Rajesh Parajuli

+9779847546279

Operating System

Syllabus

Assigments

Assignment 1
Assignment 2
Assignment 3
Assignment 4

Study Materials

Unit 1
Unit 2
Unit 3
Unit 4
Scheduling algorithms

Exam old Questions

Old questions 2081

Operating System Note Syllabus Covered

Unit 1 Introduction

Definition of Operating System?
Two Views of Operating System
What Operating System do?
Operating System Services
Operating System Structure
Simple Structure
Monolithic Structure
Layered Structure
Microkernel Structure
Modular Structure
System Calls

Unit 2 Process Management

Process and Thread Concept
Interprocess Communication
Process State
Process Synchronization
Critical Section Problem
Peterson's Solution
Semaphores
Mutex Locks
Monitors
CPU Scheduling Concepts
First come first serve (FCFS)
Shortest Job First / Shortest Job Next
Round Robin Scheduling Algorithm
Shortest Job Remaining First
Deadlock

Banker's Algorithm for Deadlock Avoidance

Unit 3 Memory Management

Main Memory Management
Swapping
Memory Allocation Strategies
Paging and its Types
Structure of the page Table
Segmentation
Virtual Memory Management
Page Replacement Algorithms

Unit 4 Storage Management

Disk Structure
Disk Scheduling algorithm
RAID Structure
File Concept and Access Methods
Directory Structure
Directory Implementation
File System Structure and operations
Allocation Methods
Free Space Management

Unit 1

Basic of Software

Software is a set of instructions, data or programs used to operate computers and execute specific tasks. There are Three types of Software:

System Software: System software is computer software designed to operate the computer hardware and to provide a platform for running application software. System software also provides services to computer users and application programs. Examples:- Windows OS, Linux/Unix OS, Andriod OS, MAC OS … etc.

Application Software: Word Processing, Microsoft Excel, Internet Explorer … etc.

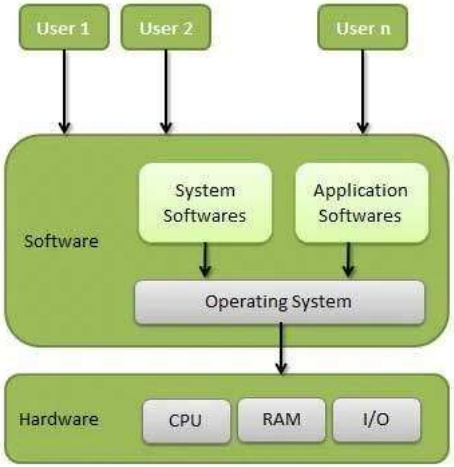
Utility Software: Utility Software refers to a type of system software that is designed to help analyse, configure, optimise and maintain a computer. Anitvius, Scanning, Disk cleanup tools… etc

Operating System Concepts

Unit 1 (PDF)

Operating system is a system software that acts as an interface between computer hardware and Computer Users and controls the execution of all kinds of programs

It provides an environment to the user so that, the user can perform its task in convenient and efficient way. Operating System is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks.



An operating system is a software that controls the internal activities of the computer hardware and provides user interface.

Two views of Operating System:

1. User's View: The user's view of the computer varies according to the interface being used. Most computer users sit in front of a personal computer(PC), consisting of a monitor, keyboard, mouse and system unit. Such a system is designed for one user to monopolize its resources. The goal is to maximize the work(or play)that the user is performing. In this case, the operating system is designed mostly for ease of use, with some attention paid to performance and none paid to resource utilization- how various hardware and software resources are shared. Performance is, of course, important to the user; but such systems are optimized for the single user experience rather than the requirements of multiple users.

In other cases, a user sits at a terminal connected to a mainframe or a minicomputer. Other users are accessing the same computer through other terminals. These users share resources and may exchange information. The operating system in some cases is designed to maximize resource utilization- to assure that available CPU time, memory and input/output are used efficiently and that no individual users takes more than her fair share.

In still other cases, users sits at workstations connected to networks of other workstations and servers. The users have dedicated resources at their disposal, but they also share resources such as networking and servers, including file, compute and print servers. Therefore, their operating system is designed to compromise between individual usability and resource utilization.

Recently, many varieties of mobile computers such as smartphones and tablets, have come into fashion. Most mobile computers are standalone units for individual users. Quite often, they are connected to networks through cellular or other wireless technologies. Increasingly, these mobile devices are replacing desktop and laptop computer for people who primarily interested in using computers for emails and web browsing.The user interface for mobile computers generally features a touch screen, where the user interacts with the system by pressing and swiping fingers across the screen rather than using a physical keyboard or mouse.

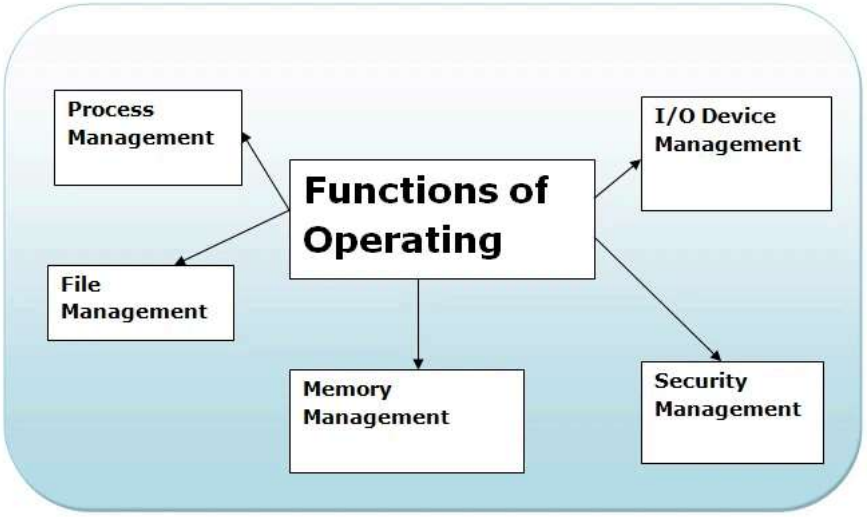
Some computers have little or no user view. For example, embedded computers in home devices and automobiles may have numeric keypads and may turn indicator lights on or off to show status, but they and their operating systems are designed primarily to run without user intervention.

2.System View :

Operating system can be viewed as a resource allocator also. A computer system consists of many resources like -hardware and software – that must be managed efficiently. The operating system acts as the manager of the resources, decides between conflicting requests, controls execution of programs etc.

From the computer's point of view, the operating system is the program most intimately involved with the hardware. In this context, we can view and operating system as a resource allocator. A computer system has many resources that may be required to solve a problem: input/output devices, CPU time, memory space, file storage space and so on. The operating system acts as the manager of this resources. Facing numerous and possibly conflicting requests for resources, the operating system must decide how to allocate them to specific programs and users so that it can operate the computer efficiently and fairly. Resource allocation is especially important where many users access the same mainframe and minicomputer. A slightly different view of an operating system emphasizes the need to control the various I/O devices and user programs. An operating system is a control program. A control program manages the execution of user programs to prevent errors and improper use of the computer. It is especially concerned with the operation and control of input/output devices

Functions / Operations of an Operating System



What does an operating System Do?

- Process Management
- I/O Device Management
- File Management
- Network Management
- Main Memory Management
- Secondary Storage Management
- Security Management
- Command Interpreter
- System Control over system
- performance Job Accounting
- Error Detection and Correction
- Coordination between other software and users
- Many more other important tasks

1. Resource management: Operating systems manage the computer's resources, such as its memory, processor, and storage, and allocate them to different tasks as needed.

2. Memory management: Operating systems manage the computer's memory and ensure that each program or process has access to the memory it needs to run.

3. Process management: Operating systems create and manage processes, which are units of work executed by the computer.

4. File management: Operating systems manage the files on the computer, including organizing them and providing access for different programs and users.
5. Security: Operating systems include security features to protect the computer from unauthorized access and viruses.
6. User interface: Operating systems provide an interface for users to interact with the computer, such as through a graphical user interface (GUI) or command-line interface (CLI).
7. Networking: Many operating systems include support for networking, allowing the computer to communicate and exchange data with other devices over a network, such as the internet or a local area network (LAN).
8. Device management: Operating systems manage the devices connected to the computer, such as printers, keyboards, and storage devices.
9. Power management: Operating systems include features to manage the power usage of the computer and conserve energy when possible.
10. Software installation and updates: Operating systems provide a mechanism for installing and updating software applications.

Operating System services

An Operating System offers services to both the users and as well as to the programs.

Operating System offers programs with an environment to execute.

Operating System offers users the services to run and execute the programs conveniently.

Some Common Services provided by an Operating System are:

1. Program Execution
2. I/O Operations
3. File System Manipulation
4. Communication
5. Error Detection
6. Resource Allocation
7. Protection

Program Execution: To execute a program, several tasks need to be performed. Both the instructions and data must be loaded into the main memory. In addition, input-output devices and files should be initialized, and other resources must be prepared. The Operating structures handle these kinds of tasks.

Control I/O Operations: I/O operation intends to read or write operation with any file or any particular I/O(input or output) device. The OS offers access to the particular I/O device when asked for. Each I/O devices have their own features. Whenever it is needed Os suggest to the user programmer to use this and get access to instruct or get results from the system.

File System Manipulation: A file represents a collection of related information. Computers can store files on the disk (secondary storage), for long-term storage purpose. Examples of storage media include magnetic tape, magnetic disk and optical disk drives like CD, DVD. Each of these media has its own properties like speed, capacity, data transfer rate and data access methods

.

A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions. Following are the major activities of an operating system with respect to file management −

- Program needs to read a file or write a file.
- The operating system gives the permission to the program for operation on file.
- Permission varies from read-only, read-write, denied and so on.
- Operating System provides an interface to the user to create/delete files.
- Operating System provides an interface to the user to create/delete directories.
- Operating System provides an interface to create the backup of file system.

Communication: In case of distributed systems which are a collection of processors that do not share memory, peripheral devices, or a clock, the operating system manages communications between all the processes. Multiple processes communicate with one another through communication lines in the network.

The OS handles routing and connection strategies, and the problems of security.

Following are the major activities of an operating system with respect to communication −

- Two processes often require data to be transferred between them.
- Both the processes can be on one computer or on different computers, but are connected through a computer network.
- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

Error Handling: Errors can occur anytime and anywhere. An error may occur in CPU, in I/O devices or in the memory hardware. Following are the major activities of an operating system with respect to error handling −

- The OS constantly checks for possible errors.
- The OS takes an appropriate action to ensure correct and consistent computing.

Resource Management: In case of multi-user or multi-tasking environment, resources such as main memory, CPU cycles and files storage are to be allocated to each user or job. Following are the major activities of an operating system with respect to resource management −

- The OS manages all kinds of resources using schedulers.
- CPU scheduling algorithms are used for better utilization of CPU.

Protection:

Considering a computer system having multiple users and concurrent execution of multiple processes, the various processes must be protected from each other's activities.

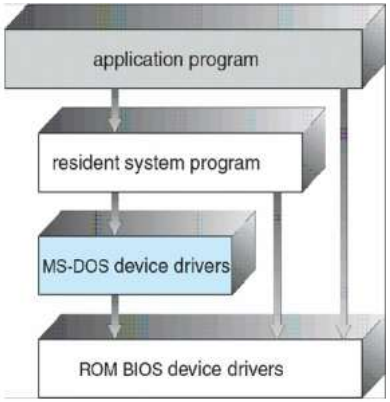
Protection refers to a mechanism or a way to control the access of programs, processes, or users to the resources defined by a computer system. Following are the major activities of an operating system with respect to protection –

- The OS ensures that all access to system resources is controlled.
- The OS ensures that external I/O devices are protected from invalid access attempts.
- The OS provides authentication features for each user by means of passwords.

Structure of Operating System

1. Simple Structure
2. Layered Structure
3. Micro kernel
4. Modular Structure

1.Simple Structure

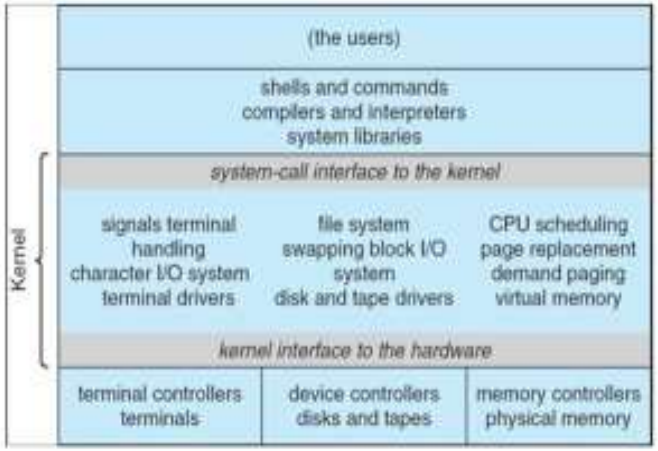


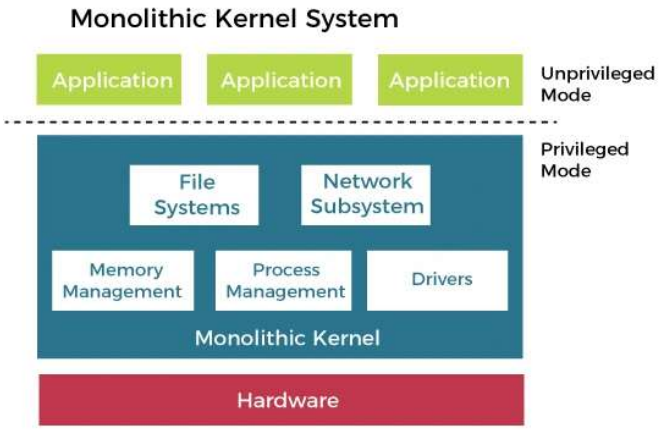
MS-DOS – written to provide the most functionality in the least space • Not divided into modules • Although MS-DOS has some structure, its interfaces and levels of functionality are not well separated– e.g. application programs can access low level drivers directly . ▯ Vulnerable to errant (malicious) programs. If one user program fails, the entire operating system gets crashed.

2.Monolithic Structure

Monolithic structure is the Unix Operating System based Approach. The monolithic operating system is a very basic operating system in which file management, memory management, device management, and process management are directly controlled within the kernel. The kernel can access all the resources present in the system. In monolithic systems, each component of the operating system is contained within the kernel. Operating systems that use monolithic architecture were first time used in the 1970s.

The monolithic operating system is also known as the monolithic kernel. This is an old operating system used to perform small tasks like batch processing and time-sharing tasks in banks. The monolithic kernel acts as a virtual machine that controls all hardware parts.





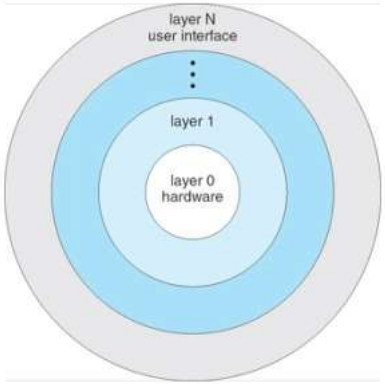
- It is simple to design and implement because all operations are managed by kernel only, and layering is not needed.
- As services such as memory management, file management, process scheduling, etc., are implemented in the same address space, the execution of the monolithic kernel is relatively fast as compared to normal systems. Using the same address saves time for address allocation for new processes and makes it faster.
- Simple design and implementation.

Disadvantages of Monolithic structure:

- If any service in the monolithic kernel fails, the entire System fails because, in address space, the services are connected to each other and affect each other.
- Lack of modularity makes maintenance and extensions difficult.
- It is not flexible, and to introduce a new service

3.Layered Structure

In this type of structure, OS is divided into layers or levels. The hardware is on the bottom layer (layer 0), while the user interface is on the top layer (layer N). These layers are arranged in a hierarchical way in which the top-level layers use the functionalities of their lower-level levels.



The following are some of the key characteristics of a layered operating system structure:

- Each layer is responsible for a specific set of tasks. This makes it easier to understand, develop, and maintain the operating system.
- Layers are typically arranged in a hierarchy. This means that each layer can only use the services provided by the layers below it.
- Layers are independent of each other. This means that a change to one layer should not affect the other layers.

Advantages of Layered Structure

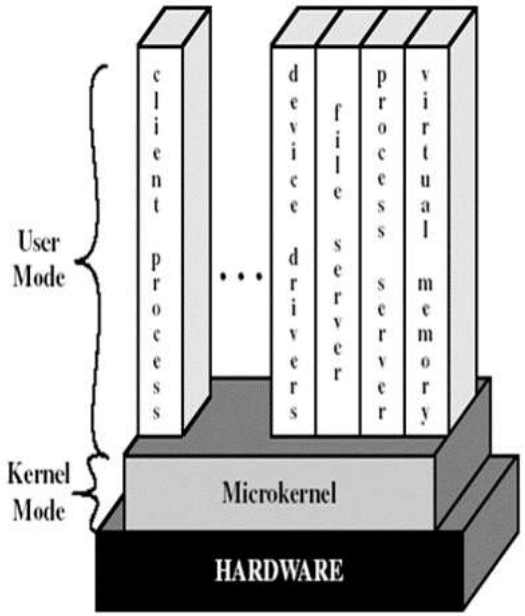
- A layered structure is highly modular, meaning that each layer is responsible for a specific set of tasks. This makes it easier to understand, develop, and maintain the operating system.
- Each layer has its functionalities, so work tasks are isolated, and abstraction is present up to some level.
- Debugging is easier as lower layers are debugged, and then upper layers are checked.

Disadvantages of Layered Structure

- In Layered Structure, layering causes degradation in performance.
- It takes careful planning to construct the layers since higher layers only utilize the functions of lower layers.
- There can be some performance overhead associated with the communication between layers. This is because each layer must pass data to the layer above it.

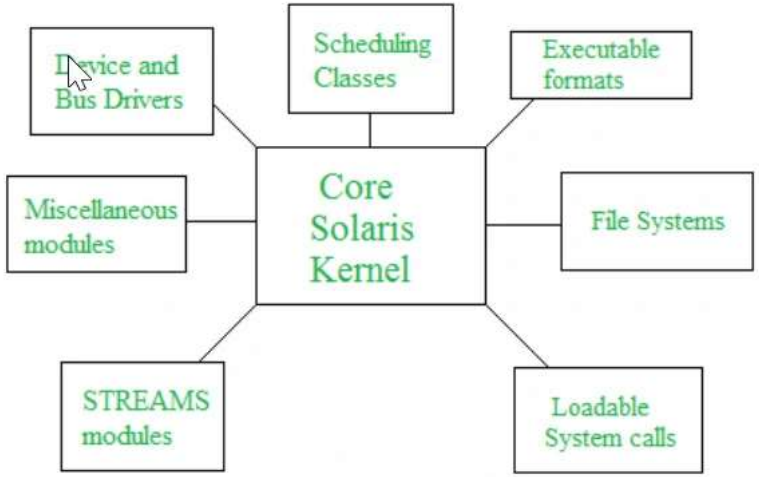
4. Microkernel Structure

This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. This results in a smaller kernel called the micro-kernel. Advantages of this structure are that all new services need to be added to user space and does not require the kernel to be modified. Thus it is more secure and reliable as if a service fails, then rest of the operating system remains untouched. Mac OS is an example of this type of OS.



5.Modular Structure

In a modular operating system structure, the operating system is divided into a set of independent modules. Each module is responsible for a specific task, such as memory management, process scheduling, or device drivers. Modules can be loaded and unloaded dynamically, as needed.



Advantages of Modular Structure

- A modular structure is highly modular, meaning that each module is independent of the others. This makes it easier to understand, develop, and maintain the operating system.
- A modular structure is very flexible. New modules can be added easily, and existing modules can be modified or removed without affecting the rest of the operating system.

Disadvantages of Modular Structure

- There can be some performance overhead associated with the communication between modules. This is because modules must communicate with each other through well-defined interfaces.
- A modular structure can be more complex than other types of operating system structures. This is because the modules must be carefully designed to ensure that they interact correctly.

System Calls

System calls provide an interface between user programs and operating system(kernel). System call is a programmatic way in which a computer program request a services from the kernel of the operating system. A system call is a method of interacting with the operating system via programs. A system call is a request from computer software to an operating system's kernel.

The kernel system can only be accessed using system calls. System calls are required for any programs that use resources.

Why do you need system calls in Operating System?

There are various situations where you must require system calls in the operating system. Following of the situations are as follows:

1. It is must require when a file system wants to create or delete a file.
2. Network connections require the system calls to sending and receiving data packets.
3. If you want to read or write a file, you need to system calls.
4. If you want to access hardware devices, including a printer, scanner, you need a system call.
5. System calls are used to create and manage new processes.

Types of System Calls

There are commonly five types of system calls.

Process Control

Process control is the system call that is used to direct the processes. Some process control examples include creating, load, abort, end, execute, process, terminate the process, etc.

File Management

File management is a system call that is used to handle the files. Some file management examples include creating files, delete files, open, close, read, write, etc.

Device Management

Device management is a system call that is used to deal with devices. Some examples of device management include read, device, write, get device attributes, release device, etc.

Information Maintenance

Information maintenance is a system call that is used to maintain information. There are some examples of information maintenance, including getting system data, set time or date, get time or date, set system data, etc.

Communication

Communication is a system call that is used for communication. There are some examples of communication, including create, delete communication connections, send, receive messages, etc.

Examples of Windows and Unix system calls

Objectives of Operating System

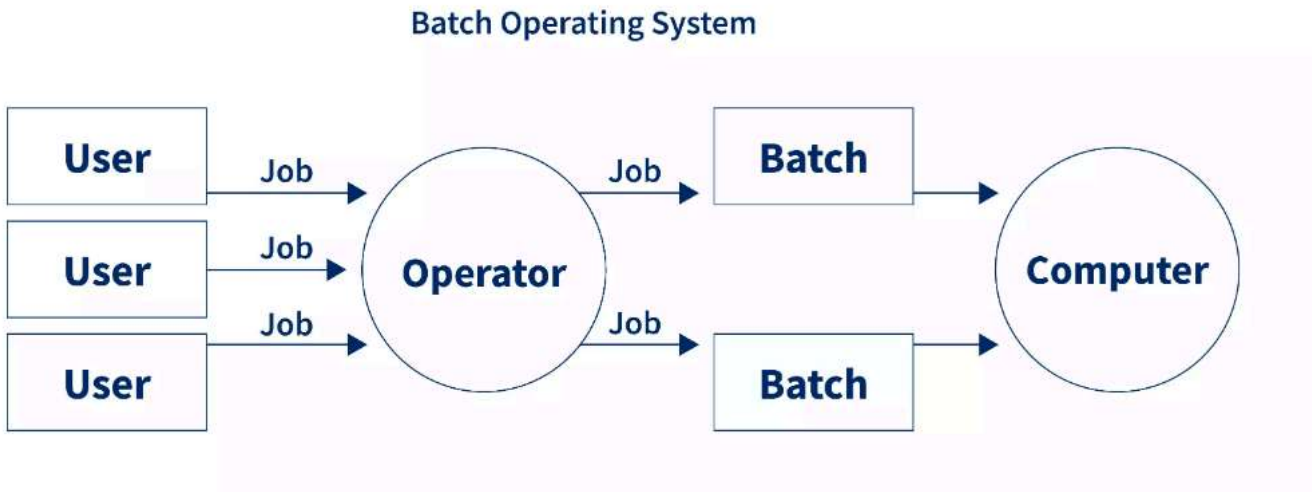
There are various examples of Windows and Unix system calls. These are as listed below in the table:

- **Convenience**: It makes a computer more suitable to use.
- **Efficiency**: It provides the computer system resources with efficiency and in easy to use format.
- **Ability to develop**: It should be built in such a way that it permits the efficient development, testing, and installation of new system functions without interfering with service.

Process	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	Fork() Exit() Wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	Open() Read() Write() Close()
Device Management	SetConsoleMode() ReadConsole() WriteConsole()	Ioctl() Read() Write()

Types of Operating System

1.Batch Operating System: Batch operating system does not interact with computer directly. Operators takes all the jobs and groups similar jobs into batches with the help of some operator and these batchee are exaguted one by one. Various Mediums are used such as Punch Cards, Paper Tape and Magnetic Tape.



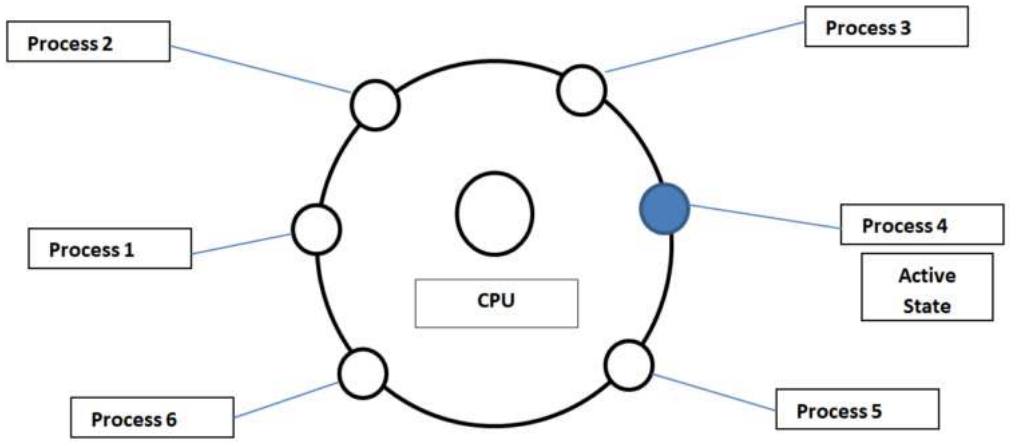
Unit 1 (PDF)

2.Time Sharing Operatiing System/ Multi-tasking Operating System: Time sharing operating system allows the user to perform more than one task at a time. Each task getting the same amount of time to execute.

Multiple jobs are running at the CPU time and also, they use the CPU simultaneously. But, the switching between the tasks is very fast due to which the user feels that all tasks are running at the same time.

Working of Time Sharing Operating System

- It is the division of CPU time for each process when more than one task are given by the user.
- A short duration of time is chosen for each process. Moreover, this time duration is very small in the order of 10-100 milliseconds. This time duration is known as **time slot, time slice, or quantum**.
- Suppose three processes namely, P1, P2, and P3 are running on the system. Now, suppose that the quantum is 4 nanoseconds (ns). Now, they will execute in the following manner.
- Process P1 will execute for 4 ns, as soon a sit gets over, process p2 starts executing for a duration of 4 ns. Further, when p2 gets complete process P3 executes for 4ns. This process continues till all the processes gets complete.
- In this way, only one process runs at a time but, the switching between the processes is very fast. Hence, the user feels that all the processes are running at the same time.



The above diagram shows the working of a time sharing OS. In this diagram, process 4 is in the **active state**. Process 5 is in a **ready state** while, processes 1, 2, 3, and 6 are in a **waiting state.**

- **Active State:** This is the state of the process which is currently processing on the CPU. Only, one process is in the active state at a time.
- **Ready State:** In this state, the process is ready for execution but, it is waiting for its turn to use the CPU. More than one process can be in the ready state at a time.
- **Waiting State: I**n this state, the process is not ready for execution. It is waiting for some input/output process to get complete.

What is quantum?

The short duration of time which is decided for each process in a time sharing OS is called quantum. We can also call it as **time slot** or **time slice.**

The real-time operating systems are used in real-time systems where the time constraints are fixed and followed strictly. This means that the time for processing and responding is very small. Moreover, the system should perform the given task in a fixed time otherwise, it results in a system failure.

Response Time is the time within which the system takes the input, processes the data, and gives the results. Moreover, they are used in systems like robots, missile launches, airplanes, etc.

Types of Real Time OS

There are three types of real-time operating systems. They are as follows:

1. Hard Real-Time Systems

In this, the time constraint is very short and strict. Even seconds of delay is not acceptable. Therefore, it is compulsory to complete the task within the given time only.

Examples are Airplanes systems, Medical treatment systems, etc.

2. Firm Real-Time Systems

In these systems, although the deadline is given but, missing them does not result in great loss. There can be some unwanted side effects in the system if the deadline is not followed.

Examples are multimedia systems.

3. Soft Real-Time Systems

As the name suggests, the system handles the deadlines softly. This means that if there are small delays in the system, it is acceptable.

Examples are Online Transaction systems, Computer games, etc.

3.Multi programming:

Multiprogramming OS is an ability of an operating system that executes more than one program using a single processor machine.

More than one task or program or jobs are present inside the main memory at one point of time.

Buffering and spooling can overlap I/O and CPU tasks to improve the system performance but it has some limitations that a single user cannot always keep CPU or I/O busy all the time.

Working of Multiprogramming operating System:

The OS could pick and start the execution of one of the jobs in memory, whenever the jobs does not need CPU that means the job is working with I/O at that time the CPU is idle at that time the OS switches to another job in memory and CPU executes a portion of it till the job issues a request for I/O and so on.

Let's P1 and P2 are two programs present in the main memory. The OS picks one program and starts executing it.

During execution if the P1 program requires I/O operation, then the OS will simply switch over to P2 program. If the p2 program requires I/O then again it switches to P3 and so on.

If there is no other program remaining after P3 then the CPU will pass its control back to the previous program.

Advantages

The advantages of multiprogramming operating system are as follows –

CPU utilization is high because the CPU is never goes to idle state.

Memory utilization is efficient.

CPU throughput is high and also supports multiple interactive user terminals.

Disadvantages

The disadvantages of multiprogramming operating system are as follows –

CPU scheduling is compulsory because lots of jobs are ready to run on CPU simultaneously.

User is not able to interact with jobs when it is executing.

Programmers also cannot modify a program that is being executed.

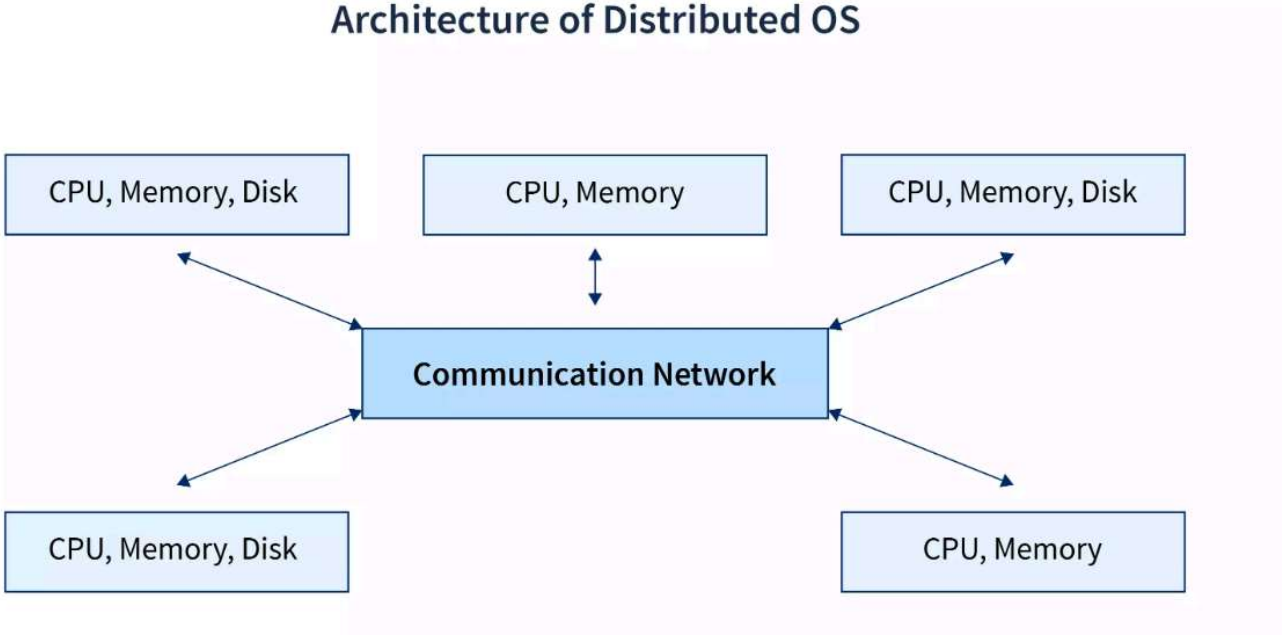
If several jobs are ready in main memory and if there is not enough space for all of them, then the system has to choose them by making a decision, this process is called job scheduling.

When the operating system selects a job from the group of jobs and loads that job into memory for execution, therefore it needs memory management, if several such jobs are ready then it needs CPU **scheduling**.

4.Distributed Operating System

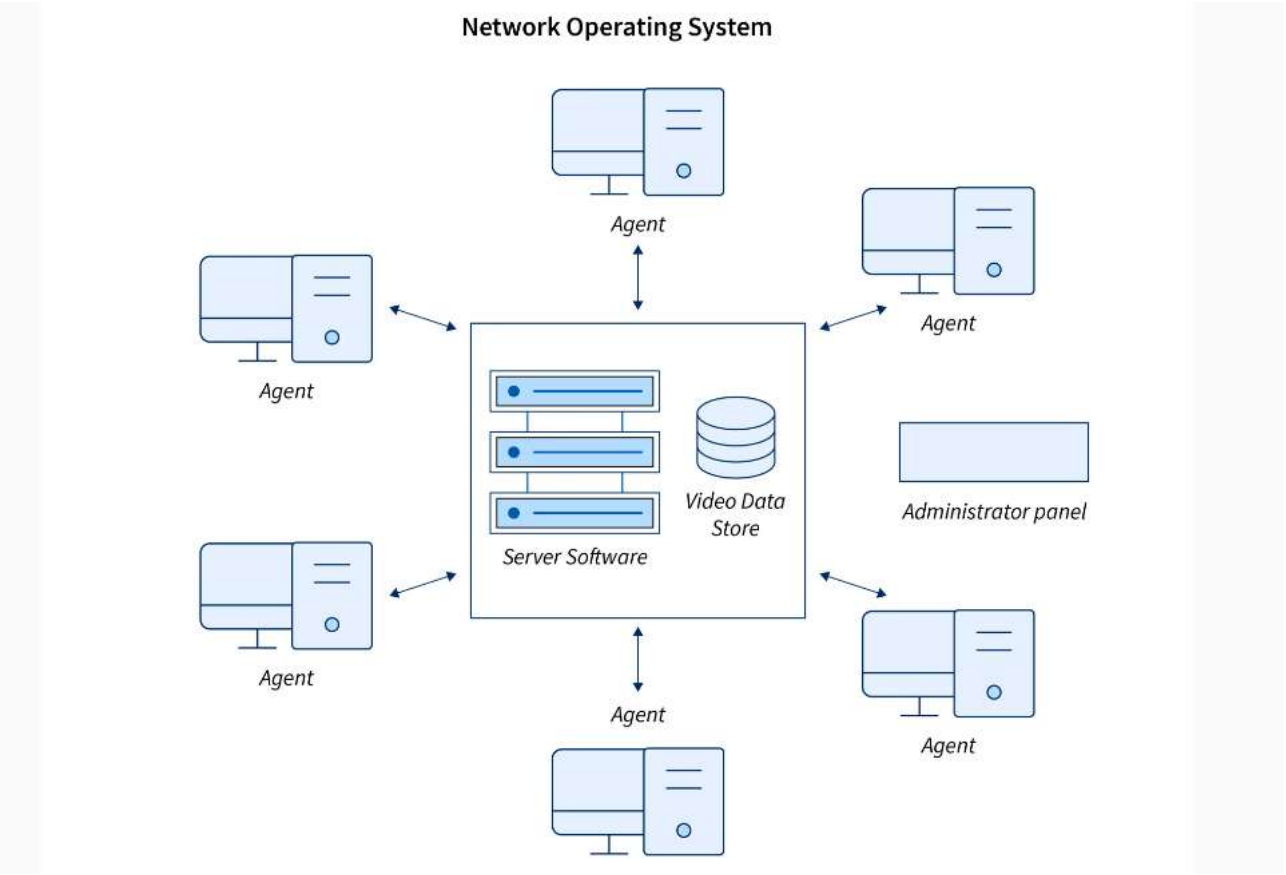
In Distributed Operating System, we have various systems and all these systems have their own CPU, main memory, secondary memory, and resources.

These systems are connected to each other using a shared communication network. Here, each system can perform its task individually. It is also known as loosely coupled system.



5.Network Operating System

This system run on a server and provide the capability to manage data, users, groups, security, applications and other networking functions. This type of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network. Network operating system is also known as tightly coupled system.



Unit 2

What are processess?

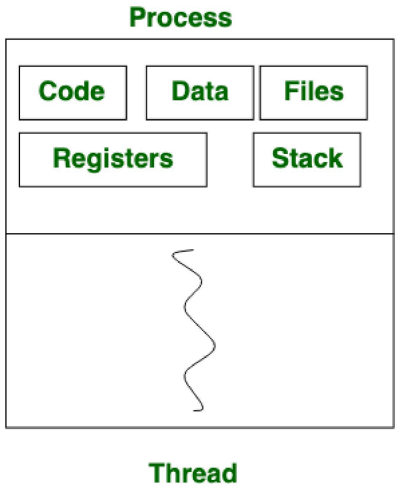
Process: Processes are basically the programs that are dispatched from the ready state and are scheduled in the CPU for execution

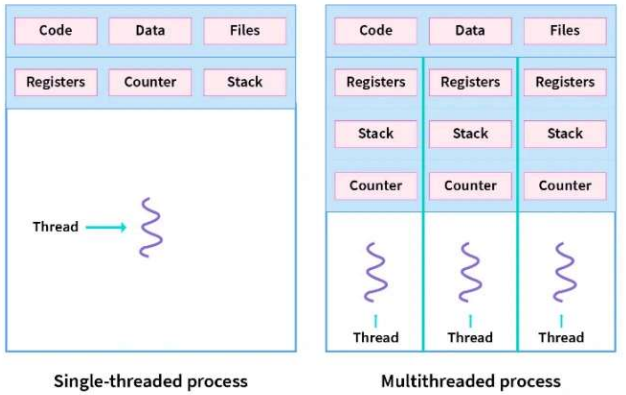
A process is mainly a program in execution where the execution of a process must progress in sequential order or based on some priority or algorithms. In other words, it is an entity that represents the fundamental working that has been assigned to a system.

When a program gets loaded into the memory, it is said to as a process. This processing can be categorized into four sections. These are:

- Heap
- Stack
- Data
- Text

Thread: Thread is the segment of a process which means a process can have multiple threads and these multiple threads are contained within a process. The thread takes less time to terminate as compared to the process but unlike the process, threads do not isolate.





Types of Thread

Threads are implemented in following two ways –

- **User Level Threads** – User managed threads.
- **Kernel Level Threads** – Operating System managed threads acting on kernel, an operating system core.

Difference between Process and Thread:

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means a segment of a process.
2.	The process takes more time to terminate.	The thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.

S.NO	Process	Thread
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
6.	Multiprogramming holds the concepts of multi-process.	We don't need multi programs in action for multiple threads because a single process consists of multiple threads.
7.	The process is isolated.	Threads share memory.
8.	The process is called the heavyweight process.	A Thread is lightweight as each thread in a process shares code, data, and resources.
9.	Process switching uses an interface in an operating system.	Thread switching does not require calling an operating system and causes an interrupt to the kernel.
10.	If one process is blocked then it will not affect the execution of other processes	If a user-level thread is blocked, then all other user-level threads are blocked.
11.	The process has its own Process Control Block, Stack, and Address Space.	Thread has Parents' PCB, its own Thread Control Block, and Stack and common Address space.
12.	Changes to the parent process do not affect child processes.	Since all threads of the same process share address space and other resources so any changes to the main thread may affect the behavior of the other threads of the process.
13.	A system call is involved in it.	No system call is involved, it is created using APIs.
14.	The process does not share data with each other.	Threads share data with each other.

Operations on process

The user can perform the following operations on a process in the operating system:

1. Process creation
2. Process scheduling or dispatching
3. Blocking
4. Preemption
5. Termination

- **Process creation**

Process creation is the initial step to process execution. It implies the creation of a new process for execution.

- **Process scheduling\dispatching**

Scheduling or dispatching refers to the event where the OS puts the process from ready to running state. It is done by the system when there are free resources or there is a process of higher priority than the ongoing process.

- **Blocking**

Block mode is a mode where the system waits for input-output. In process blocking operation, the system puts the process in the waiting state. When a task is blocked, it is unable to execute until the task prior to it has finished using the shared resource. Examples of shared resources are the CPU, network and network interfaces, memory, and disk.

- **Preemption**

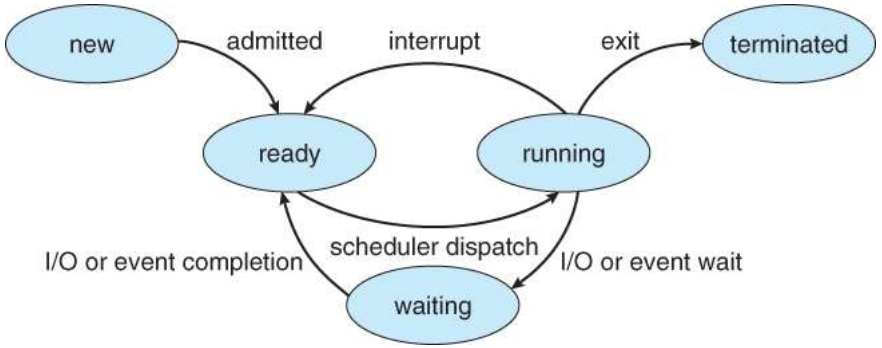
Preemption means the ability of the operating system to preempt a currently scheduled task in favour of a higher priority task. The resource being scheduled can be the processor or the I\O.

- **Termination**

Ending a process is known as process termination. There are many events that may lead to process termination, some of them are:

1. One process terminating the other process.
2. A problem in the hardware.
3. The process is fully executed, implying that the OS is finished.
4. An operating system might terminate itself due to service errors.

Process States



- **NEW**: A new process is being created.
- **READY**: A process is ready and waiting to be allocated to a processor.
- **RUNNING**: The program is being executed.
- **WAITING**: Waiting for some event to happen or occur.
- **TERMINATED**: Execution finished.

Process Attributes

A process has various attributes associated with it. Some of the Attributes are : **Process ID**, **Process State**, **CPU Scheduling Information**, **I/O Information**, **Priority**, **Process Counter**

Process ID: Every process will be given an ID called process IO to uniquely identify that process from the other processes.

Process State: Each and Every process has some states associated with it at a particular instant of time. This is denoted by process state. It can be new, ready, running, waiting and terminated.

Cpu Scheduling Information: Each Process is executed by using some process scheduling algorithms like FCFS, SJF, Round-Robin etc.

I/O Information: Each process needs some I/O devices for their execution. So, the information about device allocated and device need is crucial.

Priority: Every Process has its own priority. The process with the highest priority among the processes gets the cpu first.

Process Counter: Process (program) counter stores the address of the last instruction of the process on which the process was suspended.

Inter Process Communication

Inter process communication in OS is way by which multiple processes can communicate with each other. Shared memory in OS, message queues, FIFO etc are some of the ways to achieve ipc in os.

Processes executing concurrently in the operating system may be either independent processes or cooperating processes.

Independent processes: They cannot affect or be affected by the other processes executing in the system.

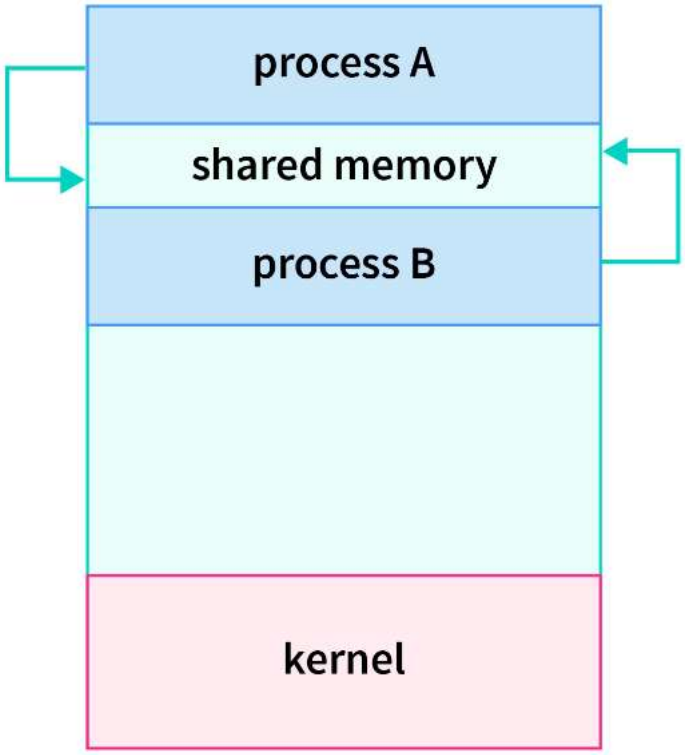
Cooperating Processes: They can affect or affected by the other processes executing in the system.

Any process that shares data with other processes is a cooperating process.There are several reasons for providing an environment that allows process cooperation:

- Information sharing
- Computation speed up
- Modularity
- Convenience

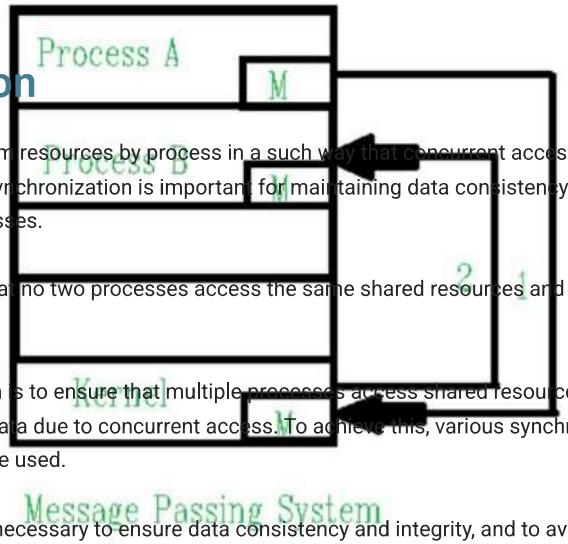
In terms of exchanging data and information, cooperating processes requires an interprocess communication (IPC) that will allow to share information among processes.
There are two fundamental models of interprocess communication:
1. Shared Memory
2. Message Passing

Shared Memory model: A region of memory that is shared by cooperating processes is established. Processes can exchange information by reading and writing data to the shared region.



Message passing: In the Message passing model, communication takes place by means of messages exchanged between the cooperating processes.

Process Synchronization



Process synchronization means sharing system resources by process in a such way that concurrent access to shared data is handled thereby minimizing the chance of inconsistent data. Synchronization is important for maintaining data consistency demands mechanisms to ensure synchronized execution of co-operating processes.

Coordinating the execution of processes so that no two processes access the same shared resources and data is known as process synchronization.

The main objective of process synchronization is to ensure that multiple processes access shared resources without interfering with each other and to prevent the possibility of inconsistent data due to concurrent access. To achieve this, various synchronization techniques such as semaphores, monitors, and critical sections are used.

In a multi-process system, synchronization is necessary to ensure data consistency and integrity, and to avoid the risk of deadlocks and other synchronization problems. Process synchronization is an important aspect of modern operating systems, and it plays a crucial role in ensuring the correct and efficient functioning of multi-process systems.

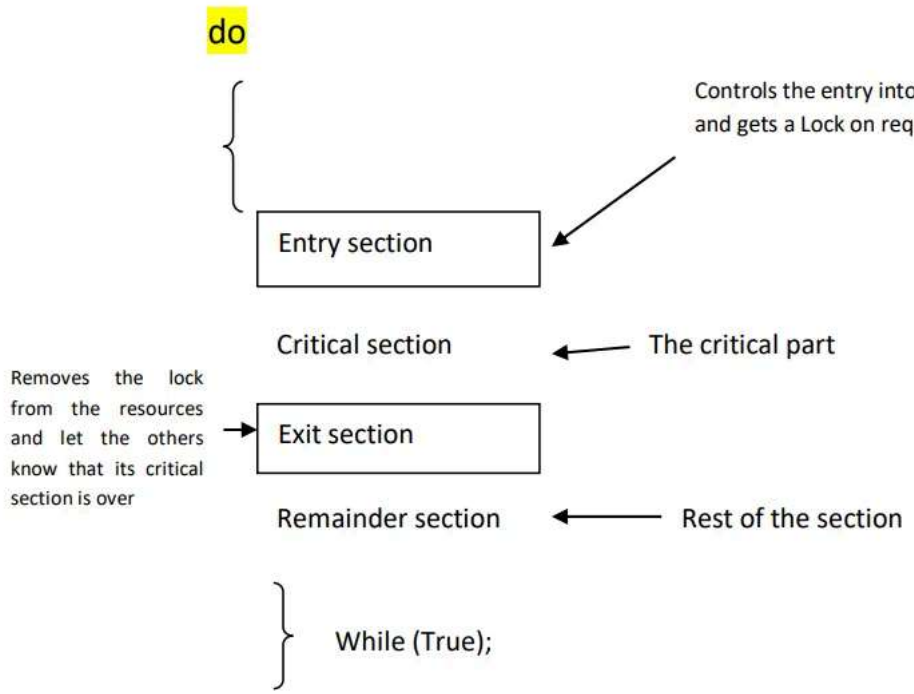
Resources can be variable, memory, code, resources like cpu, printer, scanner and other various hardware.

Critical section Problem

The critical section is a code segment where the shared variables can be accessed. An atomic action is required in a critical section i.e. only one process can execute in its critical section at a time. All the other processes have to wait to execute in their critical sections.

A critical section is a code segment that accesses shared variables and has to be executed as an atomic action. It means that in a group of co-operating processes, at a given point of time, only one process must be executing its critical section. If other processes also want to execute its critical section, it must wait until the first one finishes.

Critical section is the part of the program where shared resources are accessed by various processes. It is the place where shred variable, resources are placed.



In the above diagram, the entry section handles the entry into the critical section. It acquires the resources needed for execution by the process. The exit section handles the exit from the critical section. It releases the resources and also informs the other processes that the critical section is free.

Solution to Critical section Problem:

A solution to the critical section problems must satisfy the following conditions:

1. Mutual exclusion
2. Progress
3. Bounded waiting
4. No assumption related to H/W speed

1. Mutual exclusion: Out of a group of co-operating processes, only one process can be in its critical section at a given point of time.

2. Progress: If no process is in its critical section and if one or more process wants to execute in critical section than one of these process must be allowed to get into its critical section.

3. Bounded waiting: After a process makes a request for getting into its critical section, there is a limit for how many other processes can get into their critical section, before this process's request is granted. So after the limit time is reached, system must grant the process permission

to get into its critical section.

Peterson Solution

4. No assumption related to H/W speed: Synchronization H/W: Many systems provide H/W support for critical section code. The critical section

```
//Structure of process P[i] in peterson's solution
do{
   flag[i]=true;
   turn j;
   while(flag[j]&&turn==[j];
   Critical section
   flag[i]=false;
   remainder section
  } while(TRUE);
```

```
//Structure of process P[j] in peterson's solution
do{
   flag[j]=true;
   turn i;
   while(flag[i]&&turn==[i];
   Critical section
   flag[j]=false;
   remainder section
  } while(TRUE);
```

Semaphore

Semaphore is a hardware-based solution to the critical section problem. Dijkstra proposed the concept of semaphore in 1965. Semaphore provides general purpose solution to impose mutual exclusion among concurrently executing processes, where many processes want to execute in their critical section but only one at a time is allowed and rest all other are excluded.
A semaphore basically consists of an integer variable S, shared by processes.
S is a protected variable that can only be accessed and manipulate by two operation:- wait() and signal().
Wait() and signal() are semaphore primitives
The wait is sometimes called down(), P(), or Sleep() and signal is called up(), v() or Wake up().
The wait and signal primitives ensures that one process at a time enters in its critical section and rest all other processes wanting to in their critical sections are kept waiting.
Semaphore is an integer variable that are used to accessed the shared resources through two standard atomatic operations: wait() and signal().
Here, wait means to test and signal means to increment.

The classical definition of wait() is :

```
wait()
{
while (S<=0);
//busy wait
S=S-1;
}
```

The classical definition of signal() is:

```
signal()
{
S=S+1;
}
```

Properties of semaphore:

1. It is simple and always have a non-negative integer value
2. Works with many processes

3. Can have many different critical sections with different semaphores
4. Each critical section has unique access semaphores.
5. Can permit multiple processes into critical section at once, if desirable.
6. Solution to critical section
7. Act as resource management
8. It also decide the order of execution among the process(n-process)

Semaphores are of two types:

1.Binary Semaphore –
Binary semaphore is also known as a mutex lock. It can have only two values – 0 and 1. Its value is initialized to 1. It is used to implement the solution of critical section problems with multiple processes. Implementation of Binary semaphore:

```
Down ( semaphore S)
{
if(S.value==1)
{
S.value=0;
}
else
{
//perform block operation and move the process to thesemaphore queue* or put process(PCB) in suspended list
Sleep() or block ();
}
}

Up(semaphore S)
{
if (semaphore queue is empty)
{
S.value=1;
}
else
{
/* semaphore queue is not empty, perform wakeup( ) and move the firstprocess from semaphores queue to the ready queue or remove a process (P) from suspended list*/
wakeup(P);
}
}
```

2.Counting Semaphore –
The value of counting semaphore can range over an unrestricted domain(-∞ to ∞).
Counting semaphores can be used to control access to a given resource consisting of finite number of instances.

The semaphore is initialized to the number of resources available.
Each processes that wishes to use a resource performs a wait() operation on the semaphore (thereby decrementing the count) When a process releases a resource, it performs a signal () operation (incrementing the count).
When a count for the semaphore goes to all '0',all resources are being used.
After that, processes that wish to use a resource will block until the count becomes greater than ' 0'.

Monitors

A monitor is a programming language construct that controls access to shared data.Synchronizes execution within procedures that manipulate encapsulated data shared among procedures.

- Only one thread can execute within a monitor at a time.
- Relies upon high-level language support

Scheduling algorithms

1. First come first served Scheduling algorithm
2. Round robin Scheduling Algorithm
3. Shortest job first Scheduling Algorithm
4. Shortest Remaining time first (srtf) scheduling algorithm

Fcfs (first come first served): Fcfs is the simplest CPU scheduling algorithm in which the process that requests the cpu first. The implementation of FCFS algorithm is managed with FIFO (First in first out) queue. FCFS scheduling is non-preemptive. Nonpreemptive means, once the CPU has been allocated to a process, that process keeps the CPU until it executes a work or job or task and releases the CPU, either by requesting I/O.

Buying a movie ticket from the ticket counter is a perfect real-life example of a first come first serve (FCFS) algorithm. The person who comes first and stands in the queue gets to buy the ticket first. Similarly in the FCFS scheduling algorithm, the process that arrives first gets executed first.

In CPU-scheduling problems some terms are used while solving the problems:-

Arrival time (AT) – Arrival time is the time at which the process arrives in ready queue.

Burst time (BT) or CPU time of the process – Burst time is the unit of time in which a particular process completes its execution.

Completion time (CT) – Completion time is the time at which the process has been terminated.

Turn-around time (TAT) – The total time from arrival time to completion time is known as turn-around time. To fined TAT, Turn-around time (TAT) = Completion time (CT) – Arrival time (AT) or, TAT = Burst time (BT) + Waiting time (WT)

Waiting time (WT) – Waiting time is the time at which the process waits for its allocation while the previous process is in the CPU for execution. Waiting time (WT) = Turn-around time (TAT) – Burst time (BT)

Response time (RT) – Response time is the time at which CPU has been allocated to a particular process first time. In case of non-preemptive scheduling, generally Waiting time and Response time is same.

Gantt chart – Gantt chart is a visualization which helps to scheduling and managing particular tasks in a project. It is used while solving scheduling problems, for a concept of how the processes are being allocated in different algorithms.

C Program to implement FCFS scheduling algorithm

```
#include <stdio.h>
// Function to find the waiting time for all processes
int waitingtime(int proc[], int n,
int burst_time[], int wait_time[]) {
// waiting time for first process is 0
wait_time[0] = 0;
// calculating waiting time
for (int i = 1; i < n ; i++ )
wait_time[i] = burst_time[i-1] + wait_time[i-1] ;
return 0;
}
// Function to calculate turn around time
int turnaroundtime( int proc[], int n,
int burst_time[], int wait_time[], int tat[]) {
// calculating turnaround time by adding
// burst_time[i] + wait_time[i]
int i;
for ( i = 0; i < n ; i++)
tat[i] = burst_time[i] + wait_time[i];
return 0;
}
//Function to calculate average time
int avgtime( int proc[], int n, int burst_time[]) {
int wait_time[n], tat[n], total_wt = 0, total_tat = 0;
int i;
//Function to find waiting time of all processes
waitingtime(proc, n, burst_time, wait_time);
```

```
//Function to find turn around time for all processes
turnaroundtime(proc, n, burst_time, wait_time, tat);
//Display processes along with all details
printf("Processes Burst Waiting Turn around
");
// Calculate total waiting time and total turn
// around time
for ( i=0; i<n; i++) {
total_wt = total_wt + wait_time[i];
total_tat = total_tat + tat[i];
printf(" %d\t %d\t\t %d \t%d
", i+1, burst_time[i], wait_time[i], tat[i]);
}
printf("Average waiting time = %f
", (float)total_wt / (float)n);
printf("Average turn around time = %f
", (float)total_tat / (float)n);
return 0;
}
// main function
int main() {
//process id's
int proc[] = { 1, 2, 3};
int n = sizeof proc / sizeof proc[0];
//Burst time of all processes
int burst_time[] = {5, 8, 12};
avgtime(proc, n, burst_time);
return 0;
}
```

Round Robin Scheduling Algortihm

In Round Robin Scheduling,

CPU is assigned to the process on the basis of FCFS for a fixed amount of time.
This fixed amount of time is called as time quantum or time slice.
After the time quantum expires, the running process is preempted and sent to the ready queue.
Then, the processor is assigned to the next arrived process.
It is always preemptive in nature.

Advantages-
It gives the best performance in terms of average response time.

It is best suited for time sharing system, client server architecture and interactive system.

Disadvantages-
It leads to starvation for processes with larger burst time as they have to repeat the cycle many times.
Its performance heavily depends on time quantum.
Priorities can not be set for the processes.

Question: Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	Arrival time	Burst time
P1	0	5
P2	1	3
P3	2	1
P4	3	2
P5	4	3

If the CPU scheduling policy is Round Robin with time quantum = 2 unit, calculate the average waiting time and average turn around time.

```
#include<stdio.h>

int main()
{

int cnt,j,n,t,remain,flag=0,tq;
int wt=0,tat=0,at[10],bt[10],rt[10];
printf("Enter Total Process:\t ");
scanf("%d",&n);
remain=n;
for(cnt=0;cnt<n;cnt++)
{
printf("Enter Arrival Time and Burst Time for Process Process Number %d :",cnt+1);
scanf("%d",&at[cnt]);
scanf("%d",&bt[cnt]);
rt[cnt]=bt[cnt];
}
printf("Enter Time Quantum:\t");
```

```
scanf("%d",&tq);
printf("\n\nProcess\t|Turnaround Time|Waiting Time\n\n");
for(t=0,cnt=0;remain!=0;)
{
if(rt[cnt]<=tq && rt[cnt]>0)
{
t+=rt[cnt];
rt[cnt]=0;
flag=1;
}
else if(rt[cnt]>0)
{
rt[cnt]-=tq;
t+=tq;
}
if(rt[cnt]==0 && flag==1)
{
remain–;
printf("P[%d]\t|\t%d\t|\t%d\n",cnt+1,t-at[cnt],t-at[cnt]-bt[cnt]);
wt+=t-at[cnt]-bt[cnt];
tat+=t-at[cnt];
flag=0;
}
if(cnt==n-1)
cnt=0;
else if(at[cnt+1]<=t)
cnt++;
else
cnt=0;
}
printf("\nAverage Waiting Time= %f\n",wt*1.0/n);
printf("Avg Turnaround Time = %f",tat*1.0/n);

return 0;
}
```

Shortest job Next or shortest job first Scheduling Algortihm

This is also known as shortest job first, or SJF
This is a nonpreemptive scheduling algorithm.
Best approach to minimize waiting time.
Easy to implement in Batch systems where required CPU time is known in advance.
Impossible to implement in interactive systems where required CPU time is not known.
Processer should know in advance how much time process will take.

In SJF Scheduling,

Out of all the available processes, CPU is assigned to the process having smallest burst time.
In case of a tie, it is broken by FCFS Scheduling.
SJF Scheduling can be used in both preemptive and non-preemptive mode.
Preemptive mode of Shortest Job First is called as Shortest Remaining Time First (SRTF).

Advantages-
SRTF is optimal and guarantees the minimum average waiting time.
It provides a standard for other algorithms since no other algorithm performs better than it.

Disadvantages-
It can not be implemented practically since burst time of the processes can not be known in advance.
It leads to starvation for processes with larger burst time.
Priorities can not be set for the processes.
Processes with larger burst time have poor response time.

Question:-Consider the set of 5 processes whose arrival time and burst time are given below-

Process Id	**Arrival time**	**Burst time**
P1	3	1
P2	1	4
P3	4	2
P4	0	6
P5	2	3

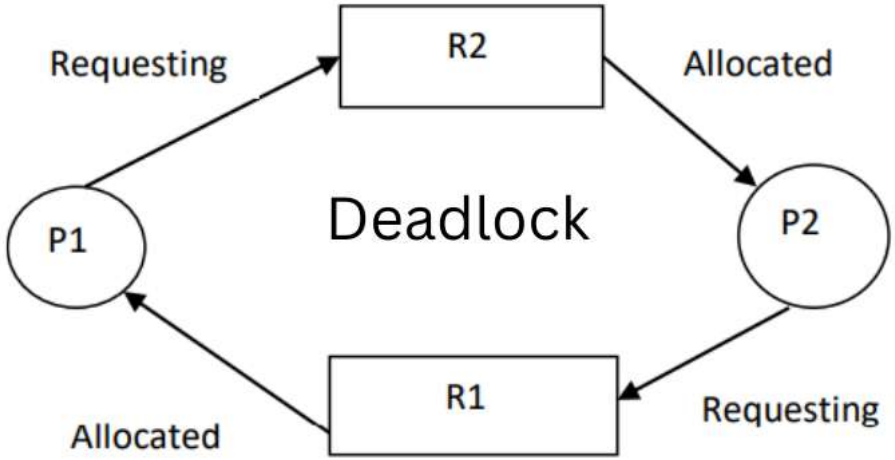
If the CPU scheduling policy is SJF non-preemptive, calculate the average waiting time and average turn around time.

Shortest Remaining Time First

SRTF is also known as SJF mode with Preemption in which jobs are scheduled according to shortest remaining time.

Deadlock

A set of process is in a deadlocked state when every process in the set is waiting for an event can be caused only by another process in the set. A system is said to be in deadlock when if a process unable to change its waiting state indefinitely because the resources requested by it are hold by another waiting process.



Necessary Conditions for Deadlock:

A deadlock situation can arise if the following four conditions hold simultaneously in the system:

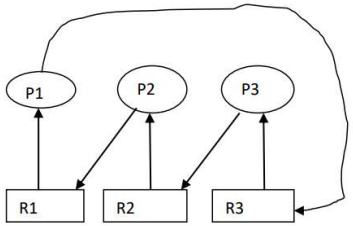
1. Mutual Exclusion
2. Hold and Wait
3. No Preemption
4. Circular Wait

Mutual Exclusion: Only one process can use the resource at a time.

Hold and Wait: A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by the other process.

No Preemption: Resources cannot be preempted i.e a resource can be released only voluntarily by process holding it after that process has completed its task.

Circular Wait: A set of process {p0…….Pn} of waiting processes must exist such that P0 is waiting for a resources held by P1, P1 is waiting for a resources held by P2……,Pn-1 is waiting for a resource held by Pn and Pn is waiting for a resource held by P0.



Deadlock Handling Methods:

1. Deadlock Ignorance (ostrich Method)
2. Deadlock Prevention
3. Deadlock avoidance (Bankers algorithm)
4. Deadlock detection and Recovery.

Bankers Algorithm for Deadlock Avoidance

Let us consider the following snapshot for understanding the banker's algorithm:

Processes	Allocation A B C	Max A B C	Available A B C
P0	1 1 2	4 3 3	2 1 0
P1	2 1 2	3 2 2	
P2	4 0 1	9 0 2	
P3	0 2 0	7 5 3	
P4	1 1 2	1 1 2	

1. calculate the content of the need matrix?
2. Check if the system is in a safe state?
3. Determine the total sum of each type of resource?

Unit 3 Memory Management

Swapping

Swapping is a memory management scheme in which any process or programs can be temporarily swapped from main memory to secondary memory so that the main memory can be made available for other processes.

- Swap-out is a way of removing a process from RAM and adding it to the hard disk /secondary storage device.
- Swap-in is a way of removing a program from a hard disk and putting it back into the main memory or RAM.

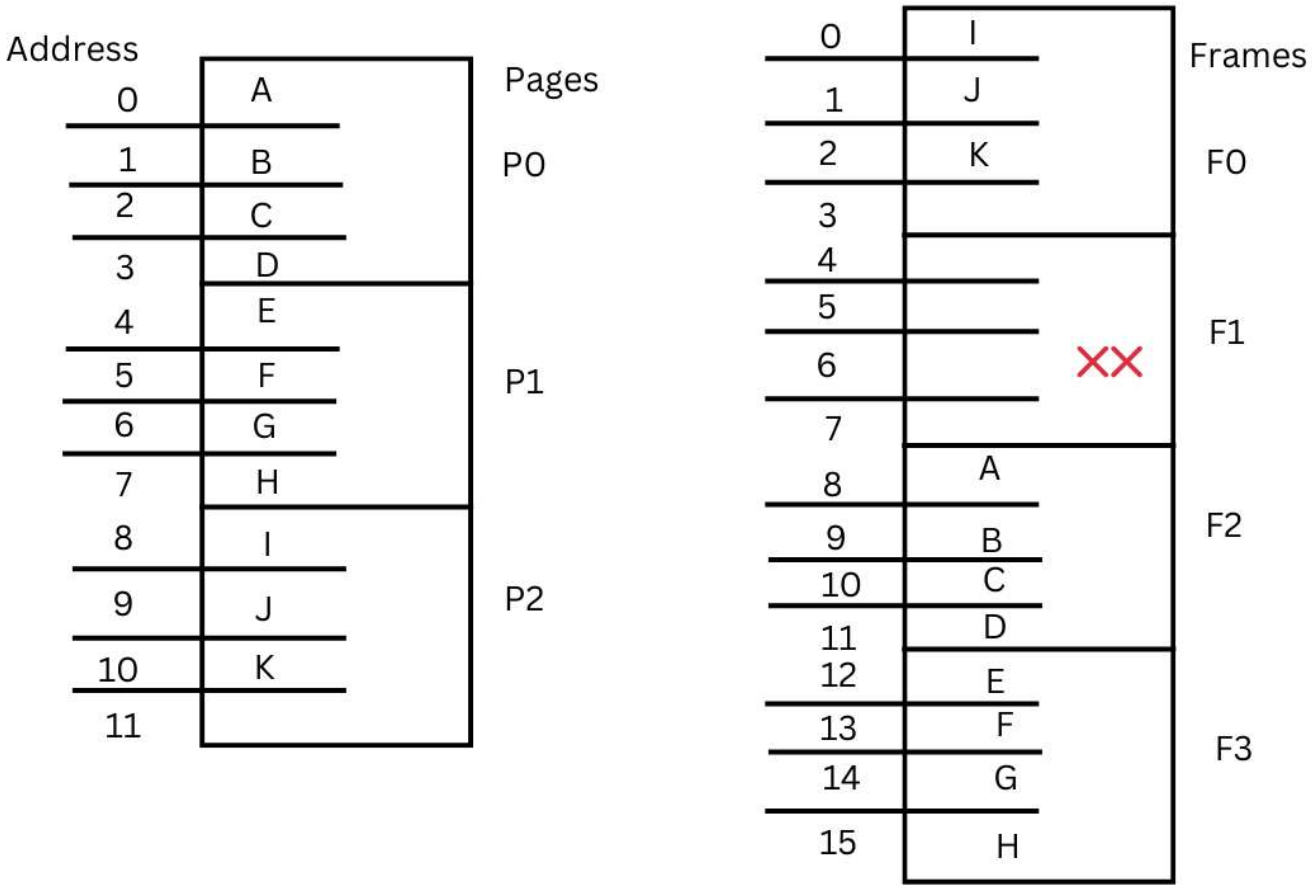
Paging

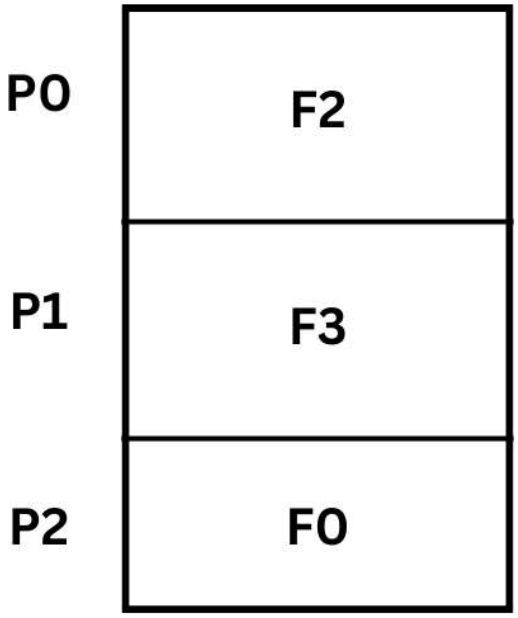
Paging is a non contigious memory allocation scheme used to retrieve processes from the secondary storage into the main memory in the form of pages. The basic purpose of paging is to separate each processes into pages. Additionally, frames will be used to separate the main memory.

In paging, the physical memory is divided into fixed-size blocks called page frames, which are the same size as the pages used by the process. The process's logical address space is also divided into fixed-size blocks called pages, which are the same size as the page frames. When a process requests memory, the operating system allocates one or more page frames to the process and maps the process's logical pages to the physical page frames. The sizes of each frame must be equal.

The mapping between logical pages and physical page frames is maintained by the page table, which is used by the memory management unit to translate logical addresses into physical addresses. The page table maps each logical page number to a physical page frame number.

A data structure called page map table is used to keep track of the relation between a page of a process to a frame in physical memory.


Address
0
1
2
3
4
5
6
7
8
9
10
11
A
B
C
D
E
F
G
H
I
J
K
Pages
P0
P1
P2
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
I
J
K
A
B
C
D
E
F
G
H
XX
Frames
F0
F1
F2
F3



Page table is used for memory mapping between logical address and physical address.

Paging:
Paging is a memory management scheme by which a computer stores and retrieves data from secondary storage for use in main memory.
Paging is a process of reading data from, and writing data to the secondary storage.
Partition memory into small equal fixed-size chunks and divide each process into the same size chunks.
The chunks of a process are called pages.
The chunks of memory are called frames.
The basic idea behind paging is to only keep those part of the processes in memory that are actually being used
Paging is a non-contiguous memory allocation technique.
The basic idea behind paging is to only keep those part of the processes in memory that are actually being used
Paging:
– divides processes into a fixed-sized pages
-Then selectively allocates pages to frames in memory, and
-manages (moves, removes, reallocates) pages in memory

The logical memory of the process is still contiguous but the pages need not be allocated contiguously in memory
By dividing the memory into fixed-sized pages, we can eliminate external fragmentation
However, paging does not eliminate internal fragmentation (e.g4KB frame size and 15 KB of process)

Summary: breaking process memory into fixed-sized pages and map them to a physical frame of memory
OS needs to do two things
1. Mapping the pages to frame, maintain a page table
2. Translate the virtual address (contiguous) into a physical address (not so easy as the physical address for a process is scattered all over the memory)

The address generated by CPU (logical address) is divided into:

Page number p is an index into a page table that contains the base address of each page in physical memory.
Page offset d is a displacement, combined with the base address to define the physical memory address that is sent to the memory unit.
Page Fault

When the page (data) requested by a program is not available in the memory, it is called a page fault. This usually results in the application being shut down.
A page is a fixed-length memory block used as a transferring unit between physical memory and external storage. A page fault occurs when a program accesses a page that has been mapped in address space but has not been loaded in the physical memory.
Page hit
Whenever a process refers to a page that is present in memory, such a condition is known as page hit.

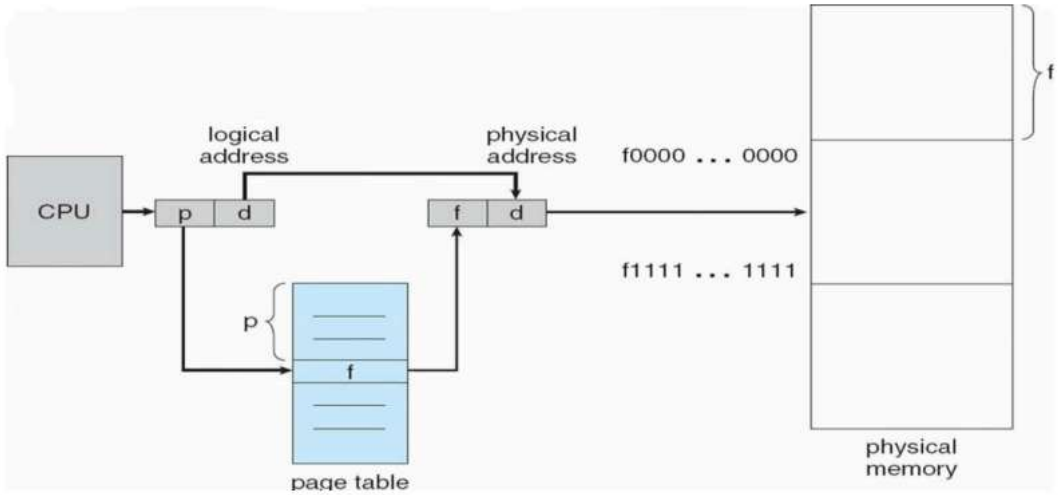
Page table:
A page table is the data structure used by a virtual memory system in a computer operating system to store the mapping between virtual addresses and physical addresses.

Frame:
Frame is a fixed sized block in physical memory space.

Advantages and Disadvantages of paging:

• Paging reduces external fragmentation, but still suffers from internal fragmentation.
• Paging is simple to implement and assumed as an efficient memory management technique.
• Due to the equal size of the pages and frames, swapping becomes very easy.
• Page table requires extra memory space, so may not be good for a system having small RAM.

Paging Hardware



Types of Paging in OS

In operating systems, paging is a memory management technique used to organize and manage memory in a computer system. There are primarily two types of Paging in OS commonly used in operating systems:

1. Fixed-size paging
2. Variable-size paging

Both fixed-size paging and variable-size paging have their advantages and trade-offs. Fixed-size paging provides simplicity and ease of management but can lead to internal fragmentation. Variable-size paging reduces fragmentation but requires more complex data structures and management algorithms.

Fixed-size Paging in Operating System

In fixed-size paging, the memory is divided into fixed-sized blocks or pages. Each page has a fixed size, typically a power of 2 (e.g., 4KB or 8KB). The entire logical address space of a process is divided into pages of the same size. This approach simplifies memory management, as each page can be easily allocated or deallocated. However, it can lead to internal fragmentation if the memory allocated to a process is not fully utilized.

Under fixed-size Paging in OS, the logical address space is divided into pages, and the physical memory is divided into frames of the same size. The page table maps logical pages to physical frames, allowing for efficient memory access and management.

Variable-size Paging in Operating System

In variable-size paging, the memory is divided into variable-sized blocks or pages. The size of each page is not fixed and can vary depending on

Segmentation

Segmentation is a non contiguous memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.

Difference Between Paging and Segmentation?

S.NO	Paging	Segmentation
1.	Paging divides programs or processes into Fixed size pages.	Segmentation divides programs into variable sized segments.

S.NO	Paging	Segmentation
2.	Paging is closer to Operating System.	Segmentation is closer to User.
3.	It suffers from internal fragmentation.	It suffers from external fragmentation.
4.	Logical address is divided into page number and page offset.	Logical address is divided into segment number and segment offset.
5.	Page table is used to maintain the page information.	Segment table is used to maintain segment information.
6.	Paging is invisible to user.	Segmentation is visible to user.
7.	Paging results in a less efficient system.	Segmentation results in a more efficient system.

the memory requirements of the process. Variable-size Paging in OS aims to reduce internal fragmentation by allocating memory in smaller chunks based on the actual memory needs of a process. This approach helps in more efficient memory utilization.

Variable-size paging requires maintaining additional data structures, such as segment tables or page tables with variable-size entries, to map logical addresses to physical frames. These data structures store information about the size and location of each page, allowing for flexible memory allocation.

Demand Paging in OS

Demand paging is a memory management scheme where pages are brought into memory only when they are demanded by the running process, reducing memory wastage and improving overall system performance.

When a page that is not present in memory is accessed, a page fault occurs, prompting the operating system to fetch the required page from secondary storage into memory. This approach allows for more efficient memory utilization as only the necessary pages are loaded, reducing initial memory overhead and allowing for larger program sizes to be accommodated.

Virtual Memory

Virtual Memory is a storage scheme that provides user an illusion of having a big main memory. This is done by treating a part of secondary memory as the main memory.

Benefits of Paging

Increased Memory Utilization: Paging in OS enables efficient utilization of physical memory by allocating memory in smaller fixed-size chunks. This allows the operating system to allocate memory to processes on-demand, rather than reserving a continuous block of memory for each process. As a result, multiple processes can share the available physical memory effectively.

Simplified Memory Management: Paging in OS simplifies memory management by removing the requirement for contiguous memory allocation. Each process is allocated a sequence of non-contiguous pages, which can be scattered throughout the physical memory. This flexibility enables efficient memory allocation and deallocation, making it easier for the operating system to manage available memory resources.

Memory Protection: Paging in OS provides an additional layer of memory protection. Each page can be marked as read-only, read-write, or not accessible, preventing unauthorized access to memory regions. This helps in isolating processes from each other, enhancing system security and stability.

Virtual Memory: Paging in OS forms the foundation for virtual memory systems. By using paging, the operating system can provide each process with a virtual address space that is larger than the available physical memory. Virtual memory allows the operating system to transparently swap pages in and out of physical memory to disk, enabling efficient memory usage even when the physical memory is limited.

Advantages of Virtual Memory:

1. The degree of multiprogramming will be increased.
2. Users can run the large applications with less real RAM size.
3. There is no need to buy more memory RAMS.

Disadvantages of Virtual Memory:

1. The system becomes slower since swapping takes time.
2. It takes more time in switching between applications.
3. The user will have lesser hard disk space for its use.

Page Replacement Algorithms

1. First in first out (FIFO)
2. Optimal Page Replacement
3. Least recently used

1. FIRST IN FIRST OUT:

Example2: Consider the page reference string 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frames. Find number of page faults.

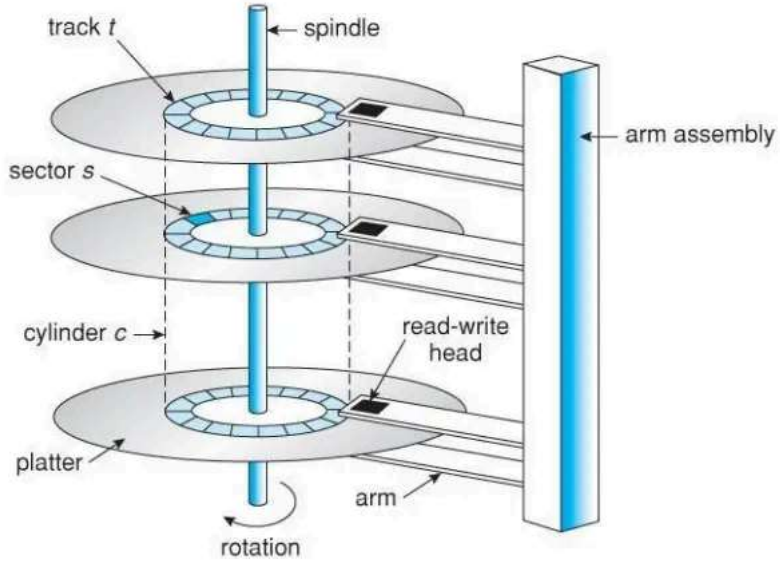
2. OPTIMAL PAGE REPLACEMENT

Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.

3. LEAST RECENTLY USED:

Example-2: Consider the page references 7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 3 with 4 page frame. Find number of page fault.

Unit-4 Disk Structure



Rajesh Parajuli

+977-9847546279
parajulirajesh2072@gmail.com

Useful Links

- Home
- Services
- Contact Me
- Terms & Conditions

Worked with

- BidhyaTech
- JK Arts
- MastaSoftsolution
- Ghodaghodi Multiple Campus

Location

Ghodaghodi Municipality-1, Kailali Nepal