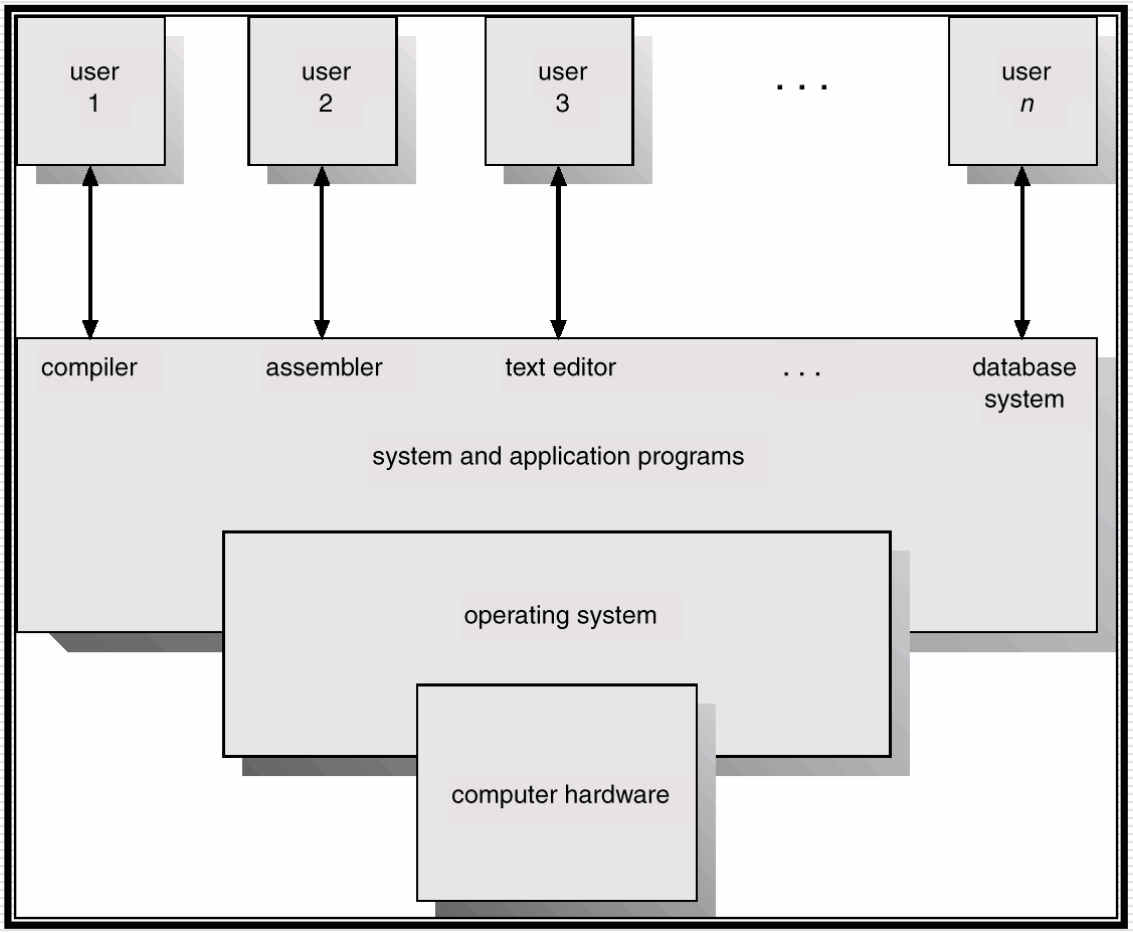
What is an Operating System?

- A program that acts as an intermediary between a user of a computer and the computer hardware.
- Operating system goals:
  - Execute user programs and make solving user problems easier.
  - Make the computer system convenient and easy to use.
- Use the computer hardware in an efficient manner

Computer System Components

- Hardware – provides basic computing resources (CPU, memory, I/O devices).
- Operating system – controls and coordinates the use of the hardware among the various application programs for the various users.
- Applications programs – define the ways in which the system resources are used to solve the computing problems of the users (compilers, database systems, video games, business programs).
- Users (people, machines, other computers).

Abstract view of System



Operating System Definitions

- Resource allocation – manages and allocates resources.
- Control program – controls the execution of user programs and operations of I/O devices.
- Kernel – the one program running at all times (all else being application programs).

Common System Components

- Process Management
- Main Memory Management
- File Management
- I/O System Management
- Secondary Management
- Networking
- Protection System
- Command-Interpreter System

Process Management

- A *process* is a program in execution. A process needs certain resources, including CPU time, memory, files, and I/O devices, to accomplish its task.
- The operating system is responsible for the following activities in connection with process management.
  - Process creation and deletion.
  - process suspension and resumption.
  - Provision of mechanisms for:
    - process synchronization
    - process communication

Processes

- What are the different process states?
  - new, running, waiting, ready, terminated.
- What is a Process Control Block (PCB)?
  - State, PC, registers, scheduling info, memory-management info, accounting info, I/O status info

Threads

- What is the difference between a process and a thread?
  - Threads are lightweight (just have a thread id, PC, registers, and a stack), processes are heavy!
- Why use threads in an application?
  - Responsiveness, resource sharing, quicker to create than processes, utilization of multiprocessor architectures.

CPU Scheduling

- Why schedule processes?
- What is the difference between preemptive and nonpreemptive scheduling?
  - Preemptive scheduling occurs at any time, nonpreemptive scheduling only happens when a process stops executing on its own!

Scheduling 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)

Scheduling Algorithms

- First Come First Serve (FCFS)
  - advantage: simple, disadvantage: at mercy of process arrival times...
- Shortest Job First (SJF)
  - advantage: really good for short jobs, disadvantage: bad for long jobs, How do you determine what jobs are short?
- Priority Scheduling
  - problem of starvation, solved by using aging mechanism...

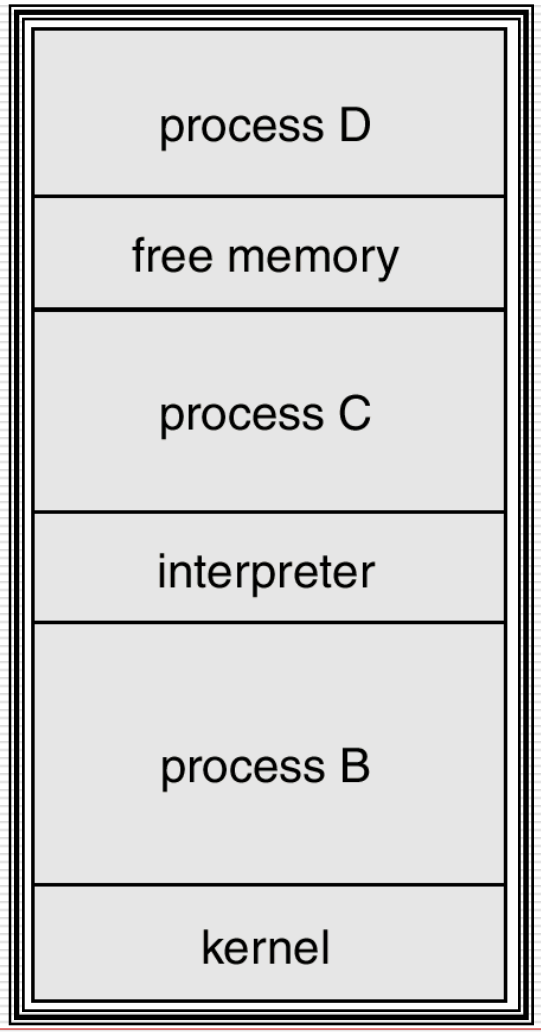
Scheduling Algorithms (cont)

- Round Robin
  - Higher turnaround time, but quicker response time...
- Multilevel Queue
  - Different queues with different scheduling algorithms...
- Multilevel Feedback Queue
  - Processes move between multiple queues based on process characteristics (eg aging)...

Main-Memory Management

- Memory is a large array of words or bytes, each with its own address. A repository of quickly accessible data shared by the CPU and I/O devices.
- Main memory is a volatile storage device. It loses its contents with power down.
- The OS is responsible for the following activities in connection with memory:
  - Keep track of which parts of memory are currently being used and by whom.
  - Decide which processes to load when memory space becomes available.
  - Allocate and deallocate memory space as needed.

Generic Memory Breakdown



Memory Management

- Swapping - allows use of more memory than the system has!
- Paging algorithms
  - Least Recently Used (LRU)
  - First In First Out (FIFO)

File Management

- A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data.
- The operating system is responsible for the following activities in connections with file management:
  - File creation and deletion.
  - Directory creation and deletion.
  - Support of primitives for manipulating files and directories.
  - Mapping files onto secondary storage.
  - File backup on stable (nonvolatile) storage media.

File System Implementation

- Partition Control Block
  - number of blocks in partition, size of the blocks, free file control block count, free file control block pointer
- File Control Blocks
  - type, ownership, permissions, dates, size, file blocks
- Different algorithms for managing used and free blocks.

I/O System Management

- A wide variety of I/O devices exist.
  - Hard drive
  - Microphone
- I/O devices can typically be classed into block and character devices.
- The I/O system consists of:
  - A buffer-caching system
  - A general device-driver interface
  - Drivers for specific hardware devices

Secondary Storage Management

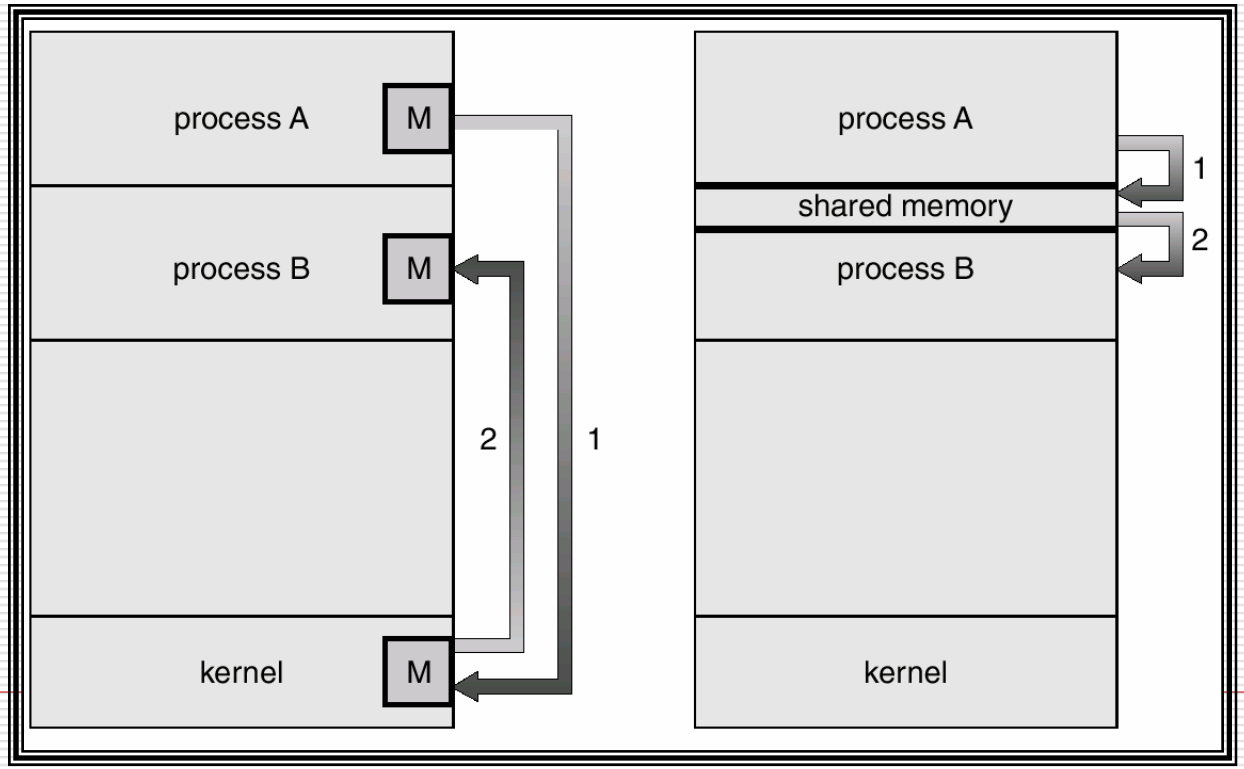
- Since main memory (*primary storage*) is small and volatile in many systems, often the computer system provides *secondary storage* to back up main memory.
- Most modern computer systems use disks as the principle on-line storage medium, for both programs and data.
- The operating system is responsible for the following activities in connection with disk management:
  - Free space management
  - Storage allocation
  - Disk scheduling

Networking / Communication

- A processor often requires communication with other processors (possibly on the same machine).
- Processors are connected through a communication network.
- Communication takes place using a *protocol.*
- This network can provide user access to various system or user resources for:
  - Computation speed-up
  - Increased data availability
  - Enhanced reliability
- Must be timely and capable of supporting variable delays.

Communication Models

- Communication may take place using either message passing or shared memory



Synchronization

- Why is synchronization needed?
  - Race conditions while sharing resources!
- What are the 3 requirements that a solution to the critical section problem must have?
  - Bounded waiting, mutual exclusion, and progress

Synchronization Mechanisms

- Hardware *Test&Set* instruction.
- Semaphores
  - signal & wait
- Monitors
  - synchronized procedures that ensure mutual exclusive access to shared data.

Protection System

- *Protection* refers to a mechanism for controlling access by programs, processes, or users to both system and user resources.
- The protection mechanism must:
  - distinguish between authorized and unauthorized usage.
  - specify the controls to be imposed.
  - provide a means of enforcement.
- Ideally will not infringe on valid sharing of data or resources.

Security

- Protect the system from:
  - unauthorized access
  - malicious modification or destruction
  - accidental misuse.
- Security features
  - authentication
  - Firewall
  - Intrusion Detection
  - ...

Command-Interpreter (Shell)

- Many commands are given to the operating system by control statements which deal with:
  - Process creation and management
  - I/O handling
  - Secondary-storage management
  - Main-memory management
  - File-system access
  - Protection
  - Networking
- Its function is to get and execute the next command line statement.

System Calls

- System calls provide the interface between a running program and the operating system.
  - Generally available as assembly-language instructions.
  - Languages defined to replace assembly language for systems programming allow system calls to be made directly (e.g., C, C++)

System Calls (Cont.)

- Three general methods are used to pass parameters between a running program and the operating system.
  - Pass parameters in *registers*.
  - Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
  - *Push* (store) the parameters onto the *stack* by the program, and *pop* off the stack by operating system.

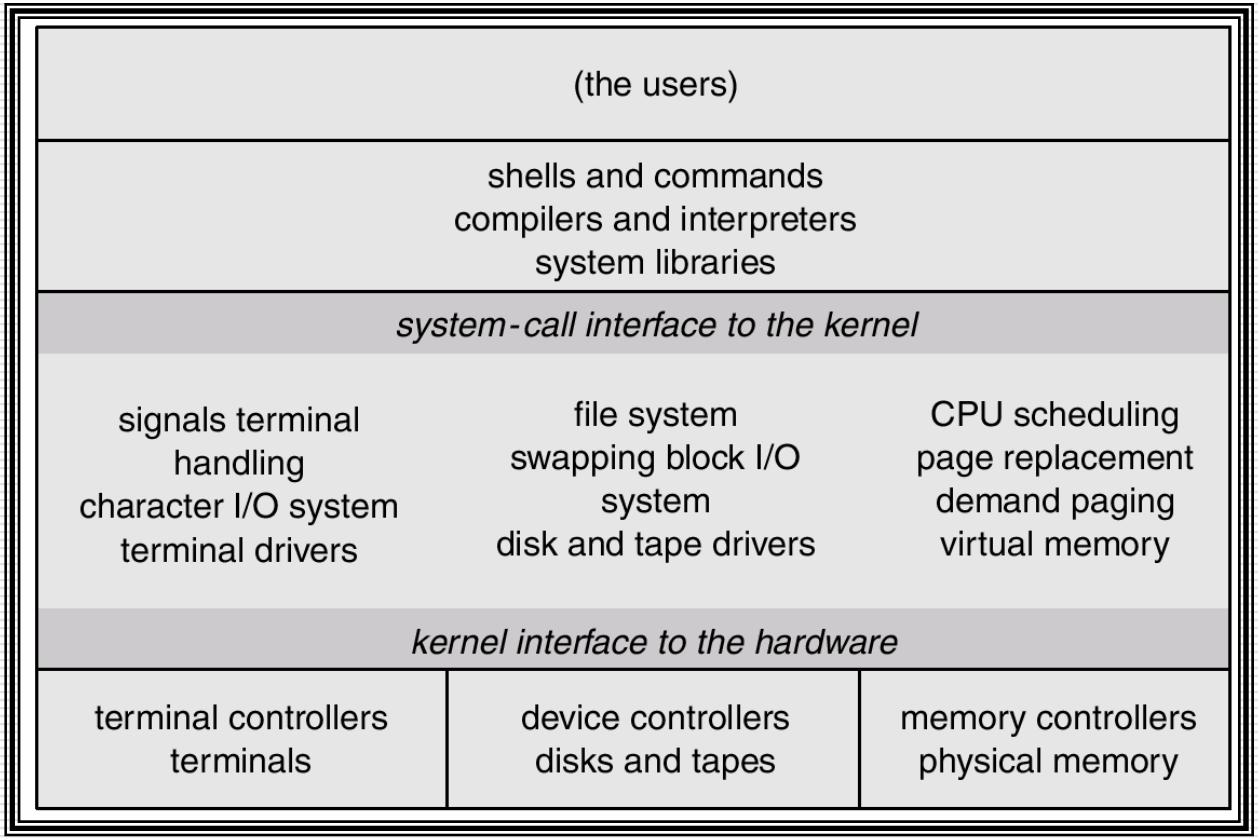
System Programs

- System programs provide a convenient environment for program development and execution. The can be divided into:
  - File manipulation
  - Status information
  - File modification
  - Programming language support
  - Program loading and execution
  - Communications
  - Application programs
- Most users' view of the operation system is defined by system programs, not the actual system calls.

UNIX System Structure

- The UNIX OS consists of two separable parts.
  - Systems programs
  - The kernel
    - Consists of everything below the system-call interface and above the physical hardware
    - Provides the file system, CPU scheduling, memory management, and other operating-system functions; a large number of functions for one level.

UNIX System Structure



System Design Goals

- User goals – operating system should be convenient to use, easy to learn, reliable, safe, and fast.
- System goals – operating system should be easy to design, implement, and maintain, as well as flexible, reliable, error-free, and efficient.