OS-MANUAL

SAGAR CHHABRIYA (SEM-4 2024)
INSTRUCTOR: SIR NOOR NABI DAHAR (HOD-CS)

Lecture #1

What is an Operating System?
A program that acts as an intermediary between a user of a computer and the computer hardware. An operating System is a collection of system programs that together control the operations of a computer system.

Some examples of operating systems are UNIX, Mach, MS-DOS, MS-Windows, Windows/NT, Chicago, OS/2, MacOS, VMS, MVS, and VM.

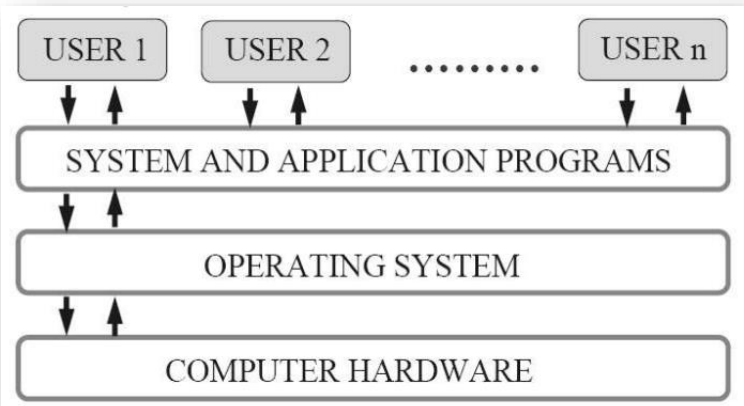
Operating system goals:

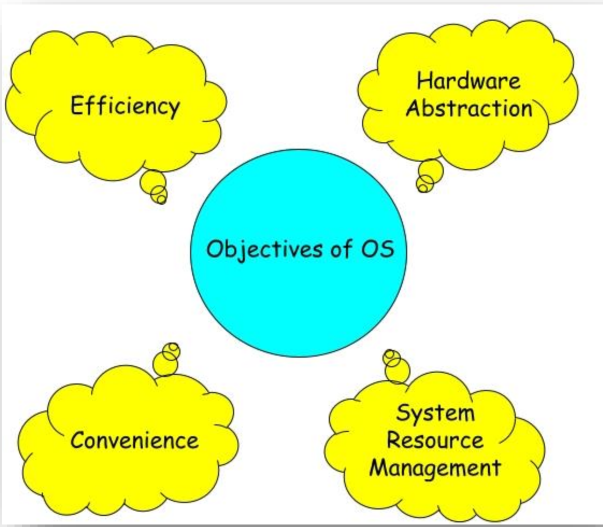
- Execute user programs and make solving user problems easier.
- Make the computer system convenient to use.
- Efficiently use the computer hardware.

Computer System Components

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

Abstract View of System Components





Operating System Definitions

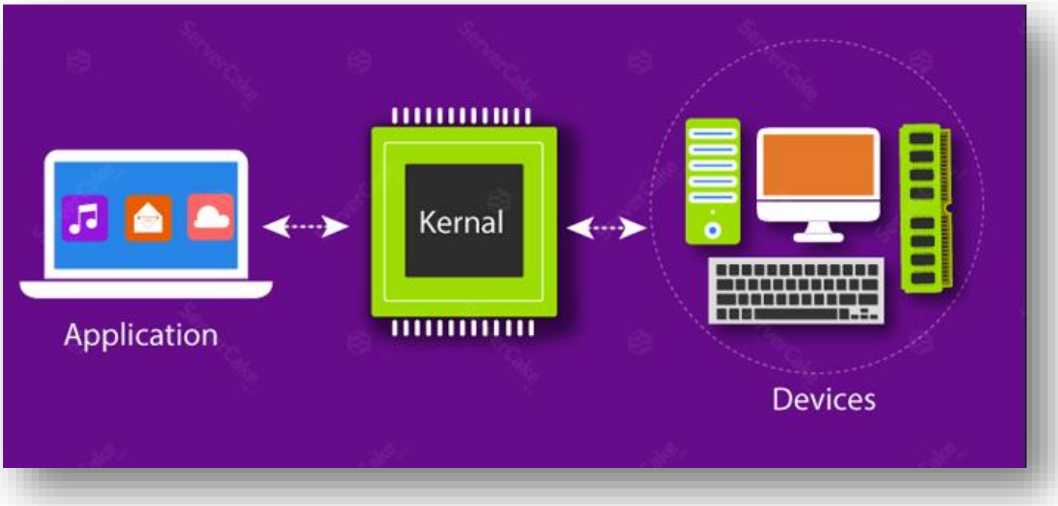
Resource allocator – manages and allocates resources.

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).

Components of OS: OS has two parts. (1) Kernel (2) Shell.

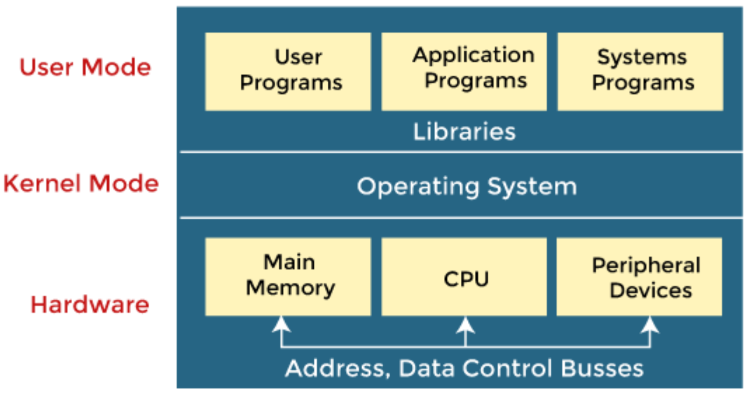
1. Kernel is an active part of an OS i.e., it is the part of OS running at all times. It is a program that can interact with the hardware. Ex: Device driver, system files, etc.
2. Shell is called as the command interpreter. It is a set of programs used to interact with the application programs. It is responsible for the execution of instructions given to the OS (called commands).

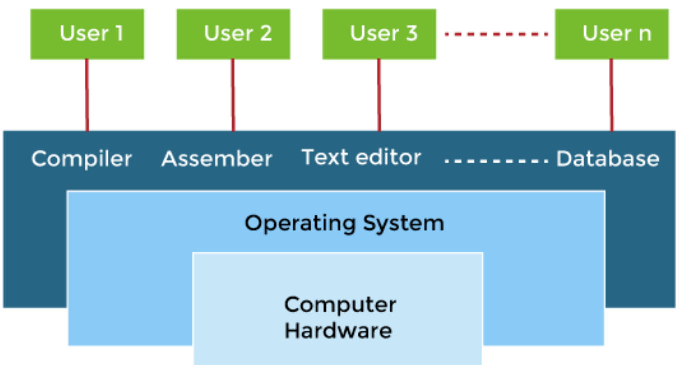


Operating systems can be explored from two viewpoints: the user and the system.

User View: From the user's point of view, the OS is designed for one user to monopolize its resources, to maximize the work that the user is performing, and for ease of use.

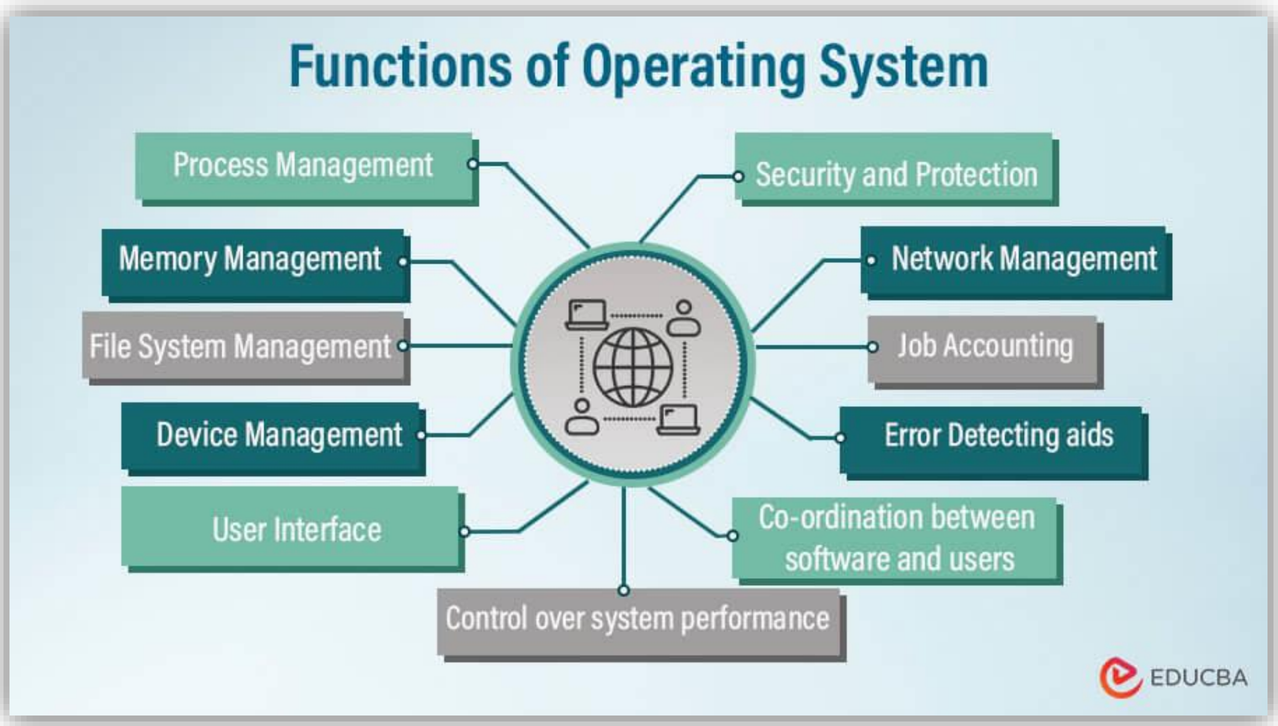
System View: From the computer's point of view, an operating system is a control program that manages the execution of user programs to prevent errors and improper use of the computer. It is concerned with the operation and control of I/O devices.





Lecture #2

Functions of Operating System:



1. ## 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.

a) Process creation and deletion.
b) Process suspension and resumption.
c) Provision of mechanisms for:
   - Process synchronization
   - Process communication

2. ## Main-Memory Management

   Memory is a large array of words or bytes, each with its address. It is 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 in the case of system failure. The operating system is responsible for the following activities in connection with memory management:

- 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 de-allocate memory space as needed.

3. 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 connection 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 (non-volatile) storage media.

4. I/O System Management

The I/O system consists of:

- A buffer-caching system
- A general device-driver interface
- Drivers for specific hardware devices

5. Secondary-Storage Management

Since main memory (primary storage) is volatile and too small to accommodate all data and programs permanently, the computer system must provide secondary storage to back up main memory. Most modern computer systems use disks as the principal online 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

6. Networking (Distributed Systems)

A distributed system is a collection of processors that do not share memory or a clock. Each processor has its local memory.

- The processors in the system are connected through a communication network.
- Communication takes place using a protocol.
- A distributed system provides user access to various system resources.
- Access to a shared resource allows:
- Computation speed-up
- Increased data availability
- Enhanced reliability

7. 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.

8. Command-Interpreter System

Many commands are given to the operating system by control statements that deal with:

- Process creation and management
- I/O handling
- Secondary-storage management
- Main-memory management
- File-system access
- Protection
- Networking

The program that reads and interprets control statements is called variously:

- Command-line interpreter
- Shell (in UNIX)

Its function is to get and execute the next command statement.

9. Operating-System Structures

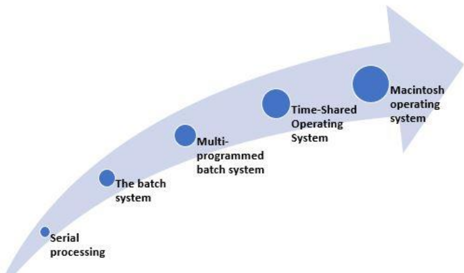
- System Components
- Operating System Services
- System Calls
- System Programs
- System Structure
- Virtual Machines
- System Design and Implementation
- System Generation

10. Common System Components

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

Lecture #3

Evolution of OS



1. Mainframe Systems

Reduce setup time by batching similar jobs Automatic job sequencing - automatically transfers control from one job to another.

- First rudimentary operating system.
- Resident monitor
- Initial control in monitor
- Control transfers to the job
- When the job is completed control transfers pack to monitor

2. Batch Processing Operating System
   This type of OS accepts more than one job and these jobs are batched/ grouped according to their similar requirements. This is done by a computer operator. Whenever the computer becomes available, the batched jobs are sent for execution and gradually the output is sent back to the user.

   - It allowed only one program at a time.
   - This OS is responsible for scheduling the jobs according to priority and the resources required.

3. Multiprogramming Operating System
   This type of OS is used to execute more than one job simultaneously by a single processor. it increases CPU utilization by organizing jobs so the CPU always has one job to execute.

   The concept of multiprogramming is described as follows:

   - All the jobs that enter the system are stored in the job pool (in disc). The operating system loads a set of jobs from the job pool into the main memory and begins to execute.
   - During execution, the job may have to wait for some task, such as an I/O operation, to complete. In a multiprogramming system, the operating system simply switches to another job and executes. When that job needs to wait, the CPU is switched to another job, and so on.
   - When the first job finishes waiting it gets the CPU back.
   - As long as at least one job needs to execute, the CPU is never idle.

Multiprogramming operating systems use the mechanism of job scheduling and CPU scheduling.

4. Multiprocessor Operating Systems
   Multiprocessor operating systems are also known as parallel OS or tightly coupled OS. Such operating systems have more than one processor in close communication that shares the computer bus, the clock, and sometimes memory and peripheral devices. It executes multiple jobs at the same time and makes the processing faster.

   Multiprocessor systems have three main advantages:

   - Increased throughput: By increasing the number of processors, the system performs more work in less time. The speed-up ratio with N processors is less than N.
   - Economy of scale: Multiprocessor systems can save more money than multiple single-processor systems because they can share peripherals, mass storage, and power supplies.
   - Increased reliability: If one processor fails to do its task, then each of the remaining processors must pick up a share of the work of the failed processor. The failure of one processor will not halt the system, only slow it down.

   The ability to continue providing service proportional to the level of surviving hardware is called graceful degradation. Systems designed for graceful degradation are called fault tolerant.

   The multiprocessor operating systems are classified into two categories:

   1. Symmetric multiprocessing system
   2. Asymmetric multiprocessing system

   - In a symmetric multiprocessing system, each processor runs an identical copy of the operating system, and these copies communicate with one another as needed.

- In an asymmetric multiprocessing system, a processor is called a master processor that controls other processors called slave processor. Thus, it establishes master-slave relationship. The master processor schedules the jobs and manages the memory for entire system.

5. Distributed Operating Systems
   In a distributed system, the different machines are connected in a network and each machine has its processor and own local memory. In this system, the operating systems on all the machines work together to manage the collective network resource.

   It can be classified into two categories:
   1. Client-server systems
   2. Peer-to-Peer systems

   Advantages of distributed systems.
   - Resources Sharing
   - Computation speed up - load sharing
   - Reliability
   - Communications
   - Requires networking infrastructure.
   - Local area networks (LAN) or Wide area networks (WAN)

6. Desktop Systems/Personal Computer Systems
   The PC operating system is designed to maximize user convenience and responsiveness. This system is neither multi-user nor multitasking.

   These systems include PCs running Microsoft Windows and the Apple Macintosh. The MS-DOS operating system from Microsoft has been superseded by multiple flavors of Microsoft Windows and IBM has upgraded MS-DOS to the OS/2 multitasking system.

   The Apple Macintosh operating system has been ported to more advanced hardware, and now includes new features such as virtual memory and multitasking.

7. Real-Time Operating Systems (RTOS)
   A real-time operating system (RTOS) is a multitasking operating system intended for applications with fixed deadlines (real-time computing). Such applications include some small embedded systems, automobile engine controllers, industrial robots, spacecraft, industrial control, and some large-scale computing systems.

   The real-time operating system can be classified into two categories:
   1. Hard real-time system
   2. Soft real-time system.

   A hard real-time system guarantees that critical tasks be completed on time. This goal requires that all delays in the system be bounded, from the retrieval of stored data to the time that it takes the operating system to finish any request made of it. Such time constraints dictate the facilities that are available in hard real-time systems.

A soft real-time system is a less restrictive type of real-time system. Here, a critical real-time task gets priority over other tasks and retains that priority until it completes. Soft real time system can be mixed with other types of systems. Due to less restriction, they are risky to use for industrial control and robotics.

Lecture #4

Operating System Services

Following are the five services provided by operating systems to the convenience of the users.

1. Program Execution
   The purpose of computer systems is to allow the user to execute programs. So the operating system provides an environment where the user can conveniently run programs. Running a program involves allocating and deallocating memory, and CPU scheduling in case of multiprocessing.
2. I/O Operations
   Each program requires an input and produces output. This involves the use of I/O. So the operating systems are providing I/O makes it convenient for the users to run programs.
3. File System Manipulation
   The output of a program may need to be written into new files or input taken from some files. The operating system provides this service.
4. Communications
   The processes need to communicate with each other to exchange information during execution. It may be between processes running on the same computer or running on different computers.

   Communication can occur in two ways:
   1. Shared memory
   2. Message Passing
5. Error Detection
   An error in one part of the system may cause the malfunctioning of the complete system. To avoid such a situation operating system constantly monitors the system for detecting errors. This relieves the user of the worry of errors propagating to various parts of the system and causing malfunctioning.

Following are the three services provided by operating systems to ensure the efficient operation of the system itself.

1. Resource allocation
   When multiple users are logged on the system or multiple jobs are running at the same time, resources must be allocated to each of them. Many different types of resources are managed by the operating system.

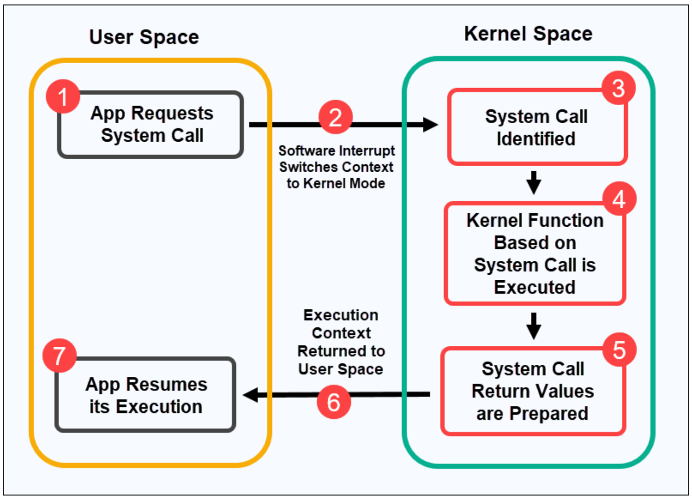
2. Accounting
   The operating systems keep track of which users use how many and which kinds of computer resources. This record-keeping may be used for accounting (so that users can be billed) or simply for accumulating usage statistics.

3. Protection
   When several disjointed processes execute concurrently, it should not be possible for one process to interfere with the others, or with the operating system itself. Protection involves ensuring that all access to system resources is controlled. Security of the system from outsiders is also important. Such security starts with each user having to authenticate him to the system, usually using a password, to be allowed access to the resources.

System Call

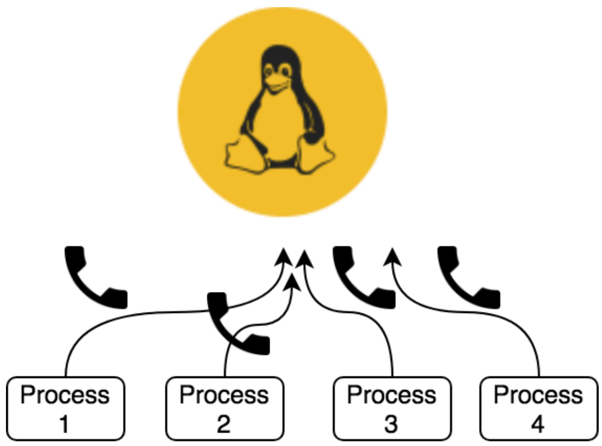
System calls provide an interface between the process and the operating system.

System calls allow user-level processes to request some services from the operating system which the process itself is not allowed to do. For example, for I/O a process involves a system call telling the operating system to read or write particular area and this request is satisfied by the operating system.



The following different types of system calls are provided by an operating system:

1. Process control
   End, abort
   Load, execute
   Create process, terminate process
   Get process attributes, set process attributes
   Wait for time
   Wait event, signal event
   Allocate and free memory



2. File management
   Create a file, delete the file
   Open, close
   Read, write, reposition
   Get file attributes, set file attributes

3. Device management
   Request device, release device
   Read, write, reposition
   Get device attributes, set device attributes
   Logically attach or detach devices

4. Information Maintenance
   Get time or date, set time or date
   Get system data, set system data
   Get process, file, or device attributes
   Set process, file, or device attributes

5. Communications
   Create, and delete communication connection
   Send, receive messages
   Transfer status information
   Attach or detach remote devices

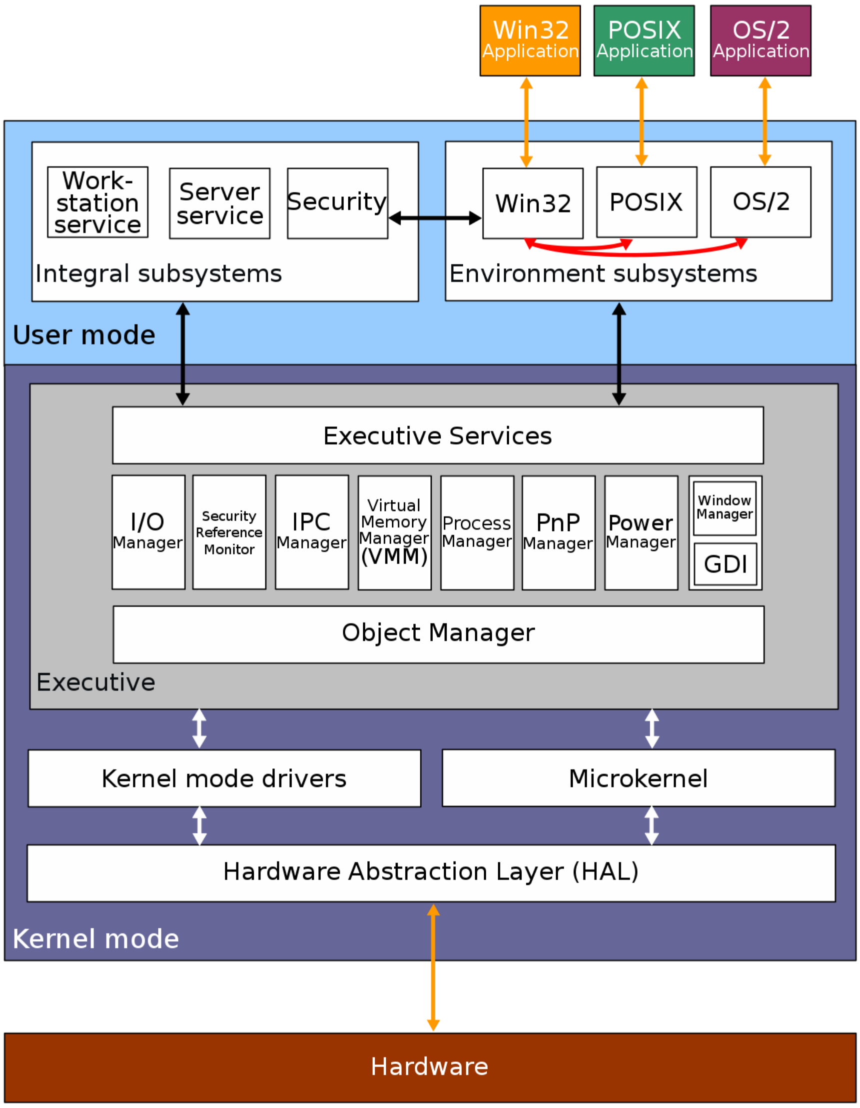
Microkernel System Structure

Moves as much from the kernel into "user" space.

Communication takes place between user modules using message passing.

Benefits:

- Easier to extend a microkernel
- Easier to port the operating system to new architectures
- More reliable (less code is running in kernel mode)
- More secure



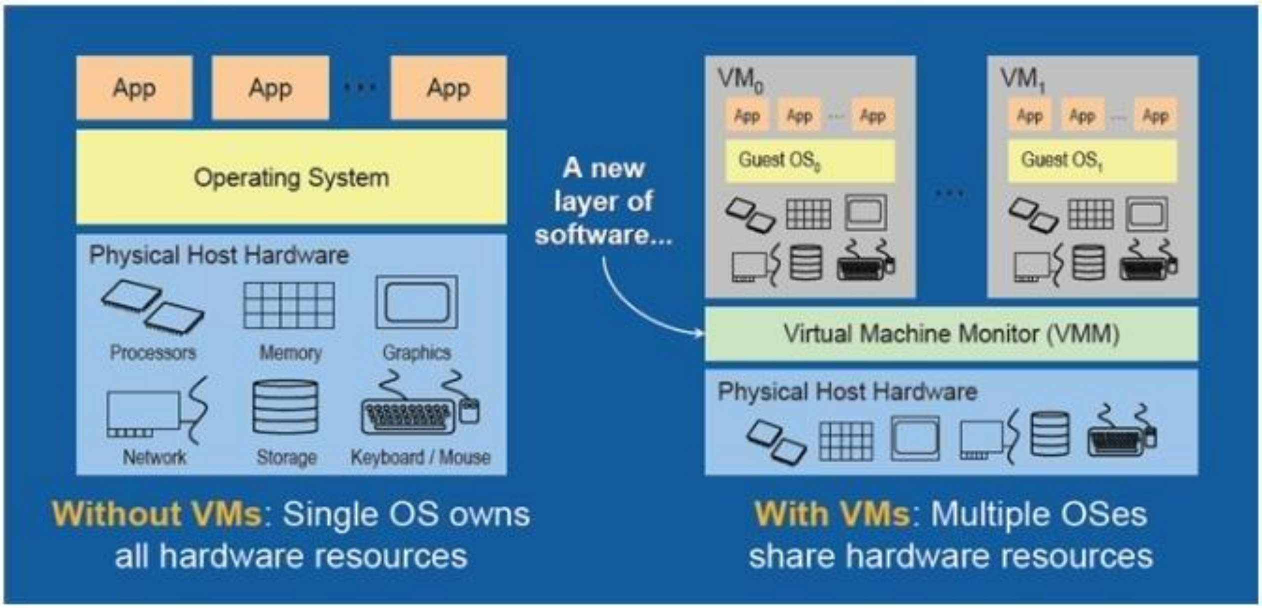
Virtual Machines

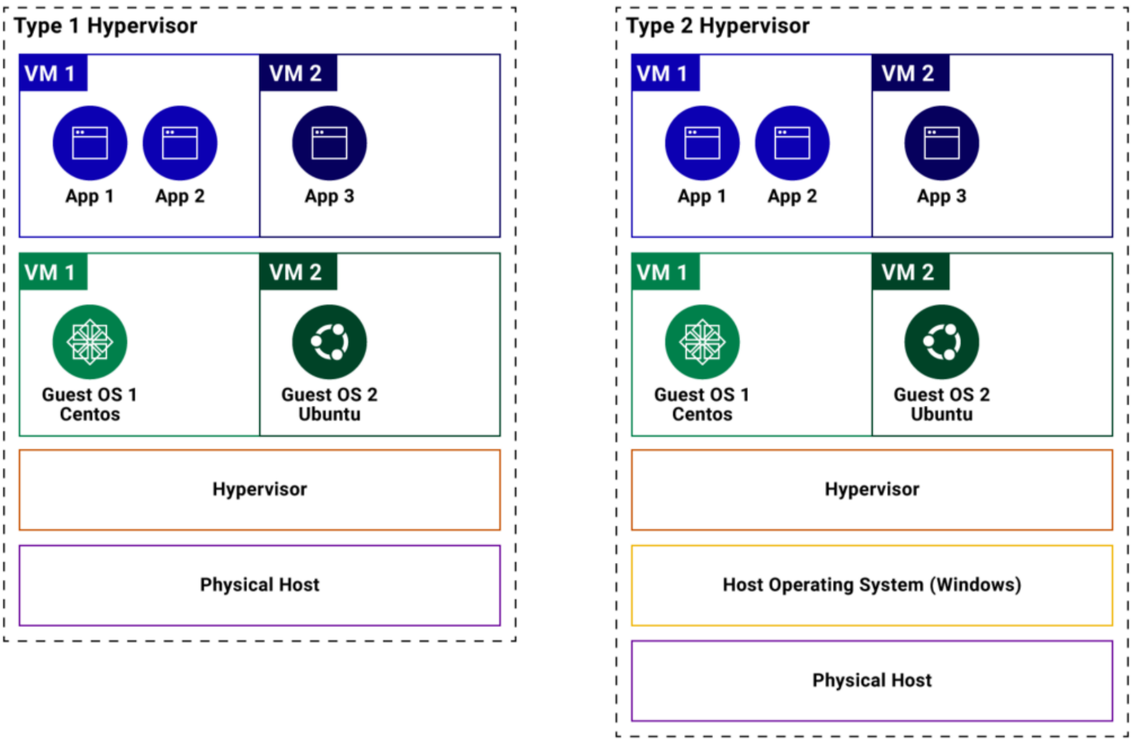
A virtual machine takes the layered approach to its logical conclusion. It treats hardware and the operating system kernel as though they were all hardware.

A virtual machine provides an interface identical to the underlying bare hardware.
The operating system creates the illusion of multiple processes, each executing on its processor with its own (virtual) memory.

The resources of the physical computer are shared to create the virtual machines.

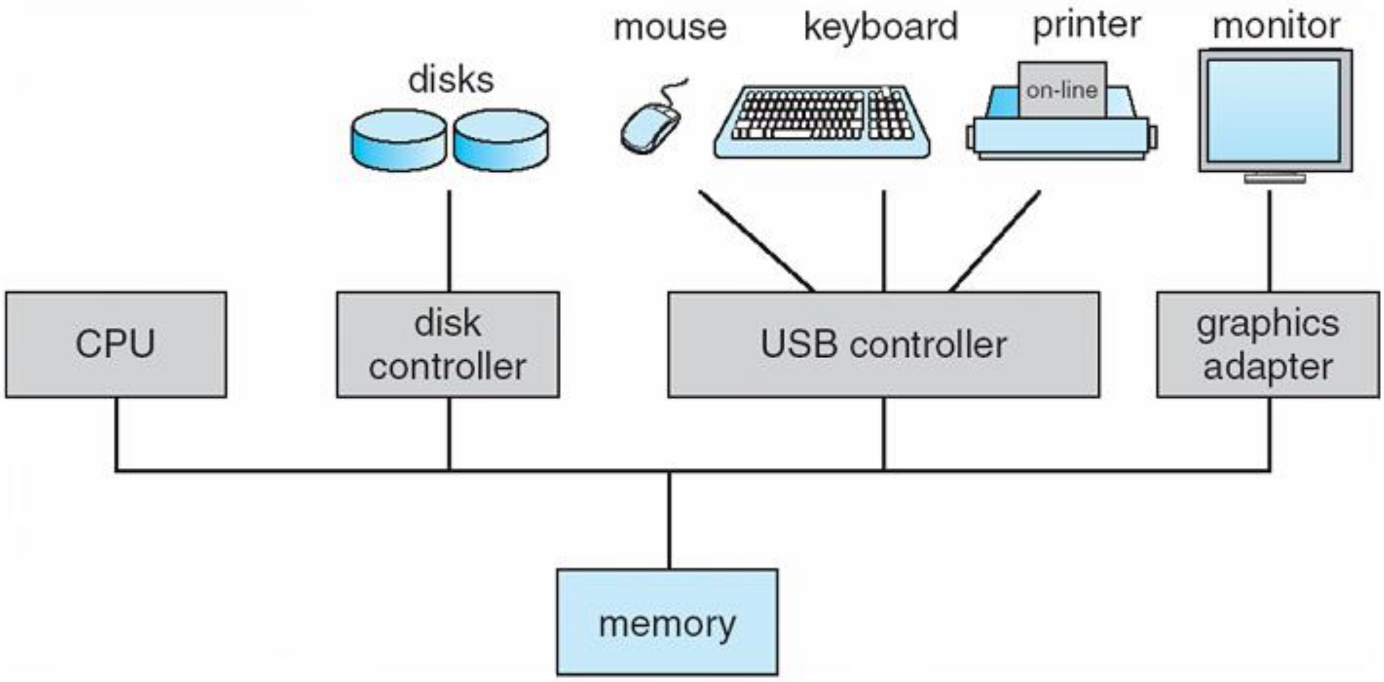
- CPU scheduling can create the appearance that users have their own processor.
- Spooling and a file system can provide virtual card readers and virtual line printers.
- A normal user time-sharing terminal serves as the virtual machine operator's console.





Lecture #5

Computer System Architecture



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 the local buffer of the controller.
- The device controller informs the 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 Time Line for a Single Process Doing Output

Direct Memory Access Structure

Used for high-speed I/O devices able to transmit information at close to memory speeds.

The 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 - an extension of main memory that provides large non-volatile storage capacity.

Magnetic disks - rigid metal or glass platters covered with magnetic recording material

- The 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.

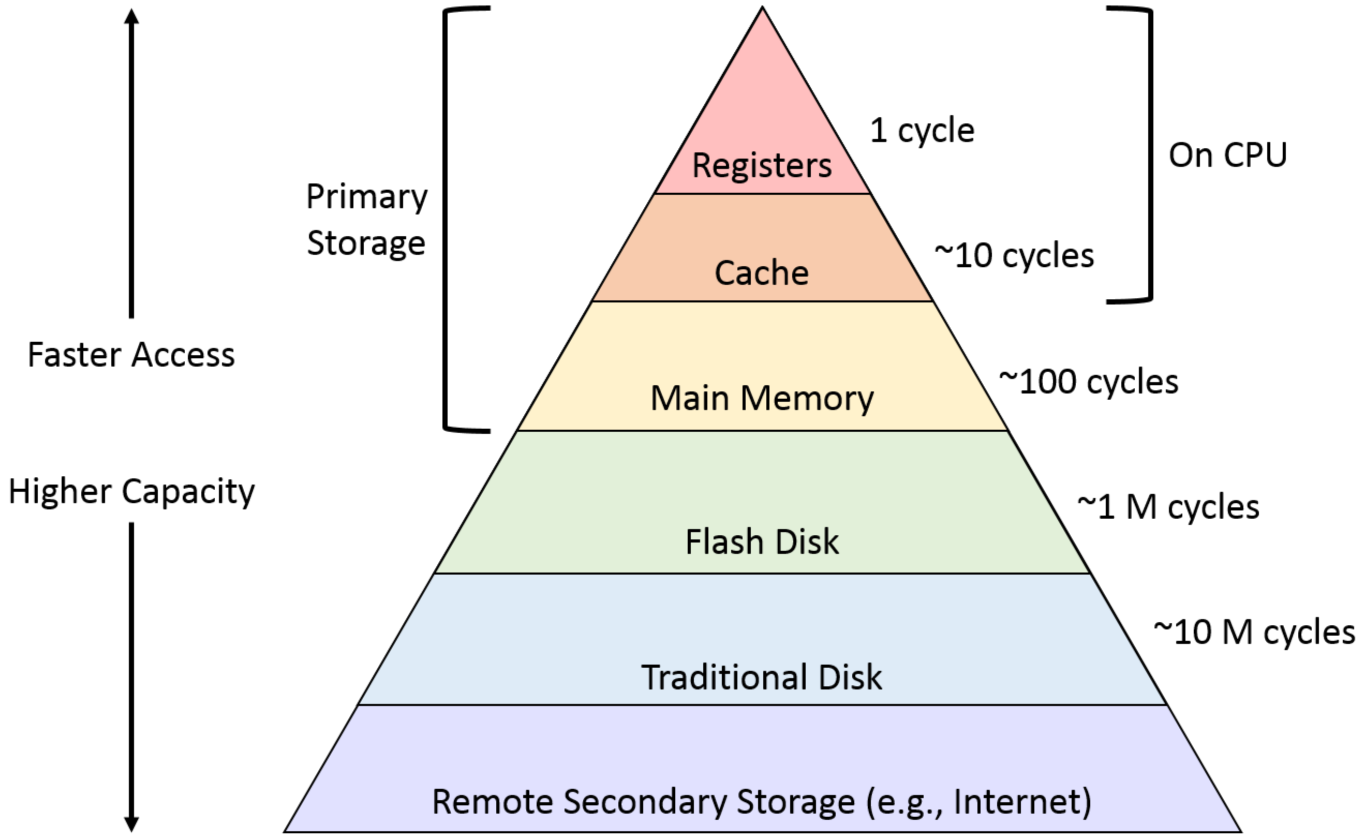
Storage Hierarchy

Storage systems are organized in a hierarchy.

- Speed
- Cost
- Volatility

Caching - copying information into a faster storage system; main memory can be viewed as a last cache for secondary storage.

- Use of high-speed memory to hold recently-accessed data.
- Requires a cache management policy.
- Caching introduces another level in storage hierarchy.
- This requires data that is simultaneously stored in more than one level to be consistent.



The Memory Hierarchy

Lecture #6

Processes Concept

Informally, a process is a program in execution. A process is more than the program code, which is sometimes known as the text section. It also includes the current activity, as represented by the value of the program counter and the contents of the processor's registers. In addition, a process generally includes the process stack, which contains temporary data (such as method parameters, return addresses, and local variables), and a data section, which contains global variables.

An operating system executes a variety of programs:

Batch system – jobs

Time-shared systems – user programs or tasks

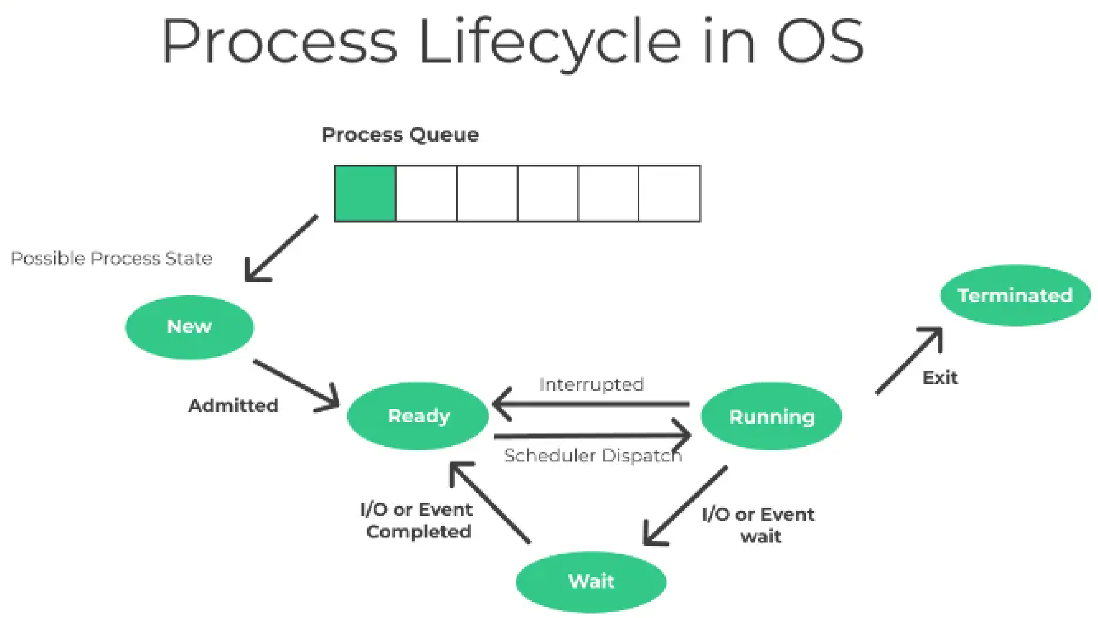
Process – a program in execution; process execution must progress sequentially.

A process includes: a program counter, stack, data section

Process State

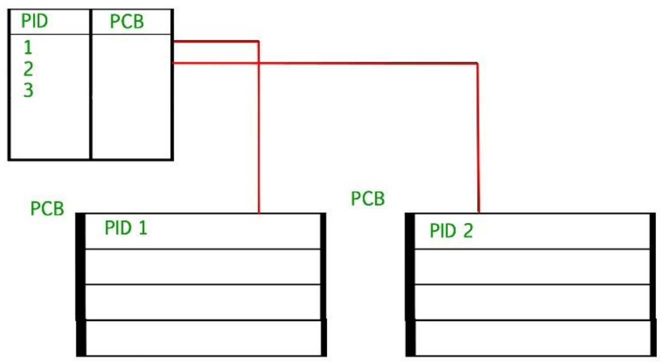
As a process executes, it changes state

- New State: The process is being created.
- Running State: A process is said to be running if it has the CPU, that is, a process using the CPU at that particular instant.
- Blocked (or waiting) State: A process is said to be blocked if it is waiting for some event to happen such that as an I/O completion before it can proceed. Note that a process is unable to run until some external event happens.
- Ready State: A process is said to be ready if it needs a CPU to execute. A ready state process is runnable but temporarily stopped running to let another process run.
- Terminated state: The process has finished execution.

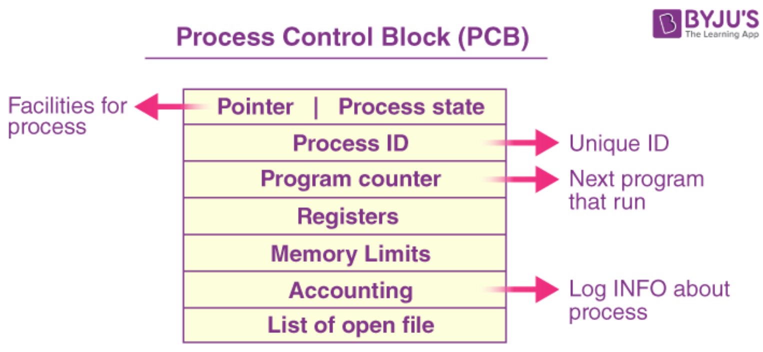


Lecture #7

Process Control Block



Process table and process control block

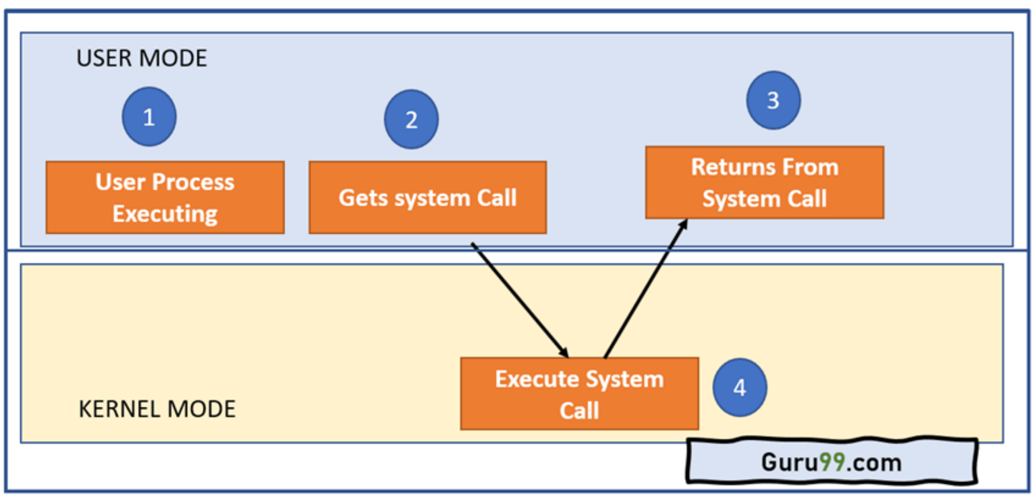


PCB Diagram

Process state: The state may be new, ready, running, and waiting, halted, and SO on.
Program counter: The counter indicates the address of the next instruction to be executed for this process.
CPU registers: The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers, and general-purpose registers, plus any condition-code information. Along with the program counter, this state information must be saved when an interrupt occurs, to allow the process to be continued correctly afterward.
CPU-scheduling information: This information includes a process priority, pointers to scheduling queues, and any other scheduling parameters.
Memory-management information: This information may include such information as the value of the base and limit registers, the page tables, or the segment tables, depending on the memory system used by the operating system.
Accounting information: This information includes the amount of CPU and real-time used, time limits, account numbers, job or process numbers, and so on.
Status information: The information includes the list of I/O devices allocated to this process, a list of open files, and so on.

The PCB simply serves as the repository for any information that may vary from process to process.

Working of System Calls



Lecture #8

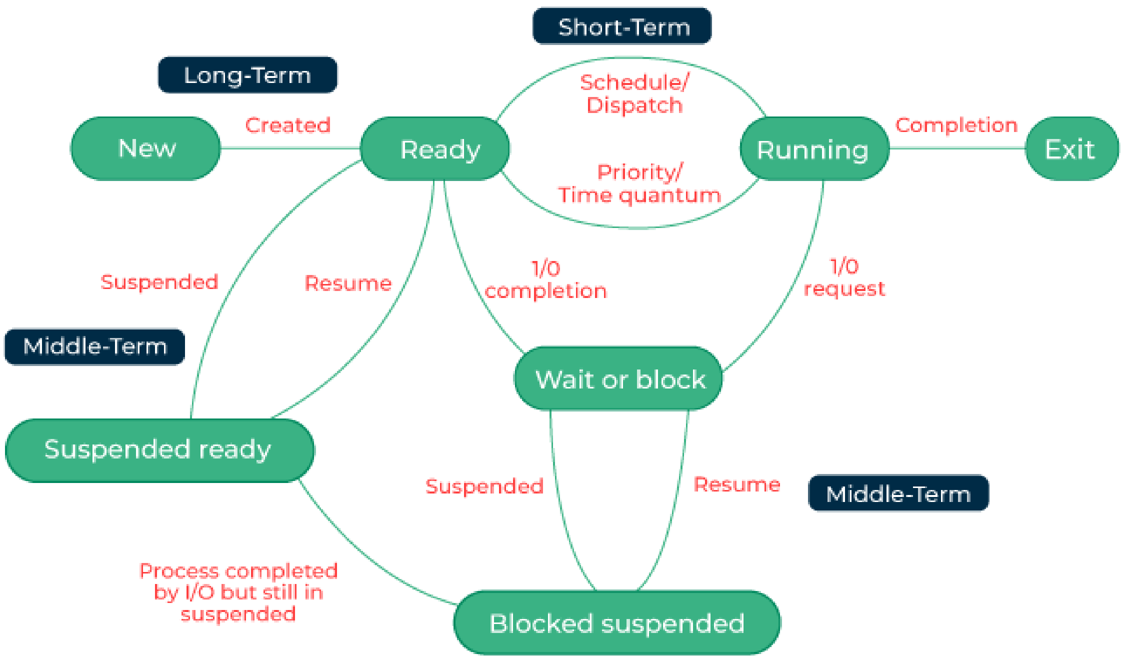
Process Scheduling Queues

Job Queue: This queue consists of all processes in the system; those processes are entered into the system as new processes.

Ready Queue: This queue consists of the processes that are residing in main memory and are ready and waiting to execute by CPU. This queue is generally stored as a linked list. A ready-queue header contains pointers to the first and final PCBs in the list. Each PCB includes a pointer field that points to the next PCB in the ready queue.

Device Queue: This queue consists of the processes that are waiting for a particular I/O device. Each device has its own device queue.

Representation of Process Scheduling



Schedulers

A scheduler is a decision maker that selects the processes from one scheduling queue to another or allocates CPU for execution.

The Operating System has three types of scheduler:

1. Long-term scheduler or Job scheduler
2. Short-term scheduler or CPU scheduler
3. Medium-term scheduler

Long-term scheduler or Job scheduler

- The long-term scheduler or job scheduler selects processes from discs and loads them into the main memory for execution. It executes much less frequently.
- It controls the degree of multiprogramming (i.e., the number of processes in memory).
- Because of the longer interval between executions, the long-term scheduler can afford to take more time to select a process for execution.

Short-term scheduler or CPU scheduler

The short-term scheduler or CPU scheduler selects a process from among the processes that are ready to execute and allocates the CPU.

The short-term scheduler must select a new process for the CPU frequently. A process may execute for only a few milliseconds before waiting for an I/O request.

Medium-term scheduler

The medium-term scheduler schedules the processes as the intermediate level of scheduling
Processes can be described as either:

- I/O-bound process - spends more time doing I/O than computations, many short CPU bursts.
- CPU-bound process - spends more time doing computations; few very long CPU bursts.

Context Switch

When the CPU switches to another process, the system must save the state of the old process and load the saved state for the new process.
Context-switch time is overhead; the system does no useful work while switching.
Time-dependent on hardware support.

Process Creation
Parent processes create children's processes, which, in turn, create other processes, forming a tree of processes.
Resource sharing:

- Parents and children share all resources.
- Children share a subset of parent's resources.
- Parent and child share no resources.

Execution:

- Parents and children execute concurrently.
- Parent waits until their children terminate.

Address space:

- Child duplicate of the parent.
- The child has a program loaded into it.

UNIX examples

- Fork system call creates a new process
- Exec system call used after a fork to replace the process' memory space with a new program.

Process Termination

The process executes the last statement and asks the operating system to decide it (exit).
Output data from child to parent (via wait).

Process resources are de-allocated by operating system:

- Parent may terminate execution of children processes (abort).

Child has exceeded allocated resources.
Task assigned to child is no longer required.

Parent is exiting:

- Operating system does not allow child to continue if its parent terminates.
- Cascading termination.

Lecture #9

CPU Scheduling Basics and Algorithms

- CPU Scheduling
- Basic Concepts
- Scheduling Criteria
- Scheduling Algorithms
- Multiple-Processor Scheduling
- Real-Time Scheduling
- Algorithm Evaluation

Basic Concepts

- Maximum CPU utilization obtained with multiprogramming
- CPU–I/O Burst Cycle – Process execution consists of a cycle of CPU execution and I/O wait.
- CPU burst distribution

Alternating Sequence of CPU and I/O Burst