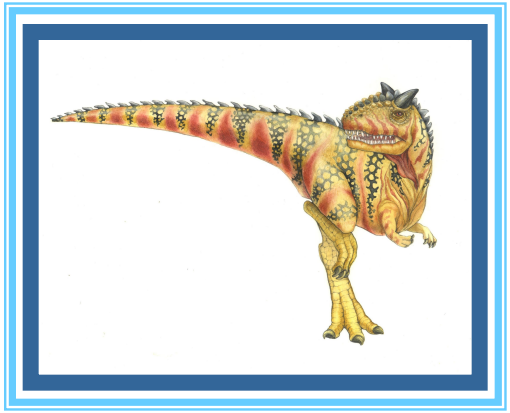
Operating Systems
CSE-204

Dr. Durr-e-Nayab
nayab.khan@uetpeshawar.edu.pk



Course Learning Outcomes

- **CLO-1:**
  Understand the basic concepts behind implementation of Operating Systems, their types, structures, features and services
- **CLO-2:**
  Demonstrate the concepts of processes, multiprocessing and multithreading in solving different problems
- **CLO-3:**
  Understand and analyze CPU/Process scheduling and synchronization algorithms.

Grading Criteria

- **In an ideal scenario:**
  - **6 Assignments:** 3 assignments before midterm and 3 before finalterm exam
  - **6 Quizzes:** 3 quizzes before midterm and 3 before finalterm exam, based on assignments

Assignments	10%
Quizzes	20%
Mid Term	20%
Final Examination	50%
TOTAL	100%

- **RESOURCES**
  - BOOKs RECOMMENDED
    - Abraham Silberschatz, Peter Baer Galvin, and Greg Gagne: **Operating System Concepts**, 9th or later Edition (Uploaded in google classroom)
  - REFERENCE BOOKS
    - **UNIX Systems Programming: Communication, Concurrency, and Threads** by K. A. Robbins and S. Robbins. Prentice Hall, 2003. ISBN: 0-13-042411-0.

About Lectures

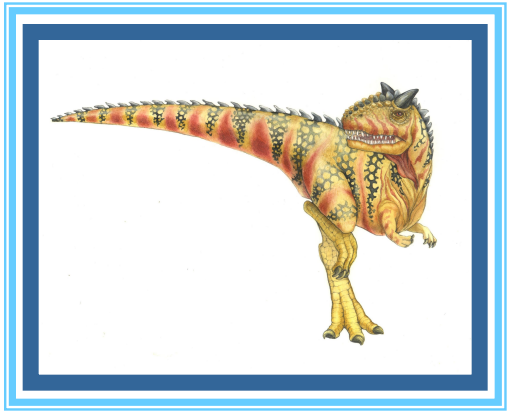
In an effort to make this course enjoyable for everybody:

- **Be on time!** This will only benefit YOU!
- If you have any queries, **Just ask** me or write it down!
- **Class participation** is mandatory. Do not just sit to receive.
- Please **turn off** your pagers and cell-phones or on silent! If you have any important or expected one, attend it outside the class
- Bring your own book, table, notebook and calculator
- In your written works it is **unacceptable to use someone else's work** without reference. Copied assignments, quizzes or other tasks will be tantamount to **cheating** and **no grading** will be done.
- Your work will **never be re-evaluated** for rejected/missed assignment.
- **Class representatives** should collect and submit the assignments, quizzes and requests on behalf of their section in my office once and for all.

Operating Systems
CSE-204

- **Overview**
  What are operating systems.
  What operating systems do.
  How the operating systems are designed and constructed.
- **Process management and Process coordination**
  Process concepts and concurrency
  Inter-Process communication, scheduling, process synchronization, and deadlock handling, and priorities
- **Memory management and Storage management**
  Main memory during the execution of a process
  Utilization of the CPU and the speed of its response
  File system, mass storage, and I/0 handling in a modern computer system
- **Protection and security**
  Processes in an OS must be protected from one another's activities
  Controlling the access of programs, processes, or users to the resources
- **Distributed systems and special purpose systems**
  Processors that do not share memory or a clock
  Mechanisms for process synchronization and communication

Chapter 1: Introduction



Chapter 1: Introduction

- What is Operating Systems
- Computer-System Organization
- Computer-System Architecture
- Operating-System Structure
- Operating-System Operations
- Process Management
- Memory Management
- Storage Management

Agenda for Today

- Describe Operating Systems in the context of Computer Systems
- Operating Systems from User and System perspective
- What does Operating Systems do?
- To explore several open-source operating systems
- Some core concepts
  - OS kernel, OS modes
  - Processes, Types, Attributes
  - Threads, Types, Attributes
  - Interrupts, Interrupt Vector, Interrupt Handling/ Service
  - System/ I/O Calls
  - Polling/ Signals
  - Trap/ Exception, Trap handling / Service
  - Program Counter

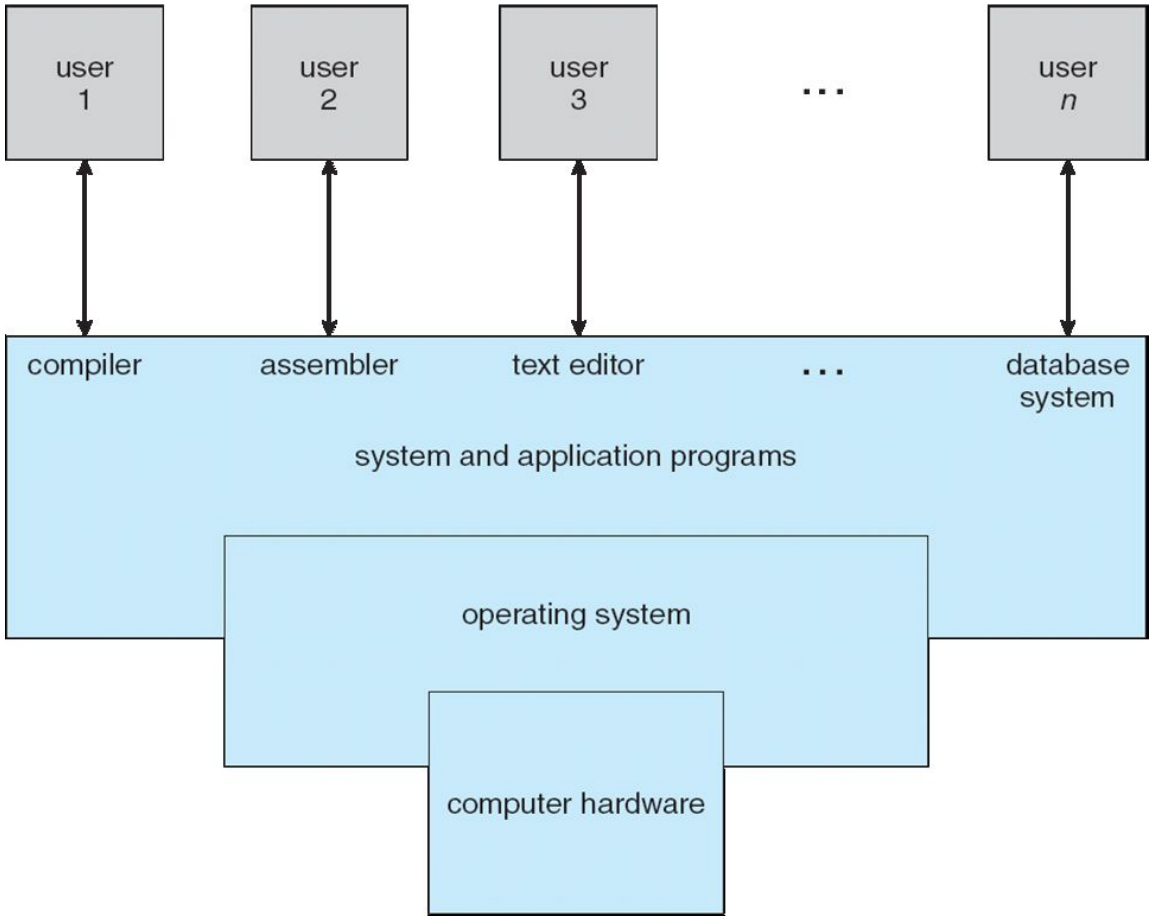
What is an Operating System?

- 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 to use
  - Use the computer hardware in an efficient manner

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



What Operating Systems Do

- Depends on the point of view
- **Users** want convenience, **ease of use** and **good performance**
  - Don't care about **resource utilization**
- But shared computer such as **mainframe** or **minicomputer** must keep all users happy
- Users of dedicated systems such as **workstations** have dedicated resources but frequently use shared resources from **servers**
- Handheld computers are resource poor, optimized for usability and battery life
- Some computers have little or no user interface, such as embedded computers in devices and automobiles

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

Operating System Definition (Cont.)

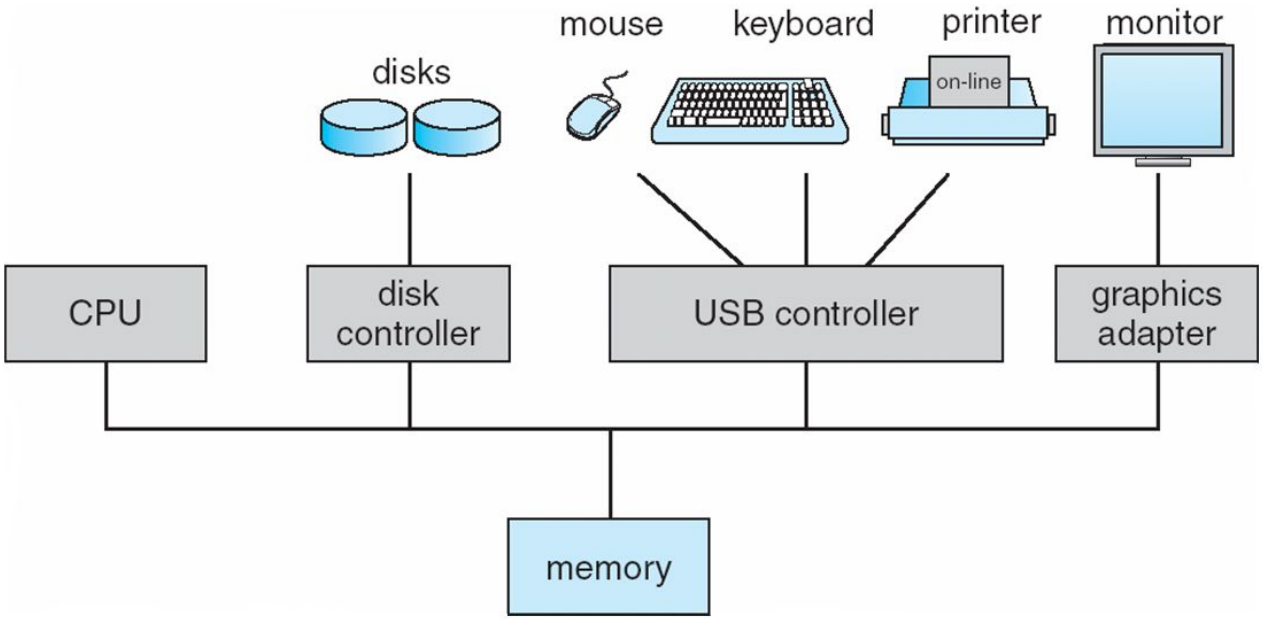
- No universally accepted definition
- "Everything a vendor ships when you order an operating system" is a good approximation
  - But varies wildly
- "The one program running at all times on the computer" is the **kernel**.
- Everything else is either
  - a system program (ships with the operating system) , or
  - an application program.

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

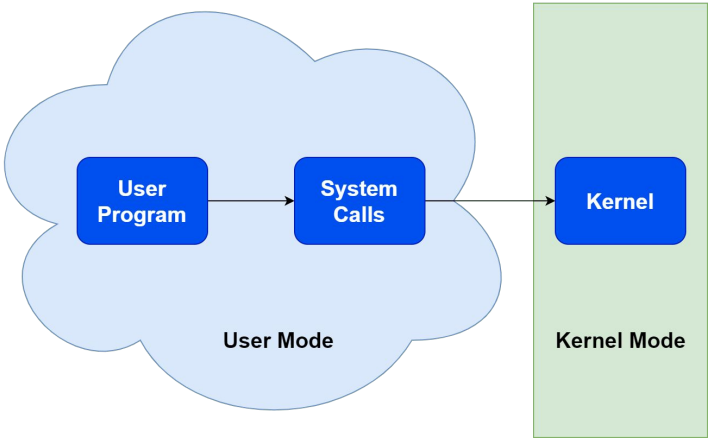
- I/O devices and the CPU can execute concurrently
- Each device controller is in charge of a particular device type
- Each device controller has a local buffer
- CPU moves data from/to main memory to/from local buffers
- I/O is from the device to local buffer of controller
- Device controller informs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

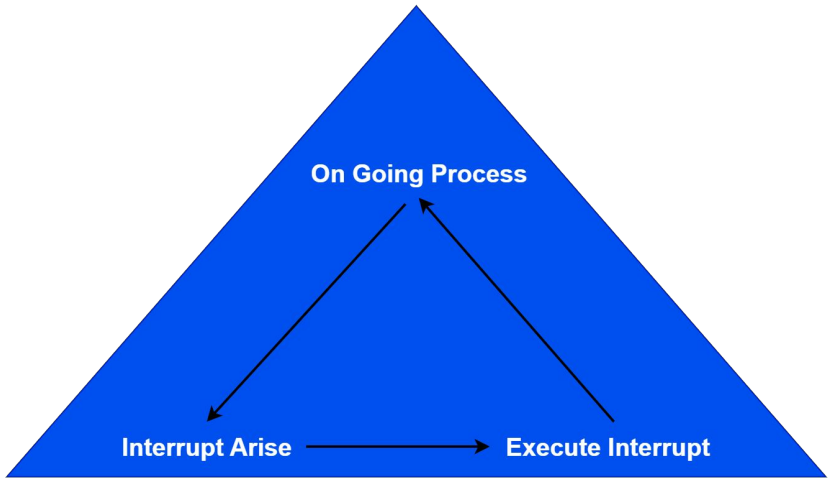
Common Functions of Interrupts

- 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
- A **trap** or **exception** is a software-generated interrupt caused either by an error or a user request, conditions like invalid memory access, division by zero, or a breakpoint can trigger a trap in an OS. Invokes Kernel Routine.
- 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





System call/ user process versus Interrupts

Operating-System Operations

- **Interrupt driven** (hardware and software)
  - Hardware interrupt by one of the devices
  - Software interrupt (**exception** or **trap):**
    - Software error (e.g., division by zero)
    - Request for operating system service
    - Other process problems include infinite loop, processes modifying each other or the operating system

I/O Structure

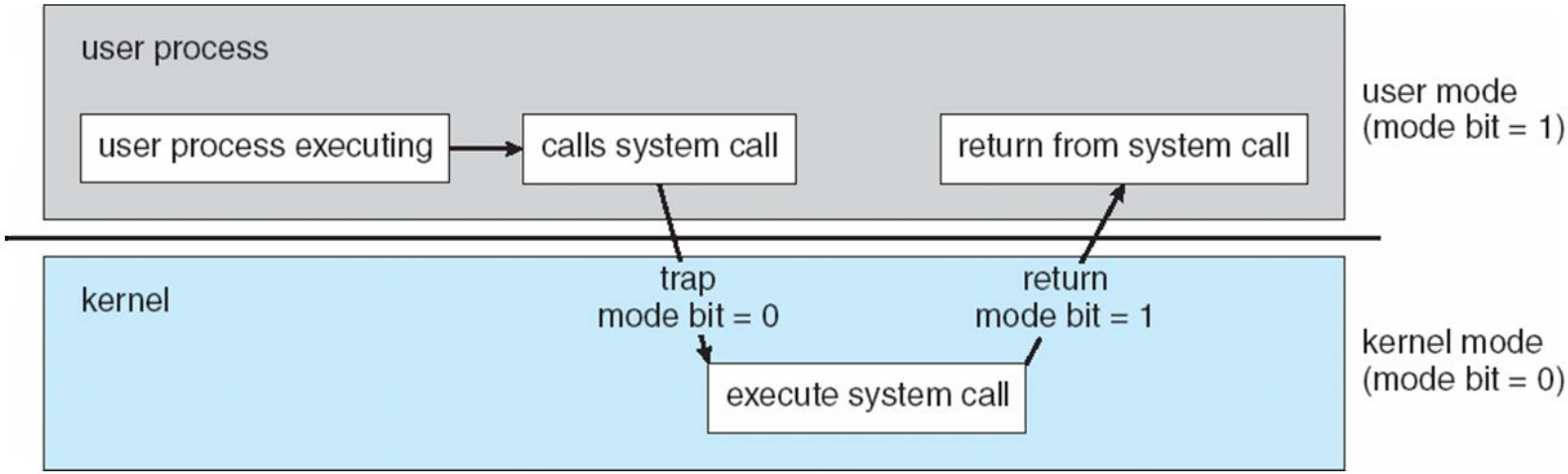
- After I/O starts, control returns to user program only upon I/O completion
  - Wait instruction idles the CPU until the next interrupt
  - Wait loop (contention for memory access)
  - At most one I/O request is outstanding at a time, no simultaneous I/O processing
- After I/O starts, control returns to user program without waiting for I/O completion
  - **System call** – request to the OS 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
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

Operating-System Operations (cont.)

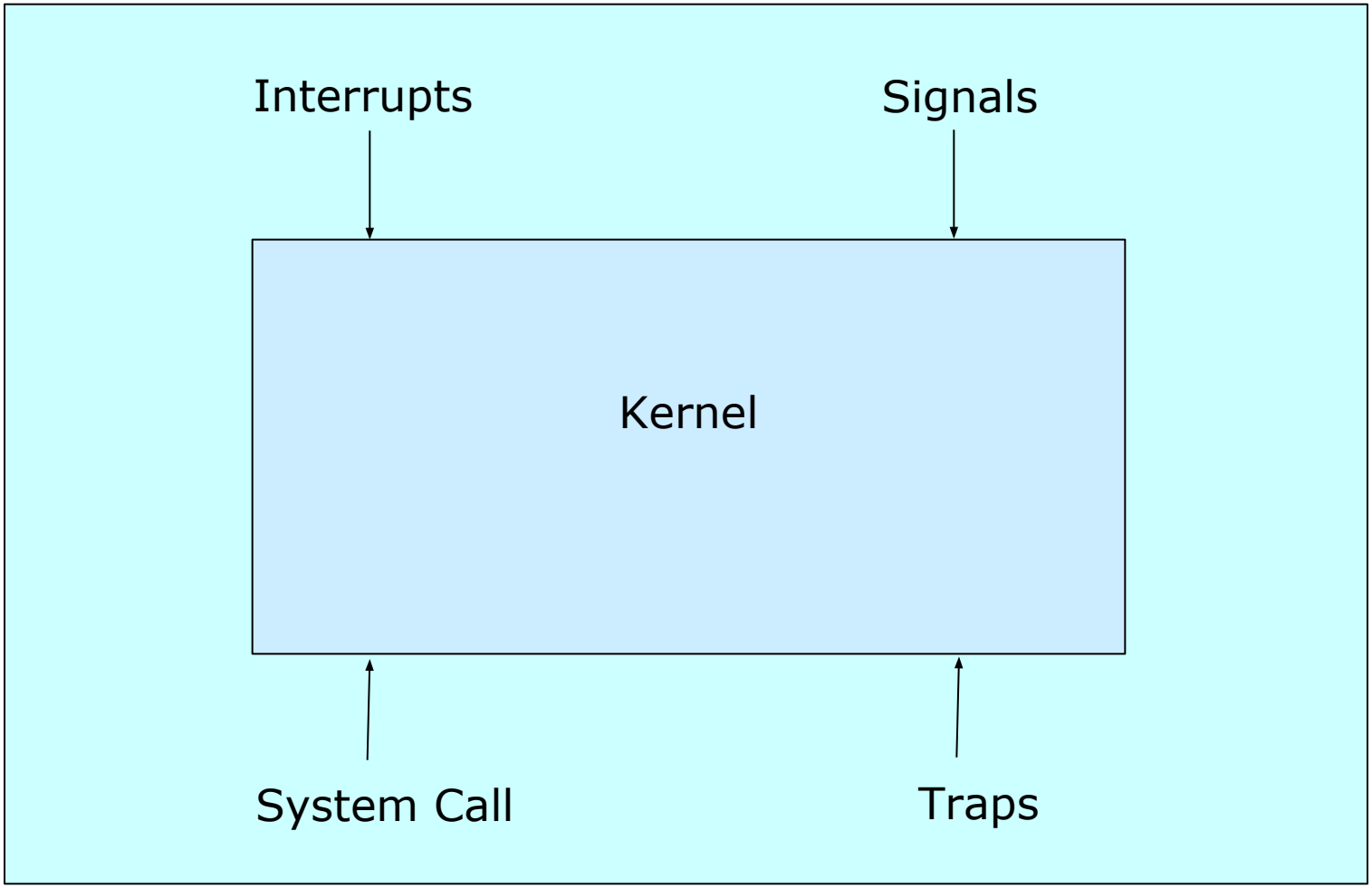
- **Dual-mode** operation allows OS to protect itself and other system components
  - **User mode** and **kernel mode**
  - **Mode bit** provided by hardware
    - Provides ability to distinguish when system is running user code or kernel code
    - Some instructions designated as **privileged**, only executable in kernel mode
    - System call changes mode to kernel, return from call resets it to user
- Increasingly CPUs support multi-mode operations
  - i.e. **virtual machine manager** (**VMM**) mode for guest **VMs**

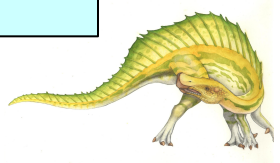
Transition from User to Kernel Mode

- Timer to prevent infinite loop / process hogging resources
  - Timer is set to interrupt the computer after some time period
  - Keep a counter that is decremented by the physical clock.
  - Operating system set the counter (privileged instruction)
  - When counter zero generate an interrupt
  - Set up before scheduling process to regain control or terminate program that exceeds allotted time



Entry Points to Kernel Mode





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 OS 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
  - OS indexes into I/O device table to determine device status and to modify table entry to include interrupt

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

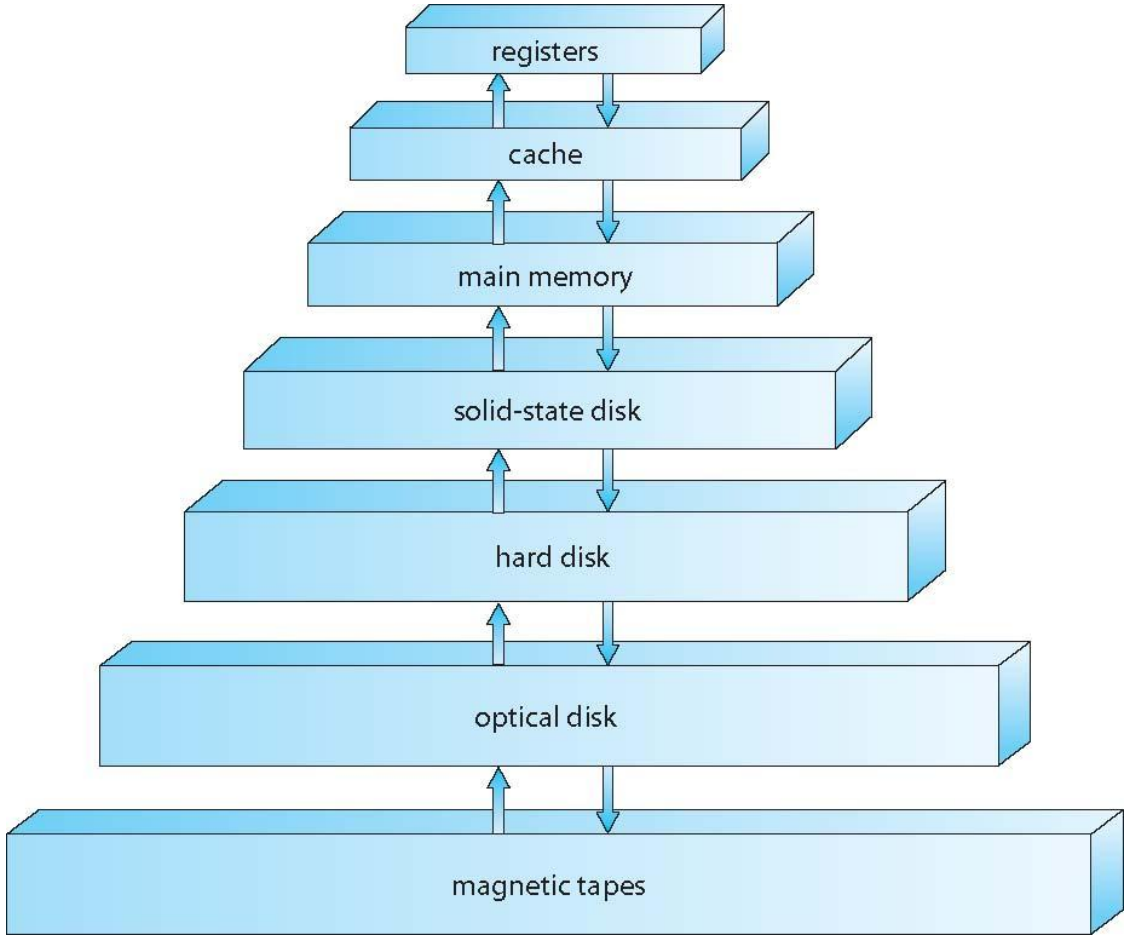
Memory Management

- To execute a program all (or part) of the instructions must be in memory
- All (or part) of the data that is needed by the program must be in memory.
- Memory management determines what is in memory and 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 directories
    - Mapping files onto secondary storage
    - Backup files onto stable (non-volatile) storage media

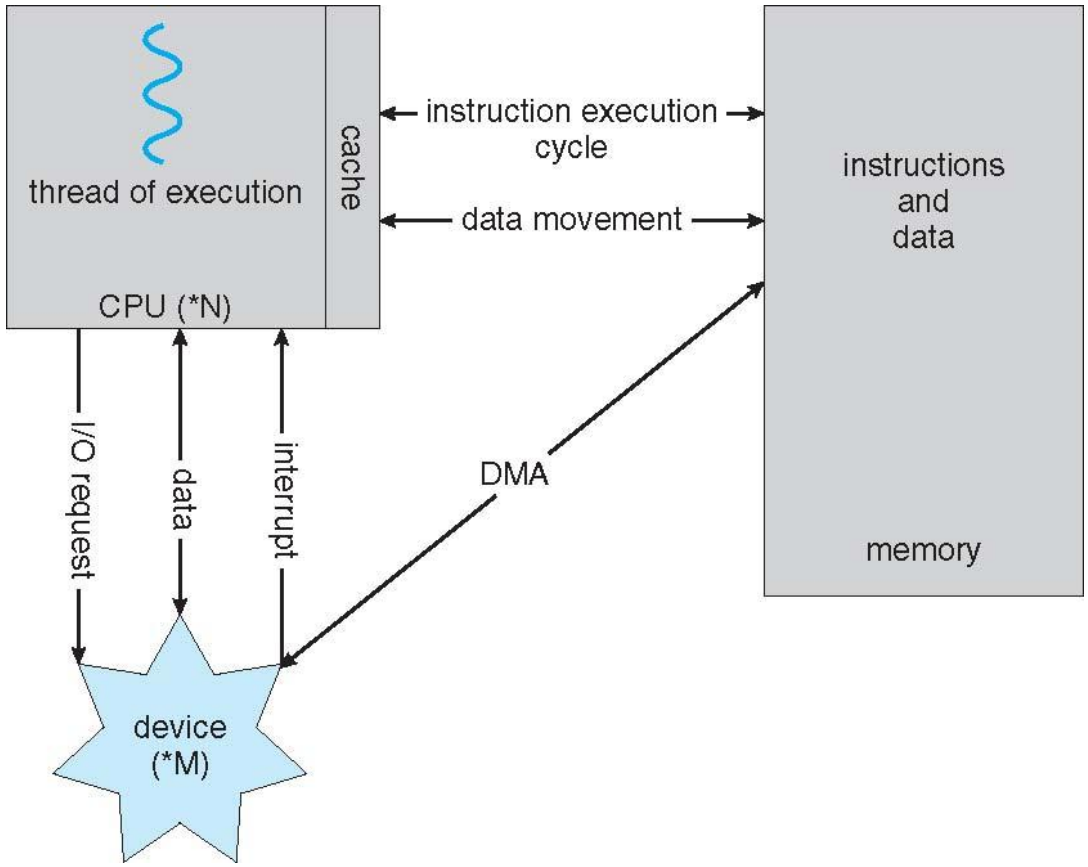
Storage-Device Hierarchy



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 – by OS or applications
  - Varies between WORM (write-once, read-many-times) and RW (read-write)

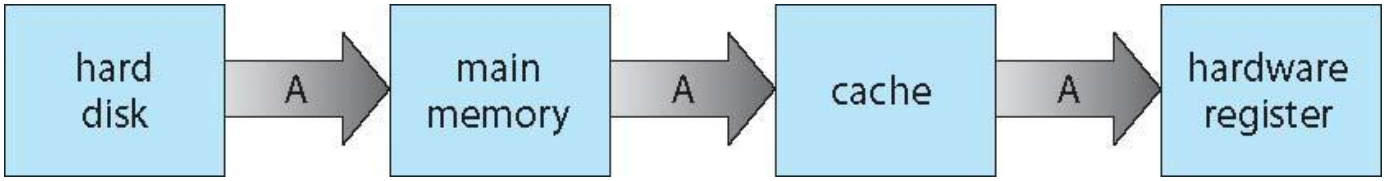
How a Modern Computer Works



A von Neumann architecture

Migration of data "A" from Disk to Register

- Multitasking environments must be careful to use most recent value, no matter where it is stored in the storage hierarchy



- Multiprocessor environment must provide **cache coherency** in hardware such that all CPUs have the most recent value in their cache
- Distributed environment situation even more complex
  - Several copies of a datum can exist
  - Various solutions covered in Chapter 17

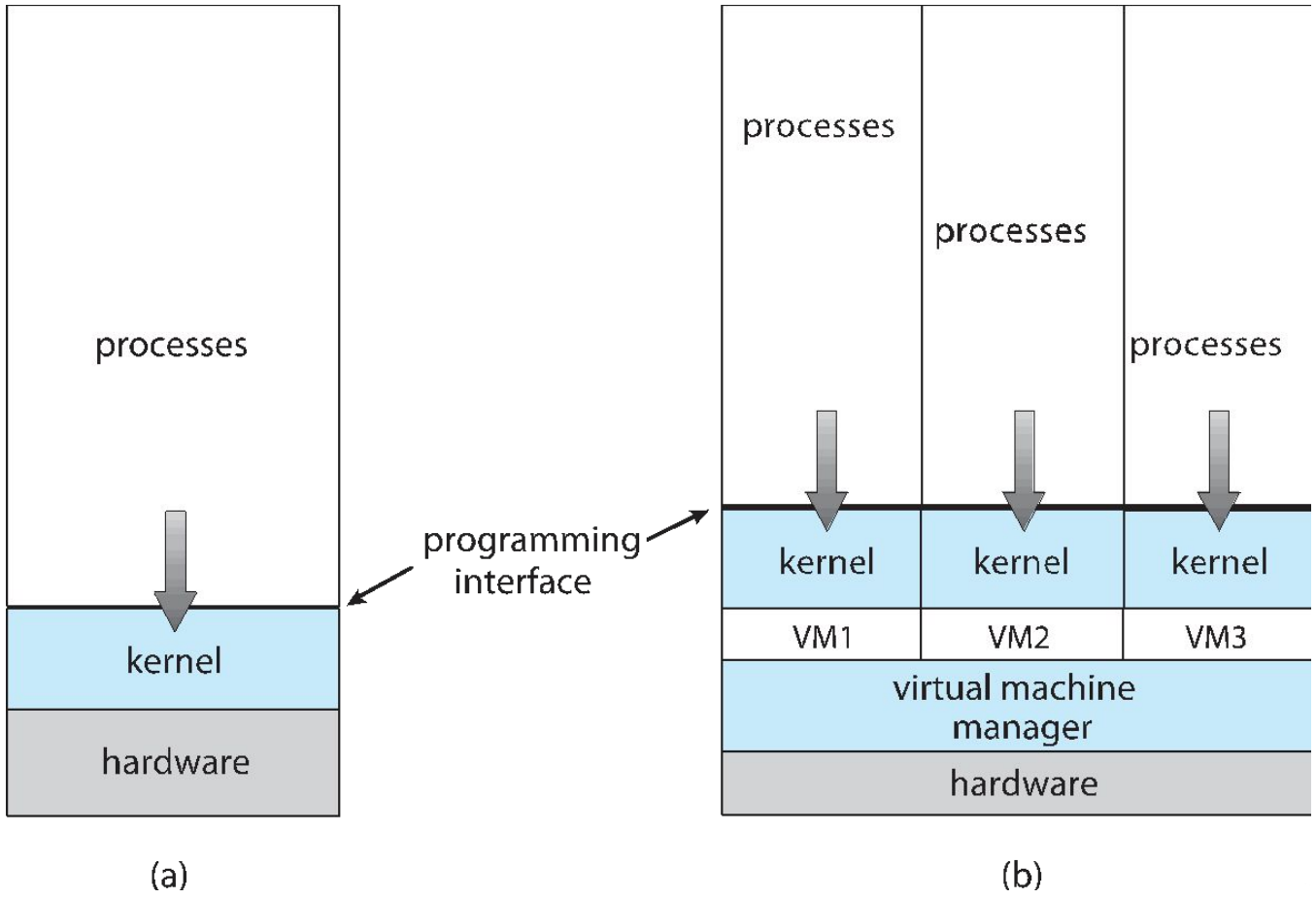
Protection and Security

- **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

Computing Environments - Virtualization

- Use cases involve laptops and desktops running multiple OSes for exploration or compatibility
  - Apple laptop running Mac OS X host, Windows as a guest
  - Developing apps for multiple OSes without having multiple systems
  - QA testing applications without having multiple systems
  - Executing and managing compute environments within data centers
- VMM can run natively, in which case they are also the host
  - There is no general purpose host then (VMware ESX and Citrix XenServer)

Computing Environments - Virtualization



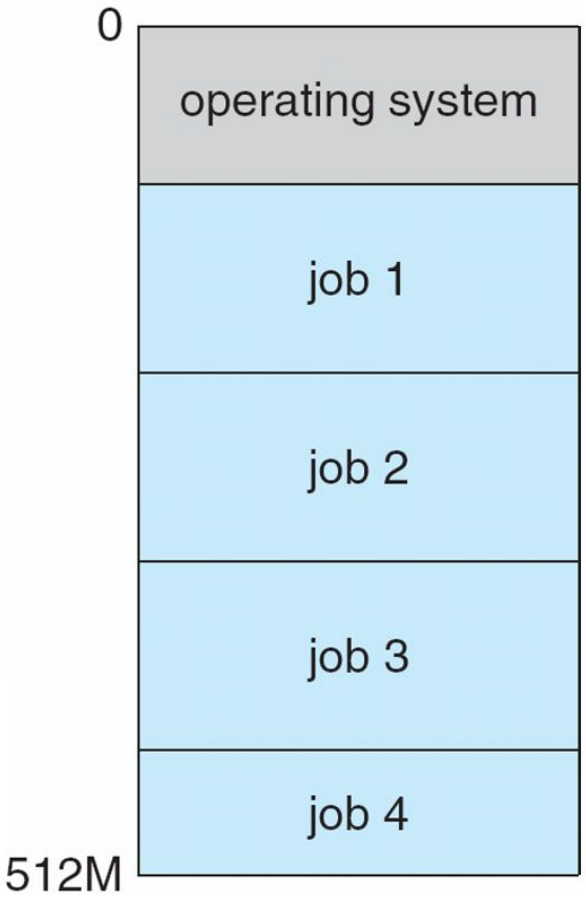
Operating System Concepts

- **Types of Operating Systems**
  - **Single User Systems** (**Personal Computer systems**)
    - Only one user at a time □ **Single Tasking Systems**
    - Interactive and executes codes for single user only
    - Processes of one user only (**Process / CPU scheduling**)
    - Single-Tasking (MS DOS/ Palm OS)
    - Multi-Tasking (Windows, UNIX, LINUX, MAC OS)
      - Pre-Emptive Mode (Predefined slots for each process)
      - Co-operative Mode (Inter process communication)
  - **Multiprogramming** (**Batch system**) 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
    - Example: payroll system, Bills, reports printing systems, Network simulators

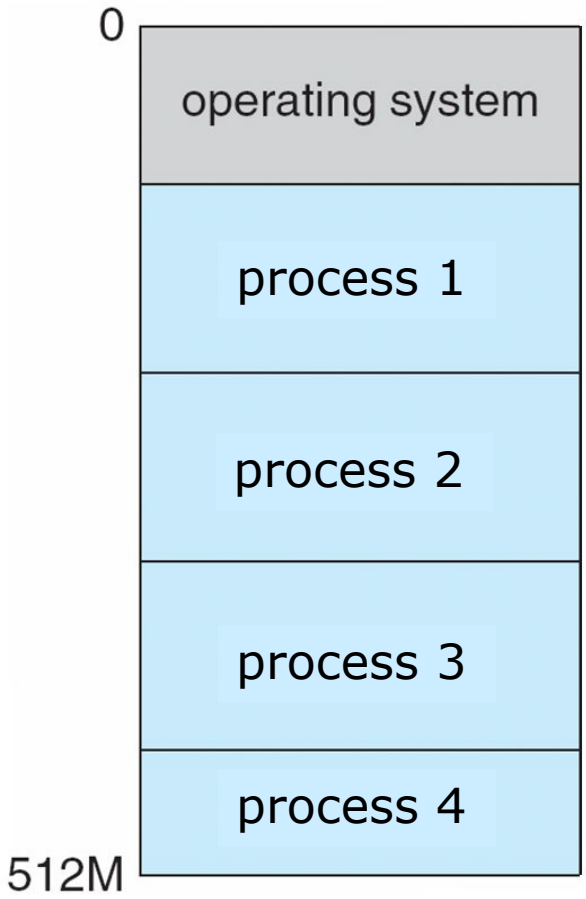
Operating System Concepts

- **Types of Operating Systems**
  - **Multi Processing** (**multitasking**) is logical extension in which CPU switches processes of similar kind frequently to give impression of parallelism
    - **Similar Processes**
    - Switching between CPU Burst and I/O burst □ **Concurrent Process Management**
    - If several jobs ready to run at the same time □ **Inter Process Communication**
    - If processes don't fit in memory, **swapping** moves them in and out to run
    - Example: Single User Systems, Windows, MAC OS

Memory Layout for OS System

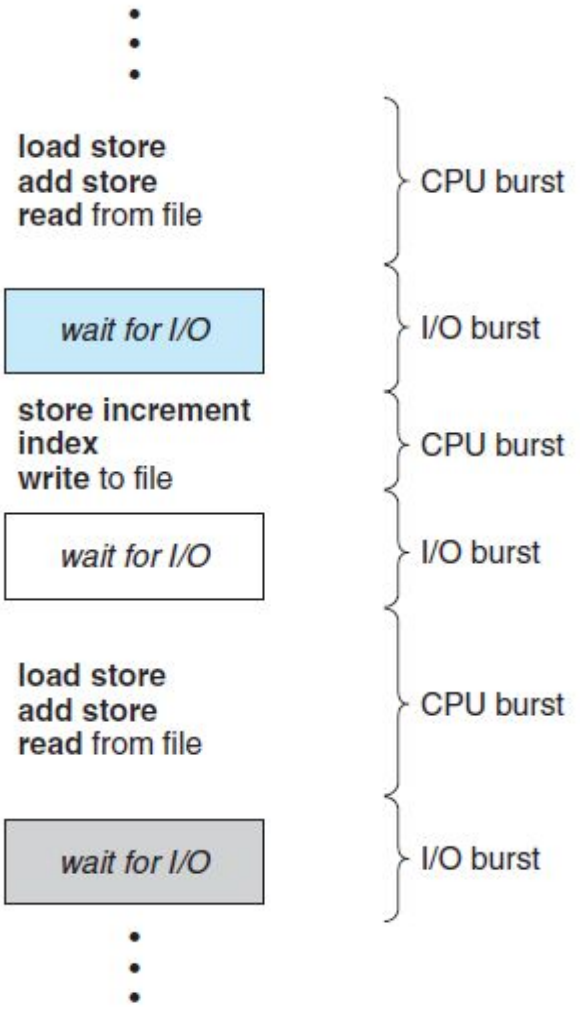


Multi Program (Batch System)
Each similar job has its own CPU and I/O bursts



Multi Process (Multi Tasking System)
Each similar process has its own CPU and I/O bursts

Process Life Cycle



Alternating sequence of CPU and I/O bursts.

Process Life Cycle

- Example: Two Processes P1 and P2.

Process 1
(5 CPU time units and 5 I/O time units)

Process 2
(5 CPU time units and 5 I/O time units)

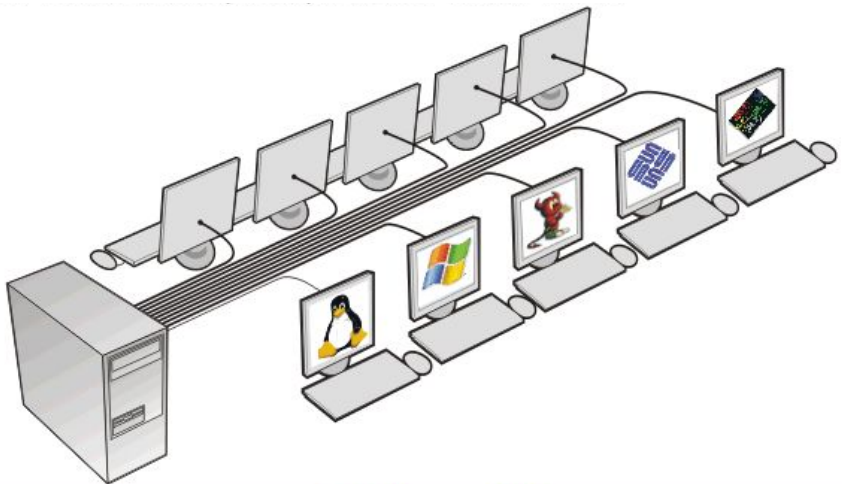
Dots: I/O Burst

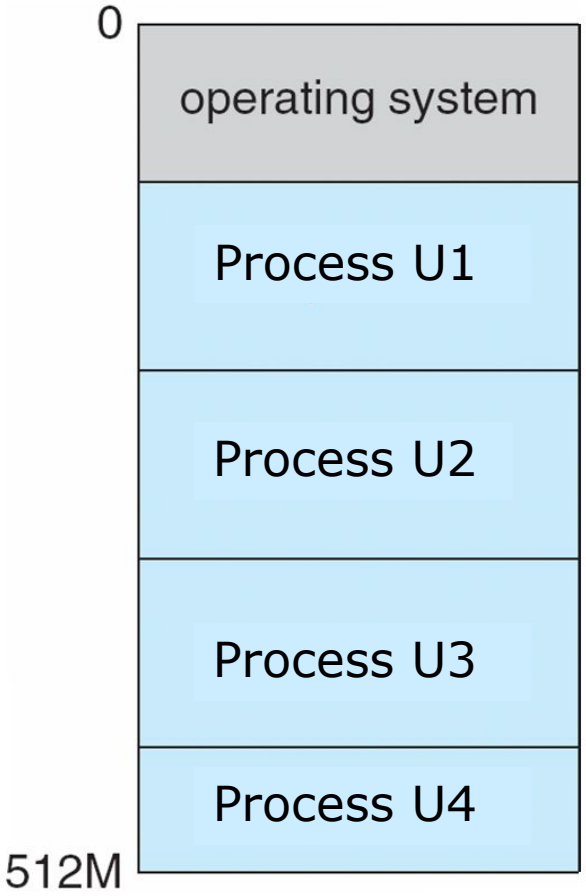
Dash: CPU Burst

Operating System Concepts

- **Timesharing** (**Multitasking and Multi users**) 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
  - A single CPU is shared among multiple users
  - A short time is assigned to each users process □ **Quantum, Slice, Slot**
  - **Virtual memory** allows execution of processes not completely in memory
  - Example: LINUX OS, UNIX OS

Operating System Concepts





Multi User (Time sharing System)
Each users similar process has its own CPU and I/O bursts

Operating System Concepts

- **Real Time OS** (**Time Critical**) gives response in real time, critical time constraints
  - **Response time** should be very fast
  - Once interrupt is given, responds abruptly
  - Real-time operating systems are **Event-driven** and **Pre-Emptive**, meaning the OS can monitor the relevant priority of competing tasks, and make changes to the task priority. Event-driven systems switch between tasks based on their priorities
  - **Hard RTOS:** Crucial applications e.g. medical applications, ventilator systems, medical imaging systems, rockets and robotics
  - **Soft RTOS:** Not crucial but real time response required e.g. Networking, sign boards

Open-Source Operating Systems

- Operating systems made available in source-code format rather than just binary **closed-source**
- Counter to the **copy protection** and **Digital Rights Management (DRM)** movement
- Started by **Free Software Foundation (FSF)**, which has "copyleft" **GNU Public License (GPL)**
- Examples include **GNU/Linux** and **BSD UNIX** (including core of **Mac OS X**), and many more
- Can use VMM like VMware Player (Free on Windows), Virtualbox (open source and free on many platforms - http://www.virtualbox.com)
  - Use to run guest operating systems for exploration

End of Chapter 1