

Operating Systems(CS301)

Course Overview

- **Overview**
- Operating Systems (OS) Objectives, Formal Definition, Evolution, Types,DMA & Multiprogramming - OS Interfaces - The Command-less command interpreter systems - Device drivers - Examples
- **Processes and Threads**
- Process Management: The process concept - Programs, Processes & Threads – Process Control Block – PCB as a data structure in contemporary operating systems - Process Hierarchy - System Calls - CPU Scheduling & algorithms. Metrics - Examples
- **Concurrency: Mutual Exclusion and Synchronization**
- Concurrent processes - The Critical Section & Mutual Exclusion problem - Algorithms - Semaphores, Critical Region, Conditional Critical Region, Monitors, Messages - Examples in Contemporary OS - Classical Process Coordination Problems. Deadlocks: Characterization - Prevention - Avoidance - Detection - Recovery - Combined Approach to Deadlock handling & Deadlock Handling in contemporary OS.
- **Concurrency: Deadlock and Starvation**
- Principles of Concurrency, Mutual Exclusion, Semaphores, Monitors, Message Passing, Readers/Writers Problem
- **Scheduling**
- Uniprocessor Scheduling: Long Term Scheduling, Medium Term Scheduling, Short Term Scheduling, Scheduling Algorithms: Short Term Scheduling Criteria, Use of Priorities, Alternative Scheduling Policies, Performance Comparison, Fair-Share Scheduling. Multiprocessor Scheduling:Granularity, Design Issue, Process Scheduling, Thread Scheduling, Real-Time Scheduling: Characteristics of RTOS, Real-Time Scheduling, Deadline Scheduling, Rate Monotonic Scheduling, Priority Inversion. Case Study: Linux & Windows Scheduling.

- **Memory Management**
- Memory Hierarchy, Static and Dynamic Memory Allocation, Overview of Swapping, Multiple Partitions Contiguous and Non-Contiguous Memory Allocation, Concepts of Paging, Segmentation.
- **Virtual Memory**
- Virtual Memory Concepts - Demand paging - Performance - Fragmentation & Compaction. Page replacement and Allocation algorithms -Memory Protection - System Calls – Linux/Windows Virtual Memory Techniques.
- **I/O management and disk scheduling**
- Terminals & Capability Databases - Emulators - Virtual Terminals - Disk Devices - Device Independence - Free space management - Performance and Reliability - Storage hierarchy.
- **File Management**
- Levels - File Systems in Disk Partitions - File-naming & File Access - Allocation strategies - Directory systems & their implementations - File Systems to device drivers - File Systems Reliability – Examples of fsck() and fsdb() utilities - File protection - Implementation issues

BOOKS RECOMMENDED

- 1)Silberschatz A & Galvin: “Operating System Concepts”, John Wiley & Sons
- 2)W. Stallings: “Operating Systems: Internals and Design Principles”, Pearson Pub.
- 3)Andrew S Tanenbaum & A Woodhul;: “Operating Systems - Design and Implementation”, PHI EEE
- 4)Crawley: “Operating Systems - An Object oriented Approach”, McGraw Hill
- 5)Kernighan & Pike: “UNIX programming Environment”, PHI-EEE

Overview

What is an Operating System?

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

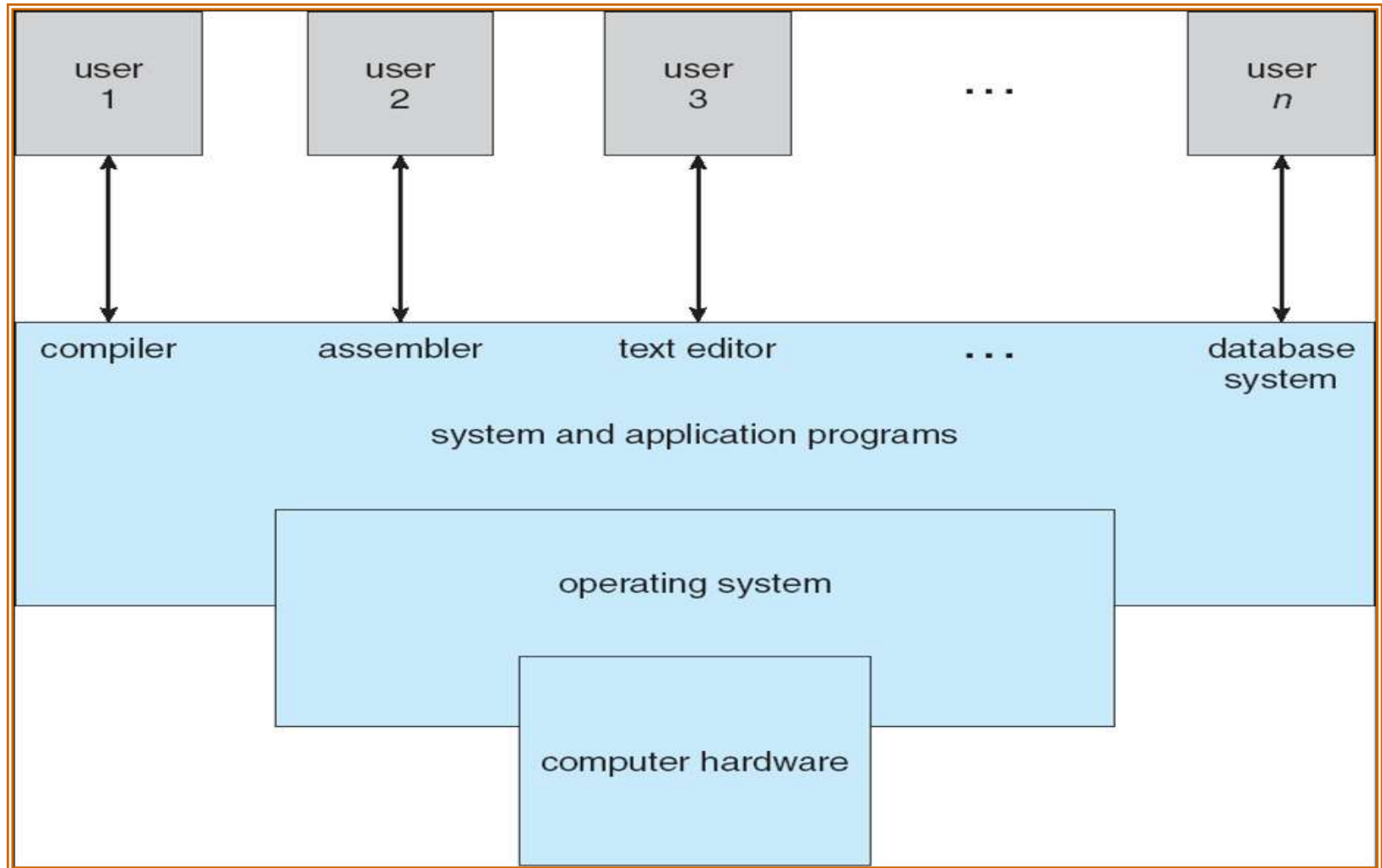
Operating System Objectives

- Convenience
- Efficiency
- Ability to evolve

Computer System Structure

- Computer system can be divided into four components
 - Hardware – provides basic computing resources
 - CPU, memory, I/O devices
 - Operating system
 - Controls and coordinates use of hardware among various applications and users
 - Application programs – define the ways in which the system resources are used to solve the computing problems of the users
 - Word processors, compilers, web browsers, database systems, video games
 - Users
 - People, machines, other computers

Four Components of a Computer System



Operating System Definition

- OS is a **resource allocator**
 - Manages all resources
 - Decides between conflicting requests for efficient and fair resource use
- OS is a **control program**
 - Controls execution of programs to prevent errors and improper use of the computer
- No universally accepted definition
- “Everything a vendor ships when you order an operating system” is good approximation
 - But varies wildly

Kernel

- “The one program running at all times on the computer” is the **kernel**. Everything else is either a system program (ships with the operating system) or an application program
- Portion of operating system that is in main memory
- Contains most frequently used functions
- Also called the nucleus

Evolution of Operating Systems

- Hardware upgrades plus new types of hardware
- New services
- Fixes

Evolution of Operating Systems

(1) Serial processing

- No operating system
- Machines run from a console with display lights, toggle switches, input device, and printer
- Schedule time
- Setup included loading the compiler, source program, saving compiled program, and loading and linking

Evolution of Operating Systems

(2) Simple batch system

— Monitor

- Software that controls the sequence of events
- Batch jobs together
- Program returns control to monitor when finished

Job Control Language

- Special type of programming language
- Provides instruction to the monitor
 - What compiler to use
 - What data to use

Hardware Features

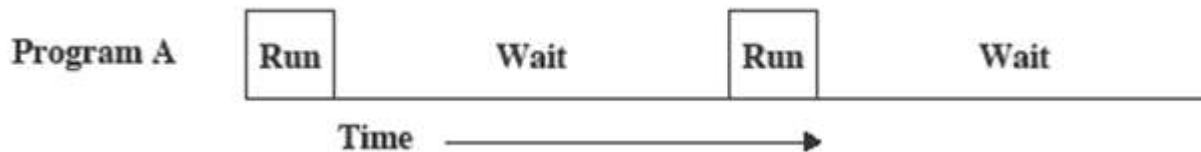
- Memory protection
 - Does not allow the memory area containing the monitor to be altered
- Timer
 - Prevents a job from monopolizing the system
- Privileged instructions
 - Certain machine level instructions can only be executed by the monitor
- Interrupts
 - Early computer models did not have this capability

Memory Protection

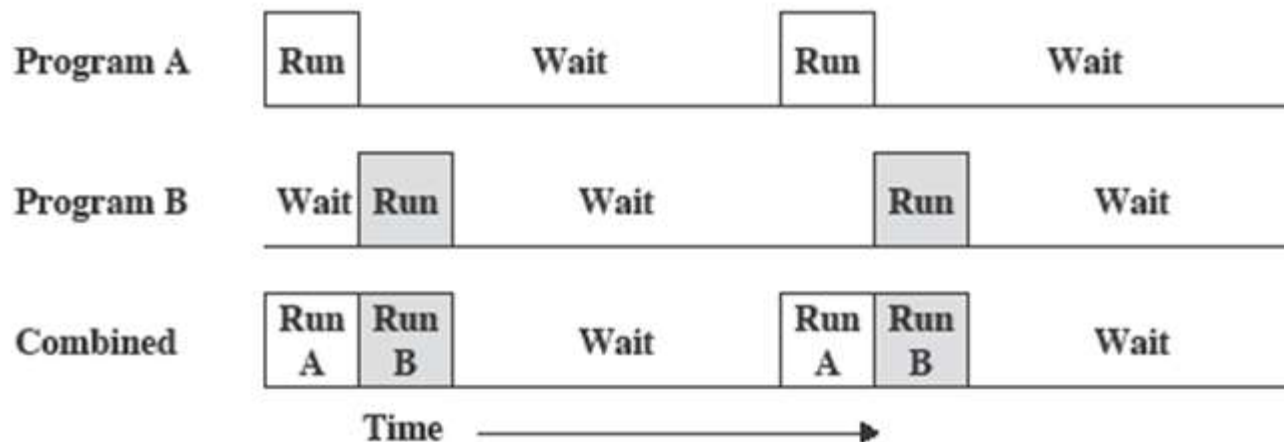
- User program executes in user mode
 - Certain instructions may not be executed
- Monitor executes in system mode
 - Kernel mode
 - Privileged instructions are executed
 - Protected areas of memory may be accessed

(3) Multiprogrammed Batch systems

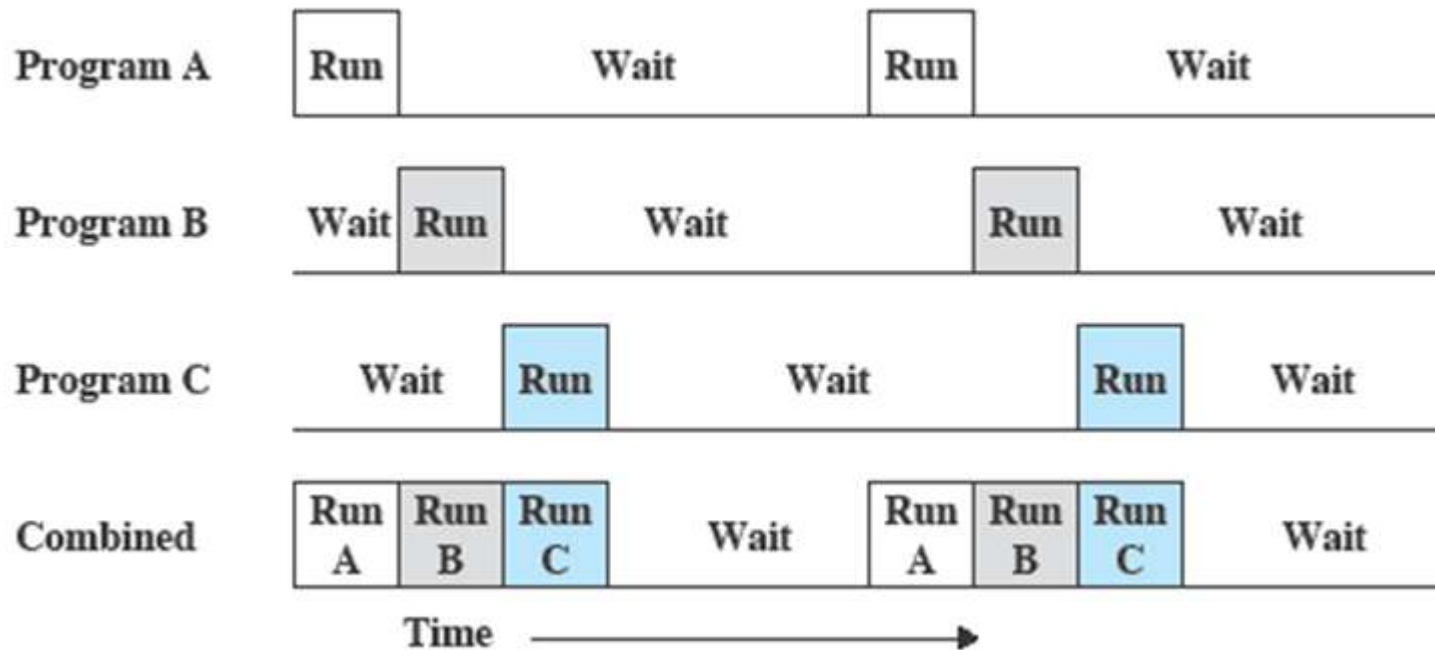
- Uniprogramming -
- Processor must wait for I/O instruction to complete before proceeding



- Multiprogramming with two programs
- When one job needs to wait for I/O, the processor can switch to the other job



- Multiprogramming with three programs



(4) Time Sharing Systems

- Using multiprogramming to handle multiple interactive jobs
- Processor's time is shared among multiple users – turn by turn...
- Multiple users simultaneously access the system through terminals

Batch Multiprogramming versus Time Sharing

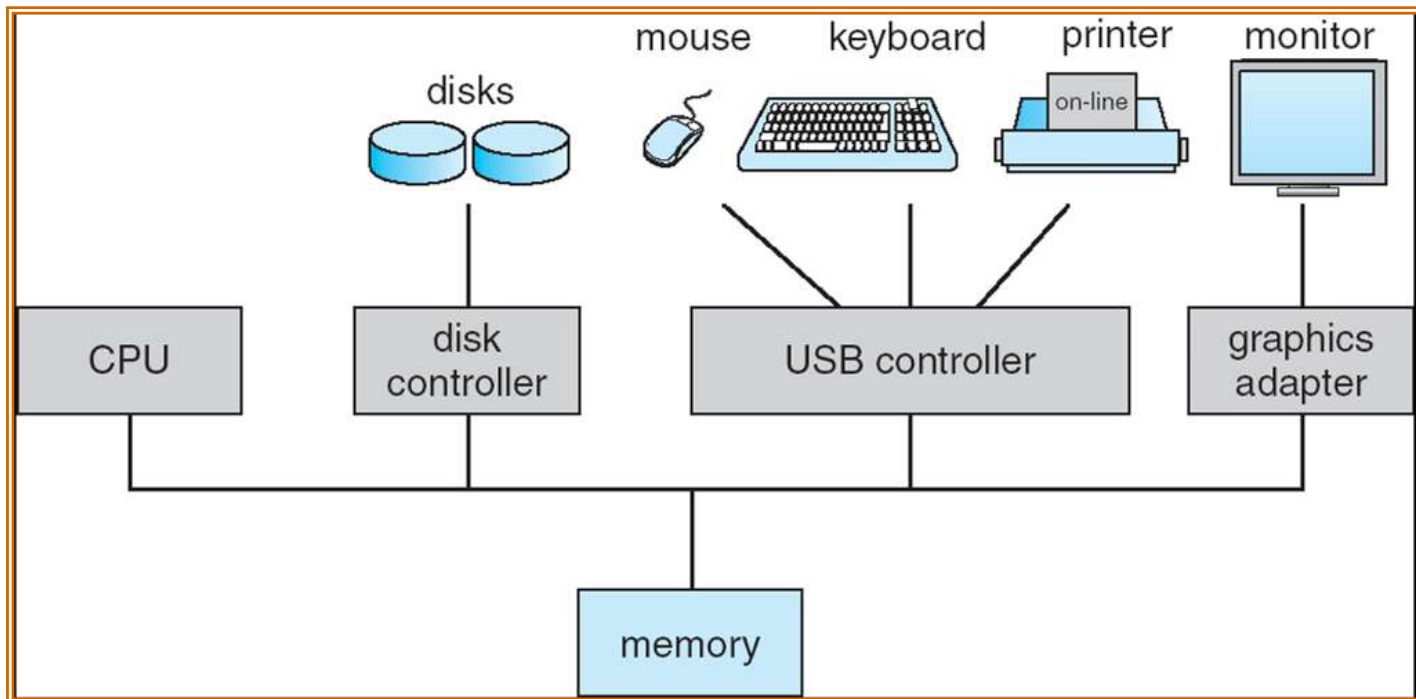
	Batch Multiprogramming	Time Sharing
Principal objective	Maximize processor use	Minimize response time
Source of directives to operating system	Job control language commands provided with the job	Commands entered at the terminal

Computer Startup

- **bootstrap program** is loaded at power-up or reboot
 - Typically stored in ROM or EPROM, generally known as **firmware**
 - Initializes all aspects of system
 - Loads operating system kernel and starts execution

Computer System Organization

- Computer-system operation
 - One or more CPUs, device controllers connect through common bus providing access to shared memory
 - Concurrent execution of CPUs and devices competing for memory cycles



Computer-System Operation

- I/O devices and the CPU can execute concurrently.
- Each device controller is in charge of a particular device type.
- Each device controller has a local buffer.
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller.
- Device controller informs CPU that it has finished its operation by causing an *interrupt*.

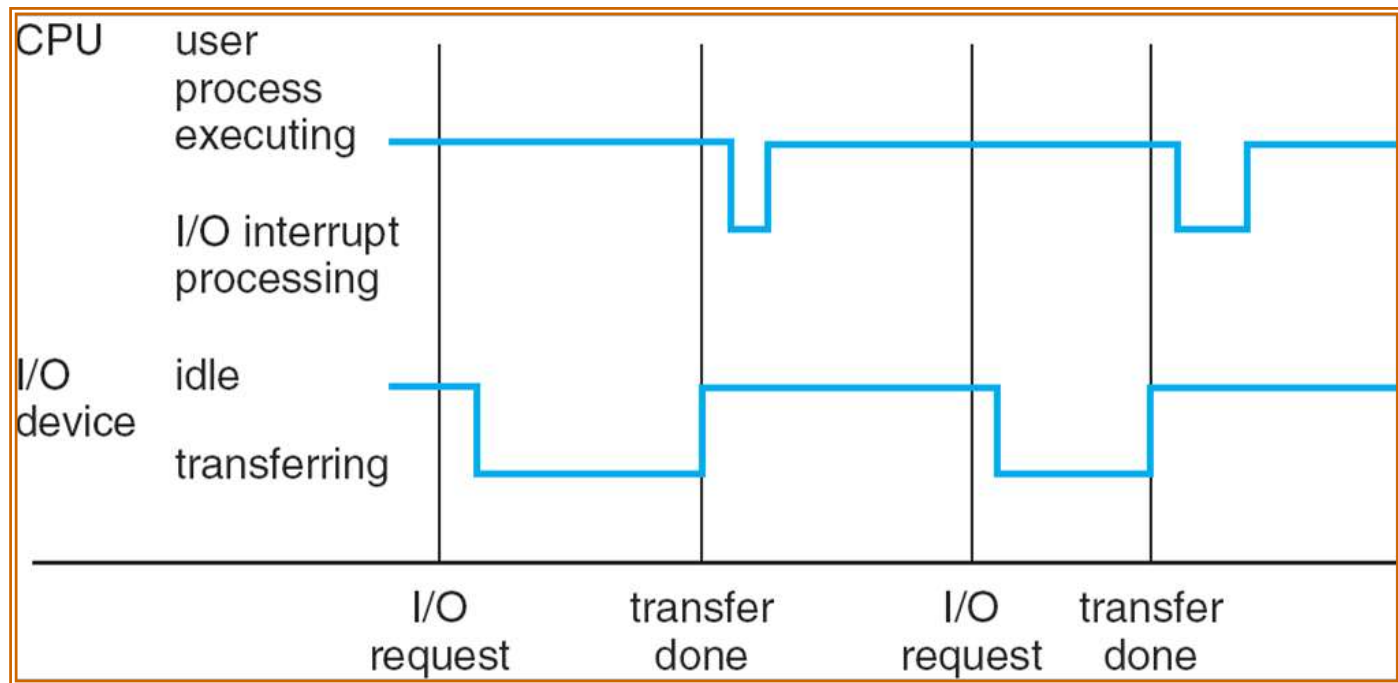
Common Functions of Interrupts

- Interrupt transfers control to the interrupt service routine generally, through the *interrupt vector*, which contains the addresses of all the service routines.
- Interrupt architecture must save the address of the interrupted instruction.
- Incoming interrupts are *disabled* while another interrupt is being processed to prevent a *lost interrupt*.
- A *trap* is a software-generated interrupt caused either by an error or a user request.
- An operating system is *interrupt* driven.

Interrupt Handling

- The operating system preserves the state of the CPU by storing registers and the program counter.
- Determines which type of interrupt has occurred:
 - *polling*
 - *vectored* interrupt system
- Separate segments of code determine what action should be taken for each type of interrupt

Interrupt Timeline

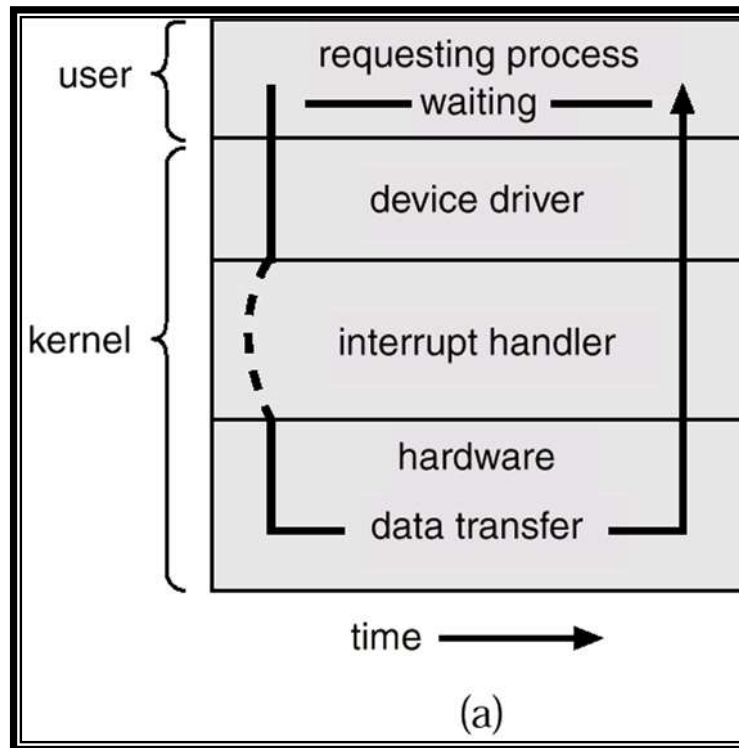


I/O Structure

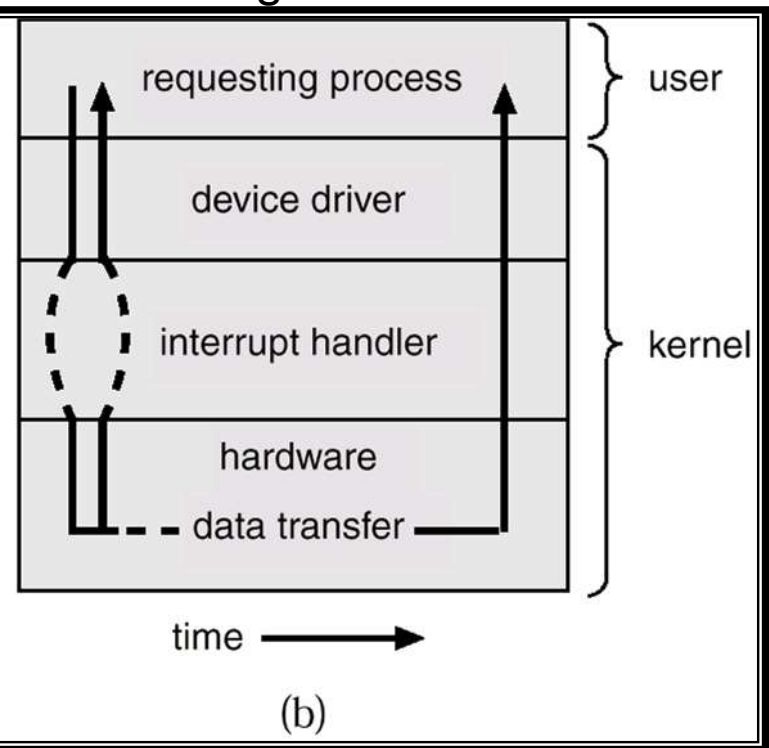
- After I/O starts, control returns to user program only upon I/O completion.
 - Wait instruction idles the CPU until the next interrupt
 - Wait loop (contention for memory access).
 - At most one I/O request is outstanding at a time, no simultaneous I/O processing.
- After I/O starts, control returns to user program without waiting for I/O completion.
 - *System call* – request to the operating system to allow user to wait for I/O completion.
 - *Device-status table* contains entry for each I/O device indicating its type, address, and state.
 - Operating system indexes into I/O device table to determine device status and to modify table entry to include interrupt.

Two I/O Methods

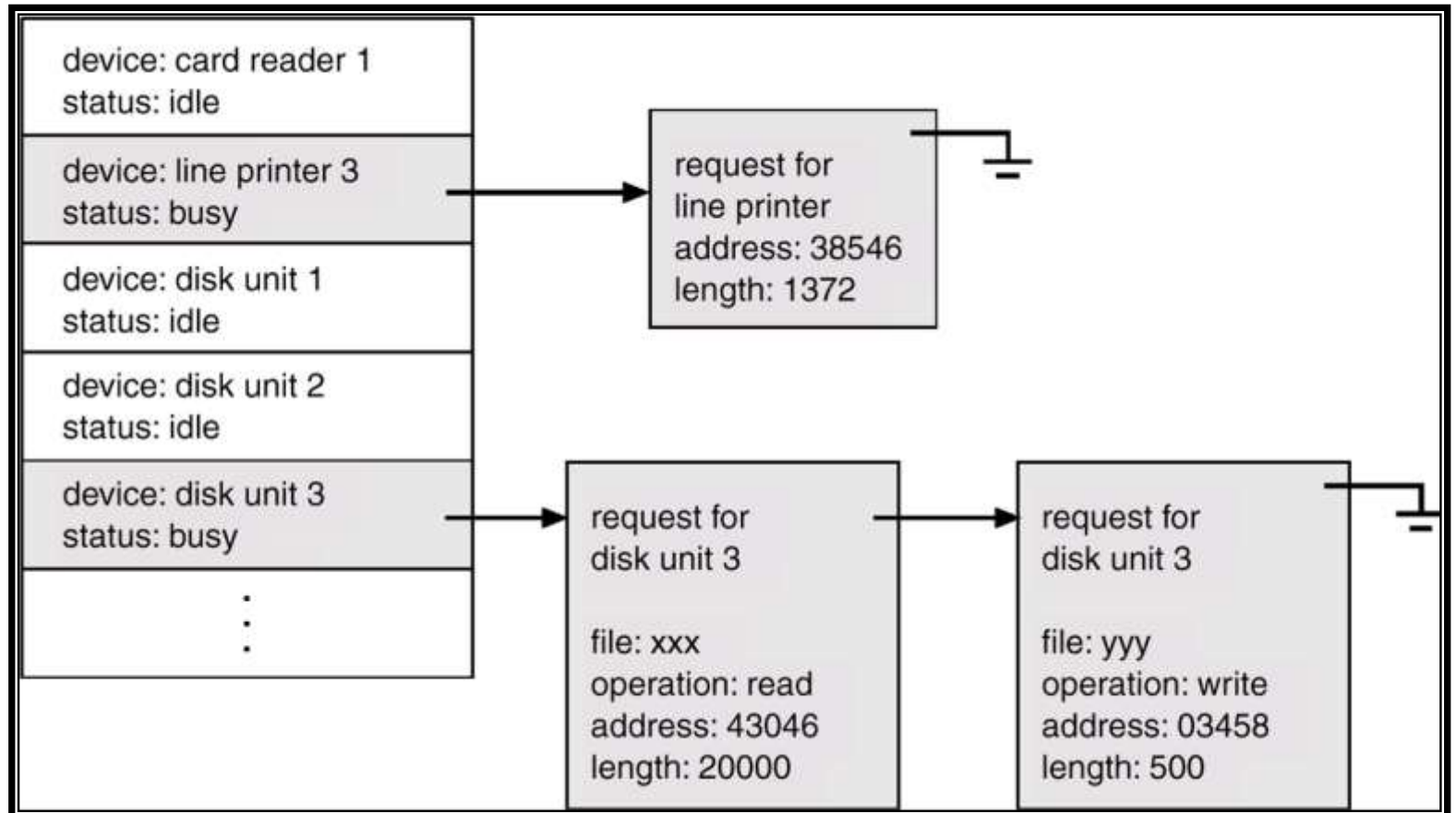
Synchronous
S



Asynchronous
S



Device-Status Table



Direct Memory Access Structure

- Used for high-speed I/O devices able to transmit information at close to memory speeds.
- Device controller transfers blocks of data from buffer storage directly to main memory without CPU intervention.
- Only one interrupt is generated per block, rather than the one interrupt per byte.

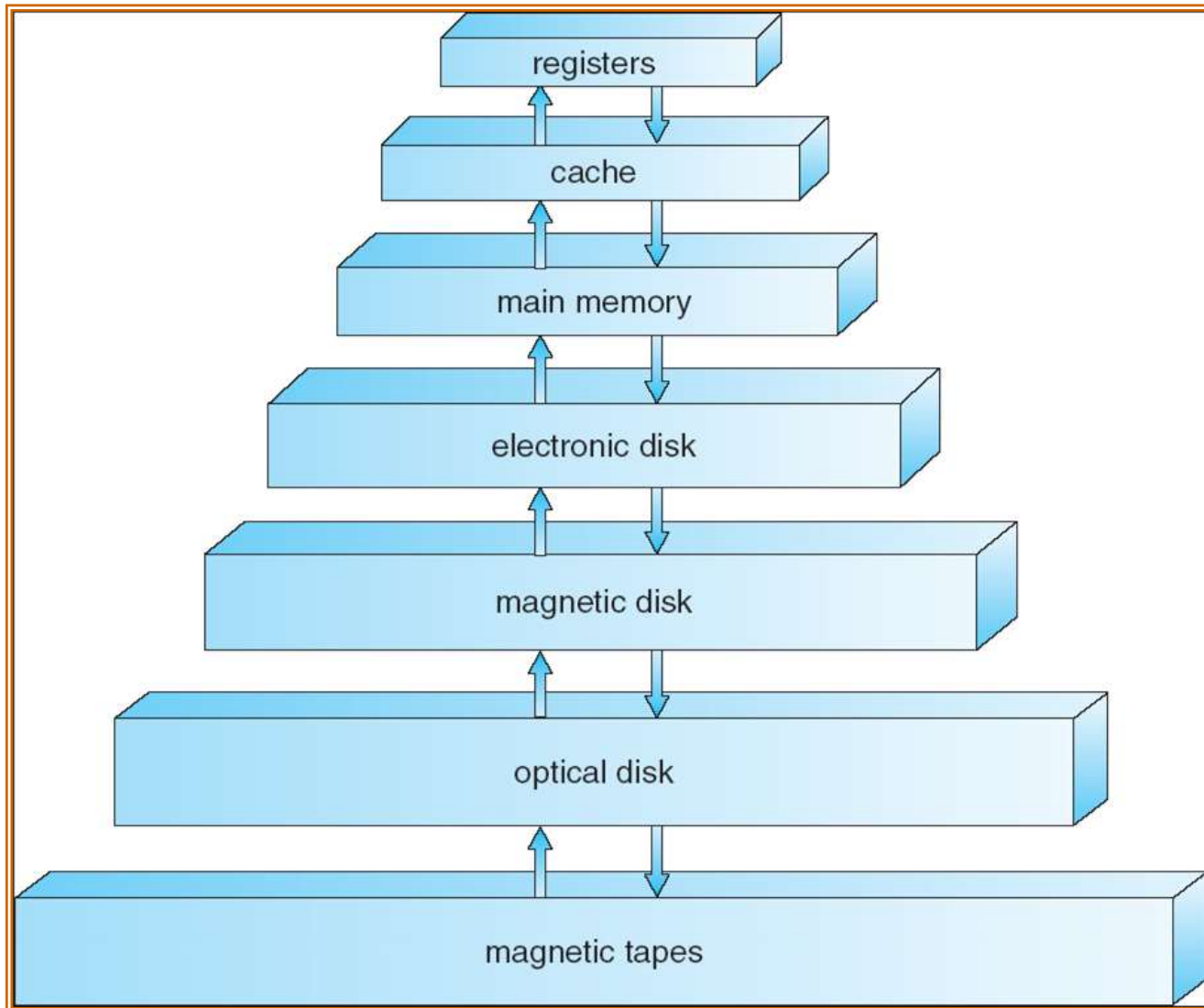
Storage Structure

- Main memory – only large storage media that the CPU can access directly.
- Secondary storage – extension of main memory that provides large nonvolatile storage capacity.
- Magnetic disks – rigid metal or glass platters covered with magnetic recording material
 - Disk surface is logically divided into *tracks*, which are subdivided into *sectors*.
 - The *disk controller* determines the logical interaction between the device and the computer.

Storage Hierarchy

- Storage systems organized in hierarchy.
 - Speed
 - Cost
 - Volatility
- *Caching* – copying information into faster storage system; main memory can be viewed as a last *cache* for secondary storage.

Storage-Device Hierarchy



Caching

- Important principle, performed at many levels in a computer (in hardware, operating system, software)
- Information in use copied from slower to faster storage temporarily
- Faster storage (cache) checked first to determine if information is there
 - If it is, information used directly from the cache (fast)
 - If not, data copied to cache and used there
- Cache smaller than storage being cached
 - Cache management important design problem
 - Cache size and replacement policy

Performance of Various Levels of Storage

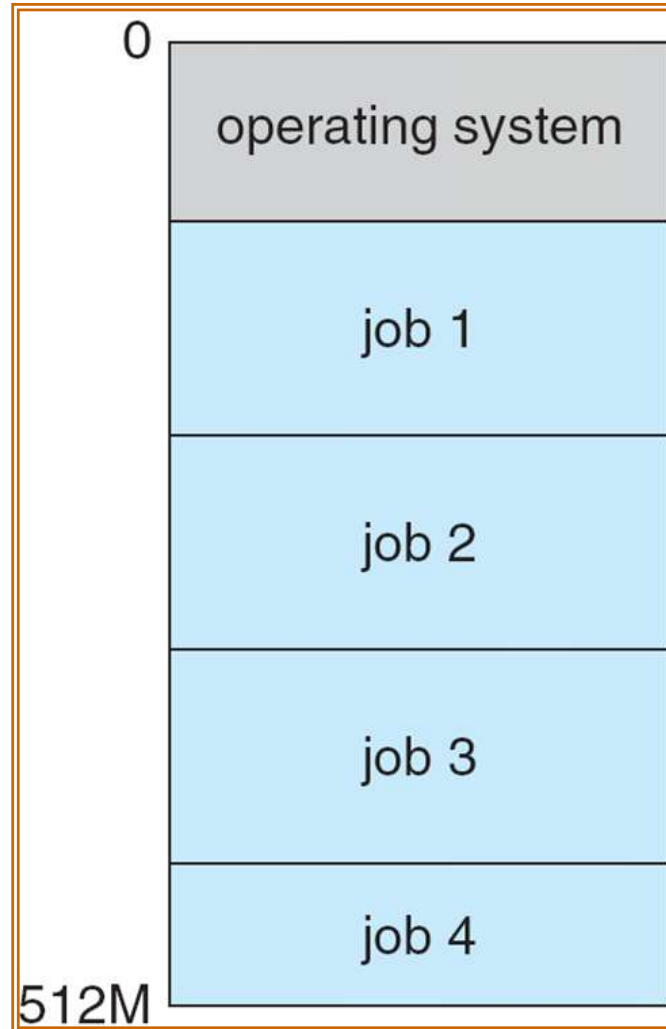
- Movement between levels of storage hierarchy can be explicit or implicit

Level	1	2	3	4
Name	registers	cache	main memory	disk storage
Typical size	< 1 KB	> 16 MB	> 16 GB	> 100 GB
Implementation technology	custom memory with multiple ports, CMOS	on-chip or off-chip CMOS SRAM	CMOS DRAM	magnetic disk
Access time (ns)	0.25 – 0.5	0.5 – 25	80 – 250	5,000.000
Bandwidth (MB/sec)	20,000 – 100,000	5000 – 10,000	1000 – 5000	20 – 150
Managed by	compiler	hardware	operating system	operating system
Backed by	cache	main memory	disk	CD or tape

Operating System Structure

- **Multiprogramming** needed for efficiency
 - Single user cannot keep CPU and I/O devices busy at all times
 - Multiprogramming organizes jobs (code and data) so CPU always has one to execute
 - A subset of total jobs in system is kept in memory
 - One job selected and run via **job scheduling**
 - When it has to wait (for I/O for example), OS switches to another job
- **Timesharing (multitasking)** is logical extension in which CPU switches jobs so frequently that users can interact with each job while it is running, creating **interactive** computing
 - **Response time** should be < 1 second
 - Each user has at least one program executing in memory ☐ **process**
 - If several jobs ready to run at the same time ☐ **CPU scheduling**
 - If processes don't fit in memory, **swapping** moves them in and out to run
 - **Virtual memory** allows execution of processes not completely in memory

Memory Layout for Multiprogrammed System

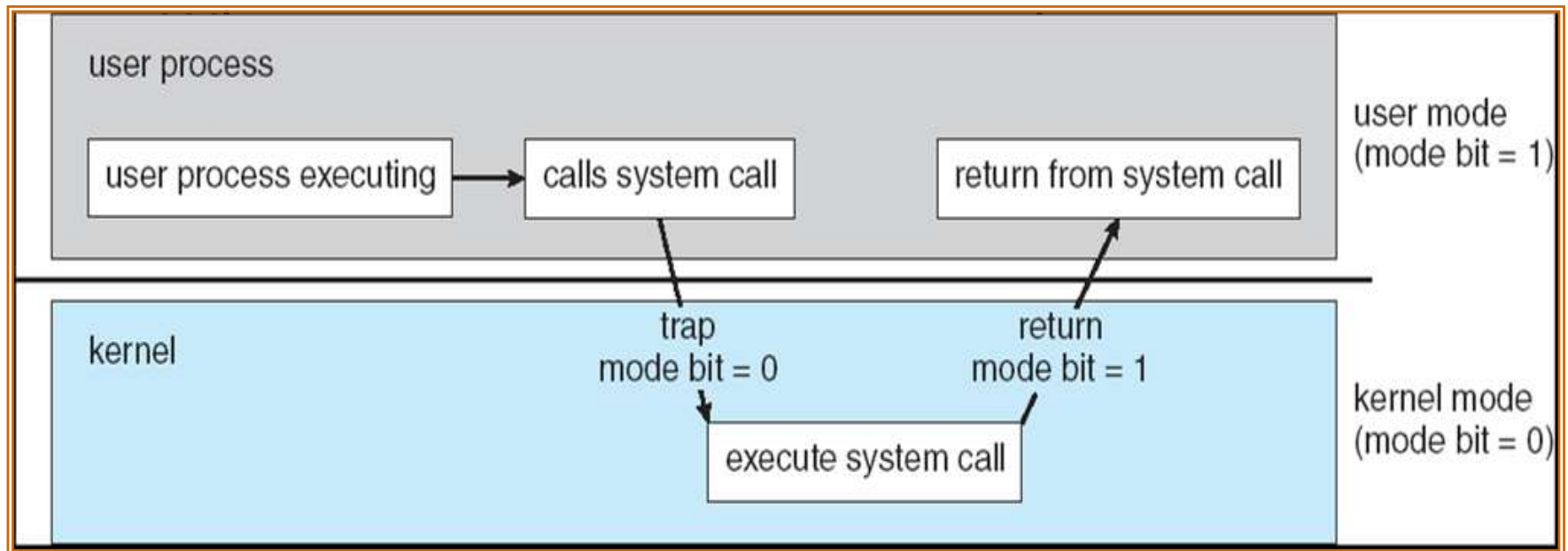


Operating-System Operations

- Interrupt driven by hardware
- Software error or request creates **exception** or **trap**
 - Division by zero, request for operating system service
- Other process problems include infinite loop, processes modifying each other or the operating system
- **Dual-mode** operation allows OS to protect itself and other system components
 - **User mode** and **kernel mode**
 - **Mode bit** provided by hardware
 - Provides ability to distinguish when system is running user code or kernel code
 - Some instructions designated as **privileged**, only executable in kernel mode
 - System call changes mode to kernel, return from call resets it to user

Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
 - Set interrupt after specific period
 - Operating system decrements counter



Process Management

- A process is a program in execution. It is a unit of work within the system. Program is a *passive entity*, process is an *active entity*.
- Process needs resources to accomplish its task
 - CPU, memory, I/O, files
 - Initialization data
- Process termination requires reclaim of any reusable resources
- Single-threaded process has one **program counter** specifying location of next instruction to execute
 - Process executes instructions sequentially, one at a time, until completion
- Multi-threaded process has one program counter per thread
- Typically system has many processes, some user, some operating system running concurrently on one or more CPUs
 - Concurrency by multiplexing the CPUs among the processes / threads

Process Management Activities

The operating system is responsible for the following activities in connection with process management:

- Creating and deleting both user and system processes
- Suspending and resuming processes
- Providing mechanisms for process synchronization
- Providing mechanisms for process communication
- Providing mechanisms for deadlock handling

Memory Management

- All data in memory before and after processing
- All instructions in memory in order to execute
- Memory management determines what is in memory when
 - Optimizing CPU utilization and computer response to users
- Memory management activities
 - Keeping track of which parts of memory are currently being used and by whom
 - Deciding which processes (or parts thereof) and data to move into and out of memory
 - Allocating and deallocating memory space as needed

Storage Management

- OS provides uniform, logical view of information storage
 - Abstracts physical properties to logical storage unit - **file**
 - Each medium is controlled by device (i.e., disk drive, tape drive)
 - Varying properties include access speed, capacity, data-transfer rate, access method (sequential or random)
- File-System management
 - Files usually organized into directories
 - Access control on most systems to determine who can access what
 - OS activities include
 - Creating and deleting files and directories
 - Primitives to manipulate files and dirs
 - Mapping files onto secondary storage
 - Backup files onto stable (non-volatile) storage media

Mass-Storage Management

- Usually disks used to store data that does not fit in main memory or data that must be kept for a “long” period of time.
- Proper management is of central importance
- Entire speed of computer operation hinges on disk subsystem and its algorithms
- OS activities
 - Free-space management
 - Storage allocation
 - Disk scheduling
- Some storage need not be fast
 - Tertiary storage includes optical storage, magnetic tape
 - Still must be managed
 - Varies between WORM (write-once, read-many-times) and RW (read-write)

I/O Subsystem

- One purpose of OS is to hide peculiarities of hardware devices from the user
- I/O subsystem responsible for
 - Memory management of I/O including buffering (storing data temporarily while it is being transferred), caching (storing parts of data in faster storage for performance), spooling (the overlapping of output of one job with input of other jobs)
 - General device-driver interface
 - Drivers for specific hardware devices

Protection and Security

- **Protection** – any mechanism for controlling access of processes or users to resources defined by the OS
- **Security** – defense of the system against internal and external attacks
 - Huge range, including denial-of-service, worms, viruses, identity theft, theft of service
- Systems generally first distinguish among users, to determine who can do what
 - User identities (**user IDs**, security IDs) include name and associated number, one per user
 - User ID then associated with all files, processes of that user to determine access control
 - Group identifier (**group ID**) allows set of users to be defined and controls managed, then also associated with each process, file
 - **Privilege escalation** allows user to change to effective ID with more rights