

A powerful scanner is in your pocket.

You no longer need to keep piles of paper or stuff your wallet with receipts. The free OKENScanner mobile app works on your mobile device, scans documents into PDFs, and automatically recognizes text.

1.Turn Paper and Whiteboard to PDF

Scan anything in a snap.

With OKENScanner, easily capture and convert documents, forms, business cards, and whiteboards into high-quality PDFs. And with different capture modes along with automatic boundary detection, you can get the best scans every time.

2.AI-Enhanced Scans

Intelligently cleaner and clearer scans.

With the powerful AI behind many of our intelligent features, OKENScanner corrects image perspective and sharpens handwritten or printed text, while removing elements you don't want, like glare and shadow.

3.Edit Freely

Fill & Sign PDF on the go.

Eliminate the hassle of finding a printer, filling a form by hand, and scanning it again. Use OKENScanner to turn paper forms into interactive digital documents you can easily fill out, sign and share.

4.Save pages as JPEG

Do more with your scans.

Flexibility to save as a JPEG to do more with your scans than work on PDFs. Upload to your favorite apps or incorporate your scans into documents.



Chapter Outline

1.0 INTRODUCTION TO OPERATING SYSTEM

1.1 HISTORY OF OPERATING SYSTEM

1.2 TYPES OF OPERATING SYSTEM

1.3 DISTINGUISH SPOOLING AND BUFFERING

1.4 GOALS OF OPERATING SYSTEM

1.5 COMPONENTS OF OPERATING SYSTEM

1.6 MULTIPROGRAMMING & TIME SHARING

1.7 DIFFERENTIATE BETWEEN DISTRIBUTED AND REALTIME SYSTEMS

1.8 MULTIPROCESSOR SYSTEM

1.9 OPERATING SYSTEM SERVICES:

1.10 SYSTEM CALL WITH AN EXAMPLE

1.11 DIFFERENT TYPES OF SYSTEM CALLS

1.12 SINGLE USER, MULTI USER OPERATING SYSTEM STRUCTURES

1.0 INTRODUCTION TO OPERATING SYSTEM

Definition : An operating system is a program that acts as an interface between the user and the computer hardware and controls the execution of all kinds of programs.

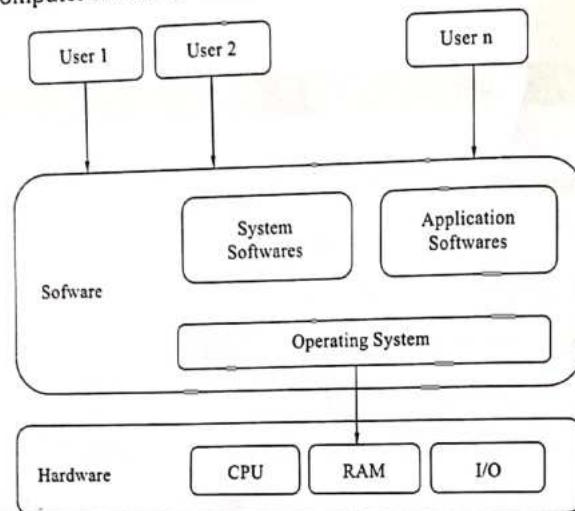


FIG 1.1 :

An operating system is software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers.

1.1 HISTORY OF OPERATING SYSTEM

A brief history of operating systems in tabular form is given as follows :

Generation	Year	Electronic Device Used	Operating System Device used
First Generation	1945 - 1955	Vacuum Tubes	Plug Boards
Second Generation	1955 - 1965	Transistors	Batch Systems
Third Generation	1965-1980	Integrated Circuits	Multiprogramming
Fourth Generation	1980 - Present	Large scale Integration Devices	Personal Computers

The First Generation (1945 - 1955) : Vacuum Tubes And Plugboards

- Digital computers were not constructed until the Second World War.
- Calculating engines with mechanical relays were built at that time.

- However, the mechanical relays were very slow and were later replaced with vacuum tubes.

- These machines were enormous but were still very slow.

- By the 1950's punch cards were introduced and this improved the computer system. Instead of using plug boards, programs were written on cards and read into the system.

The Second Generation (1955 - 1965) : Transistors And Batch Systems

- Transistors led to the development of the computer systems that could be manufactured and sold to paying customers.
- These machines were known as mainframes and were locked in air-conditioned computer rooms with staff to operate them.
- The Batch System was introduced to reduce the wasted time in the computer.
- A tray full of jobs was collected in the input room and read into the magnetic tape. After that, the tape was rewound and mounted on a tape drive.
- Then the batch operating system was loaded in which read the first job from the tape and ran it. The output was written on the second tape.
- After the whole batch was done, the input and output tapes were removed and the output tape was printed.

The Third Generation (1965 - 1980) : Integrated Circuits and Multiprogramming

- Until the 1960's, there were two types of computer systems i.e the scientific and the commercial computers. These were combined by IBM in the System/360.
- This used integrated circuits and provided a major price and performance advantage over the second generation systems.
- The third generation operating systems also introduced multiprogramming. This meant that the processor was not idle while a job was completing its I/O operation.
- Another job was scheduled on the processor so that its time would not be wasted.

The Fourth Generation (1980 - Present): Personal Computers

- Personal Computers were easy to create with the development of large-scale integrated circuits.
- These were chips containing thousands of transistors on a square centimeter of silicon. Because of these, microcomputers were much cheaper than minicomputers and that made it possible for a single individual to own one of them.

- The advent of personal computers also led to the growth of networks. This created network operating systems and distributed operating systems.
- The users were aware of a network while using a network operating system and could log in to remote machines and copy files from one machine to another.

1.2 TYPES OF OPERATING SYSTEM

- Batch Operating System** : This type of operating system does not interact with the computer directly.

There is an operator which takes similar jobs having the same requirement and group them into batches.

It is the responsibility of the operator to sort jobs with similar needs.

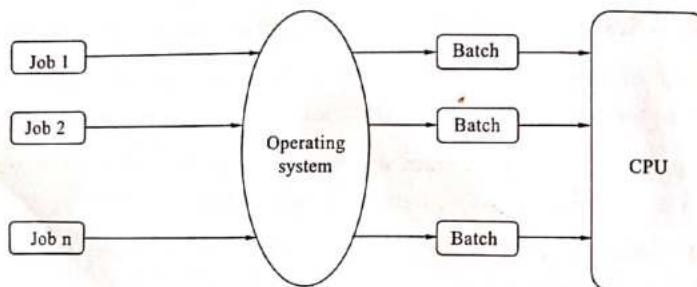


FIG 1.2 :

Advantages of Batch Operating System :

- It is very difficult to guess or know the time required for any job to complete. Processors of the batch systems know how long the job would be when it is in queue
- Multiple users can share the batch systems
- The idle time for the batch system is very less
- It is easy to manage large work repeatedly in batch systems

Disadvantages of Batch Operating System :

- The computer operators should be well known with batch systems
- Batch systems are hard to debug
- It is sometimes costly
- The other jobs will have to wait for an unknown time if any job fails

Examples of Batch based Operating System: Payroll System, Bank Statements, etc

- Time-Sharing Operating Systems** : Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of CPU as they use a single system. These systems are also known as Multitasking Systems. The task can be from a single user or different users also.

The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.

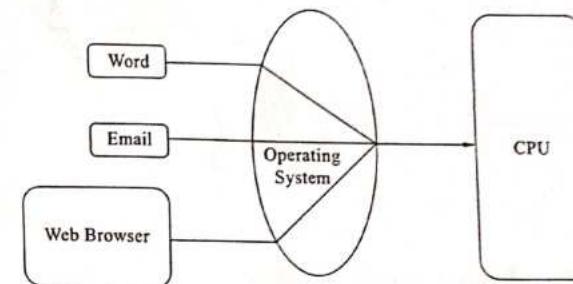


FIG 1.3 :

Advantages of Time-Sharing OS :

- Each task gets an equal opportunity
- Fewer chances of duplication of software
- CPU idle time can be reduced

Disadvantages of Time-Sharing OS :

- Reliability problem
- One must have to take care of the security and integrity of user programs and data
- Data communication problem

Examples of Time-Sharing OSs are: Multics, Unix, etc.

- Distributed Operating System** : These types of the operating system is a recent advancement in the world of computer technology and are being widely accepted all over the world and, that too, with a great pace. Various autonomous interconnected computers communicate with each other using a shared communication network. Independent systems possess their own memory unit and CPU.

These are referred to as loosely coupled systems or distributed systems. These system's processors differ in size and function.

The major benefit of working with these types of the operating system is that it is always possible that one user can access the files or software which are not actually present on his system but some other system connected within this network i.e., remote access is enabled within the devices connected in that network.

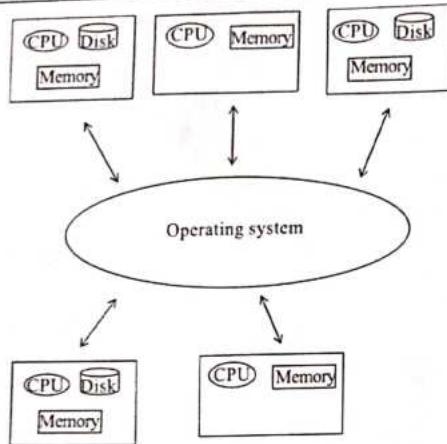


FIG 1.4 :

Advantages of Distributed Operating System :

- Failure of one will not affect the other network communication, as all systems are independent from each other
- Electronic mail increases the data exchange speed
- Since resources are being shared, computation is highly fast and durable
- Load on host computer reduces
- These systems are easily scalable as many systems can be easily added to the network
- Delay in data processing reduces

Disadvantages of Distributed Operating System:

- Failure of the main network will stop the entire communication
- To establish distributed systems the language which is used are not well defined yet
- These types of systems are not readily available as they are very expensive. Not only that the underlying software is highly complex and not understood well yet

Examples of Distributed Operating System are- LOCUS, etc.

4. Network Operating System : These systems run on a server and provide the capability to manage data, users, groups, security, applications, and other networking functions.

These types of operating systems allow shared access of files, printers, security, applications, and other networking functions over a small private network.

One more important aspect of Network Operating Systems is that all the users are well aware of the underlying configuration, of all other users within the network, their individual connections, etc. and that's why these computers are popularly known as **tightly coupled systems**.

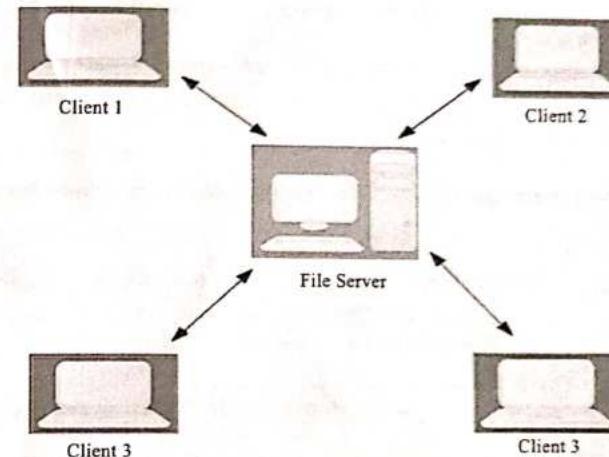


FIG 1.5 :

Advantages of Network Operating System :

- Highly stable centralized servers.
- Security concerns are handled through servers.
- New technologies and hardware up-gradation are easily integrated into the system.
- Server access is possible remotely from different locations and types of systems.

Disadvantages of Network Operating System:

- Servers are costly
- User has to depend on a central location for most operations
- Maintenance and updates are required regularly

Examples of Network Operating System are: Microsoft Windows Server 2003, Microsoft Windows Server 2008, UNIX, Linux, Mac OS X, Novell NetWare, and BSD, etc.

5. Real-Time Operating System : These types of OSs serve real-time systems. The time interval required to process and respond to inputs is very small. This time interval is called **response time**.

Real-Time Systems are used when there are time requirements that are very strict like missile systems, air traffic control systems, robots, etc.

Two types of Real-Time Operating System which are as follows :

Hard Real-time Systems : These OSs are meant for applications where time constraints are very strict and even the shortest possible delay is not acceptable.

- These systems are built for saving life like automatic parachutes or airbags which are required to be readily available in case of any accident. Virtual memory is rarely found in these systems.

Soft Real-time Systems : These OSs are for applications where time-constraint is less strict.

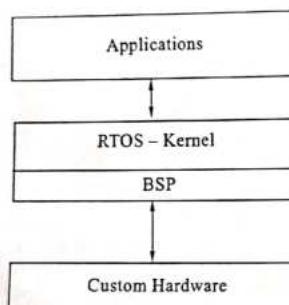


FIG 1.6 :

Advantages of RTOS :

- **Maximum Consumption** : Maximum utilization of devices and system, thus more output from all the resources
- **Task Shifting** : The time assigned for shifting tasks in these systems are very less. For example, in older systems, it takes about 10 microseconds in shifting one task to another, and in the latest systems, it takes 3 microseconds.
- **Focus on Application** : Focus on running applications and less importance to applications which are in the queue.
- **Real-time operating system in the embedded system** : Since the size of programs are small, RTOS can also be used in embedded systems like in transport and others.
- **Error Free** : These types of systems are error-free.
- **Memory Allocation** : Memory allocation is best managed in these types of systems.

Disadvantages OF RTOS :

- **Limited Tasks** : Very few tasks run at the same time and their concentration is very less on few applications to avoid errors.
- **Use heavy system resources** : Sometimes the system resources are not so good and they are expensive as well.
- **Complex Algorithms** : The algorithms are very complex and difficult for the designer to write on.
- **Device driver and interrupt signals** : It needs specific device drivers and interrupts signals to respond earliest to interrupts.
- **Thread Priority** : It is not good to set thread priority as these systems are very less prone to switching tasks.

Examples of Real-Time Operating Systems are: Scientific experiments, medical imaging systems, industrial control systems, weapon systems, robots, air traffic control systems, etc.

1.3 DISTINGUISH SPOOLING AND BUFFERING

	SPOOLING	BUFFERING
Basic Difference	It overlaps the input/output of one job with the execution of another job.	It overlaps the input/output of one job with the execution of the same job.
Full form (stands for)	Simultaneous peripheral operation online	No full form
Efficiency	Spooling is more efficient than buffering.	Buffering is less efficient than spooling.
Consider Size	It considers disk as a huge spool or buffer.	Buffer is a limited area in main memory.
Remote Processing	It can process data at remote places.	It does not support remote processing.

1.4 GOALS OF OPERATING SYSTEM

- There are two types of goals of an Operating System i.e. Primary Goals and Secondary Goal.
- 1) **Primary Goal** : The primary goal of an Operating System is to provide a user-friendly and convenient environment. 2) **Efficient utilization of the resources**
- We know that it is not compulsory to use the Operating System, but things become harder when the user has to perform all the process scheduling and converting the user code into machine code is also very difficult. So, we make the use of an Operating System to act as an intermediate between us and the hardware.

- All you need to do is give commands to the Operating System and the Operating System will do the rest for you. So, the Operating System should be convenient to use.

2) **Secondary Goal :** The secondary goal of an Operating System is efficiency.

- The Operating System should perform all the management of resources in such a way that the resources are fully utilised and no resource should be held idle if some request to that resource is there at that instant of time.

1.5 COMPONENTS OF OPERATING SYSTEM

An operating system is a large and complex system that can only be created by partitioning into small parts.

These pieces should be a well-defined part of the system, carefully defining inputs, outputs, and functions.

The components of an operating system play a key role to make a variety of computer system parts work together. There are the following components of an operating system, such as:

- | | |
|---------------------------------|-------------------------------|
| 1. Process Management | 2. File Management |
| 3. Network Management | 4. Main Memory Management |
| 5. Secondary Storage Management | 6. I/O Device Management |
| 7. Security Management | 8. Command Interpreter System |

Operating system components help you get the correct computing by detecting CPU and memory hardware errors.

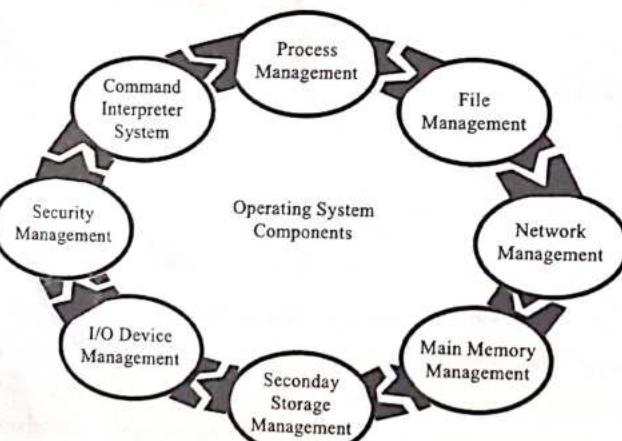


FIG 1.7 :

- Process Management :** The process management component is a procedure for managing many processes running simultaneously on the operating system. Every running software application program has one or more processes associated with them.

For example, when you use a search engine like Chrome, there is a process running for that browser program.

Process management keeps processes running efficiently. It also uses memory allocated to them and shutting them down when needed.

The execution of a process must be sequential so, at least one instruction should be executed on behalf of the process.

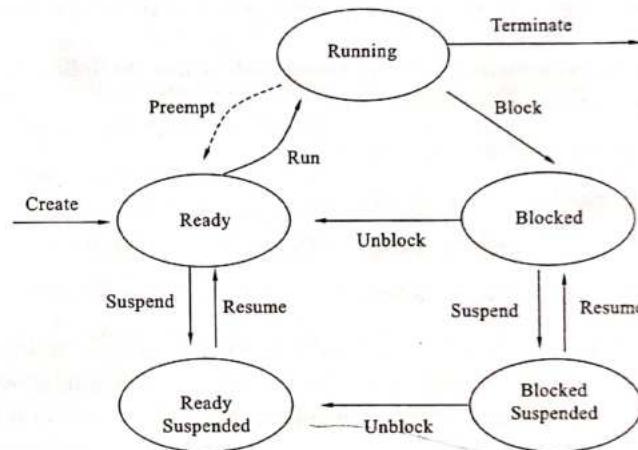


FIG 1.8 :

Functions of Process Management : Here are the following functions of process management in the operating system, such as:

- Process creation and deletion.
- Suspension and resumption.
- Synchronization process
- Communication process

- File Management :** A file is a set of related information defined by its creator. It commonly represents programs (both source and object forms) and data. Data files can be alphabetic, numeric, or alphanumeric.

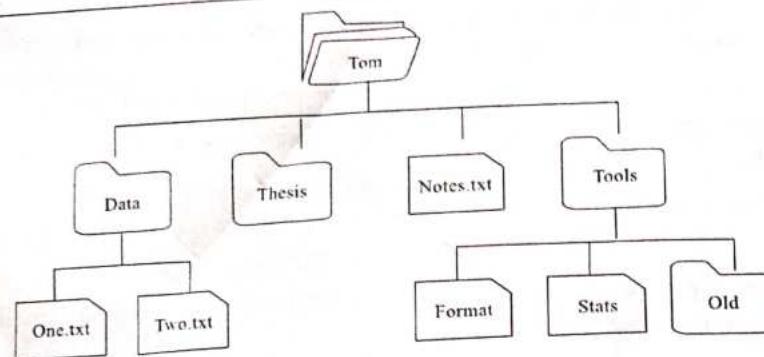


FIG 1.9

Function of File Management : The operating system has the following important activities in connection with file management :

- File and directory creation and deletion.
 - For manipulating files and directories.
 - Mapping files onto secondary storage.
 - Backup files on stable storage media.
3. **Network Management :** Network management is the process of administering and managing computer networks. It includes performance management, provisioning of networks, fault analysis, and maintaining the quality of service.

Computer Networks
When we hook up computers together using data communication facilities, we call this a computer network.

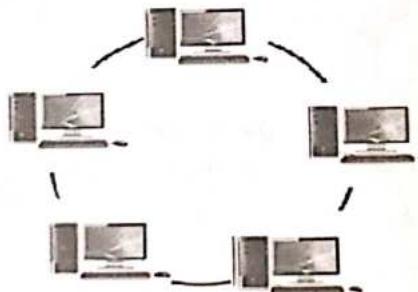


FIG 1.10 :

A distributed system is a collection of computers or processors that never share their memory and clock. In this type of system, all the processors have their local memory, and the processors communicate with each other using different communication cables, such as fibre optics or telephone lines.

The computers in the network are connected through a communication network, which can configure in many different ways.

The network can fully or partially connect in network management, which helps users design routing and connection strategies that overcome connection and security issues.

Functions of Network Management : Network management provides the following functions, such as :

- Distributed systems help you to various computing resources in size and function. They may involve minicomputers, microprocessors, and many general-purpose computer systems.
- A distributed system also offers the user access to the various resources the network shares.
- It helps to access shared resources that help computation to speed up or offers data availability and reliability.

4. **Main Memory Management** (Main memory is a large array of storage or bytes, which has an address. The memory management process is conducted by using a sequence of reads or writes of specific memory addresses.)

It should be mapped to absolute addresses and loaded inside the memory to execute a program. The selection of a memory management method depends on several factors.

However, it is mainly based on the hardware design of the system. Each algorithm requires corresponding hardware support. Main memory offers fast storage that can be accessed directly by the CPU. It is costly and hence has a lower storage capacity. However, for a program to be executed, it must be in the main memory.

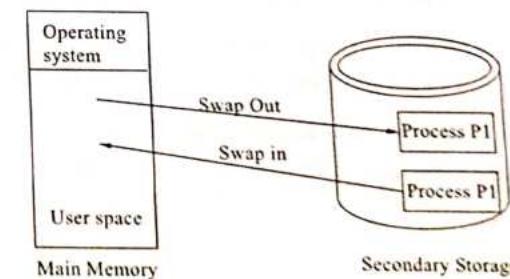


FIG 1.11 :

Functions of Memory Management : An Operating System performs the following functions for Memory Management in the operating system :

- It helps you to keep track of primary memory.
- Determine what part of it are in use by whom, what part is not in use.
- In a multiprogramming system, the OS decides which process will get memory and how much.
- Allocates the memory when a process requests.
- It also de-allocates the memory when a process no longer requires or has been terminated.

5. Secondary-Storage Management : The most important task of a computer system is to execute programs. These programs help you to access the data from the main memory during execution.

This memory of the computer is very small to store all data and programs permanently.

The computer system offers secondary storage to back up the main memory.

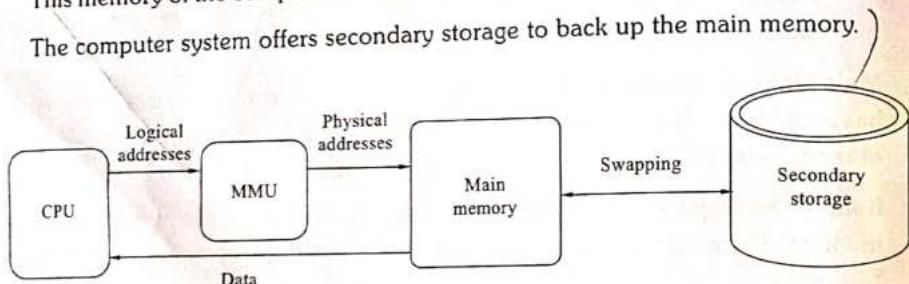


FIG 1.12 :

Today modern computers use hard drives/SSD as the primary storage of both programs and data. However, the secondary storage management also works with storage devices, such as USB flash drives and CD/DVD drives.

Programs like assemblers and compilers are stored on the disk until it is loaded into memory, and then use the disk is used as a source and destination for processing.

Functions of Secondary Storage Management : Here are some major functions of secondary storage management in the operating system :

- Storage allocation
- Free space management
- Disk scheduling

6. I/O Device Management : One of the important use of an operating system that helps to hide the variations of specific hardware devices from the user.

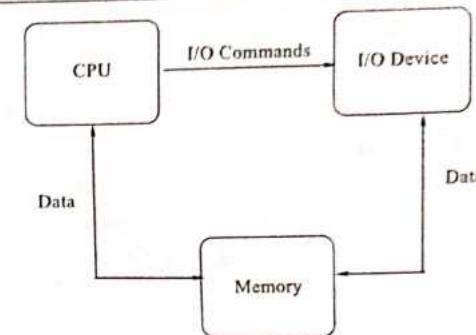


FIG 1.13 :

Functions of I/O Management : The I/O management system offers the following functions, such as :

- It offers a buffer caching system
- It provides general device driver code
- It provides drivers for particular hardware devices.

I/O helps you to know the individualities of a specific device.

7. Security Management : The various processes in an operating system need to be secured from other activities. Therefore, various mechanisms can ensure those processes that want to operate files, memory CPU, and other hardware resources should have proper authorization from the operating system.

Security refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by computer controls to be imposed, together with some means of enforcement.

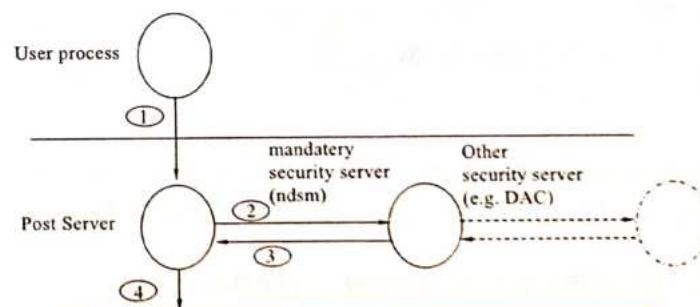
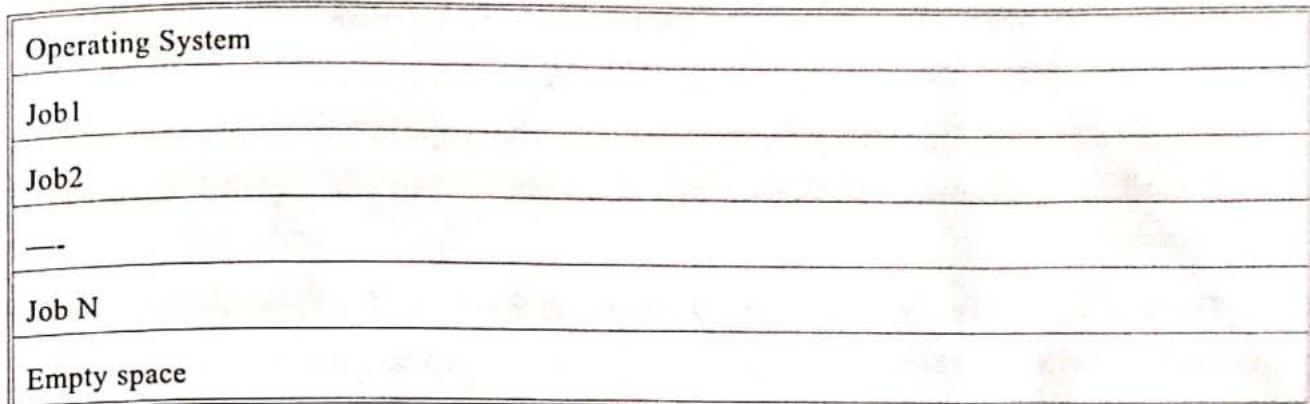


FIG 1.14 :

For example, memory addressing hardware helps to confirm that a process can be executed within its own address space.

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.

To increase resource utilization, multiprogramming approaches.



The OS could pick and start the execution of one of the jobs in memory, whenever the job 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.

Time - Sharing Operating Systems : Each task is given some time to execute so that all the tasks work smoothly. Each user gets the time of CPU as they use a single system.

These systems are also known as Multitasking Systems. The task can be from a single user or different users also.

The time that each task gets to execute is called quantum. After this time interval is over OS switches over to the next task.

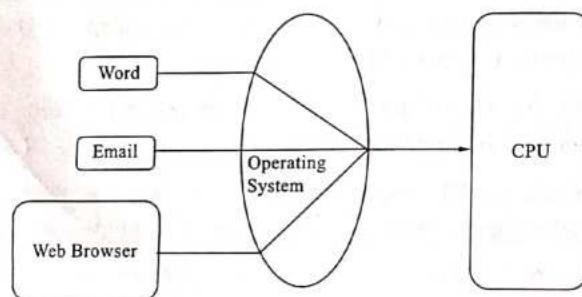


FIG 1.16 :

Advantages of Time-Sharing OS :

- Each task gets an equal opportunity
- Fewer chances of duplication of software
- CPU idle time can be reduced

Disadvantages of Time-Sharing OS :

- Reliability problem
- One must have to take care of the security and integrity of user programs and data
- Data communication problem

Examples of Time-Sharing OSs are: Multics, Unix, etc.

1.7 DIFFERENTIATE BETWEEN DISTRIBUTED AND REALTIME SYSTEMS

REAL-TIME OPERATING SYSTEM (RTOS)

- Real-time operating system (RTOS) is an operating system intended to serve real time application that process data as it comes in, mostly without buffer delay.
- The full form of RTOS is Real time operating system.
- In a RTOS, Processing time requirement are calculated in tenths of seconds increments of time. It is time-bound system that can be defined as fixed time constraints.
- In this type of system, processing must be done inside the specified constraints. Otherwise, the system will fail.

Distributed Operating System (DOS) :

- A distributed operating system is an essential type of operating system. Distributed systems use many central processors to serve multiple real-time applications and users. As a result, data processing jobs are distributed between the processors.
- It connects multiple computers via a single communication channel. Furthermore, each of these systems has its own processor and memory. Additionally, these CPUs communicate via high-speed buses or telephone lines. Individual systems that communicate via a single channel are regarded as a single entity. They're also known as **loosely coupled systems**.

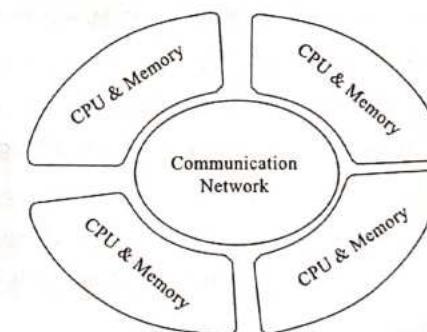


FIG 1.17:

1.8 MULTIPROCESSOR SYSTEM

A Multiprocessor is a computer system with two or more central processing units (CPUs) share full access to a common RAM.

The main objective of using a multiprocessor is to boost the system's execution speed, with other objectives being fault tolerance and application matching. There are two types of multiprocessors, one is called **shared memory multiprocessor** and another is **distributed memory multiprocessor**.

In shared memory multiprocessors, all the CPUs share the common memory but in a distributed memory multiprocessor, every CPU has its own private memory.

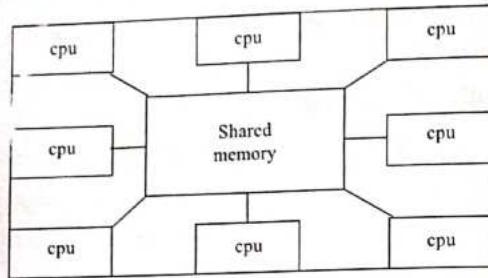


FIG 1.18 :

Applications of Multiprocessor –

- As a uniprocessor, such as single instruction, single data stream (SISD).
- As a multiprocessor, such as single instruction, multiple data stream (SIMD), which is usually used for vector processing.
- Multiple series of instructions in a single perspective, such as multiple instruction, single data stream (MISD), which is used for describing hyper-threading or pipelined processors.
- Inside a single system for executing multiple, individual series of instructions in multiple perspectives, such as multiple instruction, multiple data stream (MIMD).

Benefits of Using A Multiprocessor :

- Enhanced performance.
- Multiple applications.
- Multi-tasking inside an application.
- High throughput and responsiveness.
- Hardware sharing among CPUs.

Disadvantages of Multiprocessing Operating System :

- Operating system of multiprocessing is more complex and sophisticated as it takes care of multiple CPUs at the same time.

1.9 OPERATING SYSTEM SERVICES:

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system :

- | | |
|--|--|
| <ul style="list-style-type: none"> Program execution File System manipulation Error Detection Protection | <ul style="list-style-type: none"> I/O operations Communication Resource Allocation |
|--|--|

- Program Execution :** Operating systems handle many kinds of activities from user programs to system programs like printer spooler, name servers, file server, etc. Each of these activities is encapsulated as a process.

A process includes the complete execution context (code to execute, data to manipulate, registers, OS resources in use). Following are the major activities of an operating system with respect to program management -

- Loads a program into memory.
- Executes the program.
- Handles program's execution.
- Provides a mechanism for process synchronization.
- Provides a mechanism for process communication.
- Provides a mechanism for deadlock handling.

- I/O Operation :** An I/O subsystem comprises of I/O devices and their corresponding driver software. Drivers hide the peculiarities of specific hardware devices from the users.

An Operating System manages the communication between user and device drivers.

- I/O operation means read or write operation with any file or any specific I/O device.
- Operating system provides the access to the required I/O device when required.

3. 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.

4. 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 contention and 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.

5. 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.

6. 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.

7. 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.

1.10 DEFINE SYSTEM CALL WITH AN EXAMPLE

The interface between a process and an operating system is provided by system calls. In general, system calls are available as assembly language instructions.

They are also included in the manuals used by the assembly level programmers. System calls are usually made when a process in user mode requires access to a resource. Then it requests the kernel to provide the resource via a system call.

A figure representing the execution of the system call is given as follows “

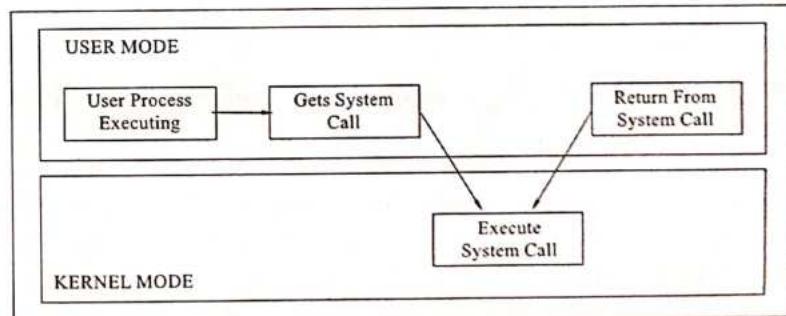


FIG 1.19 :

As can be seen from this diagram, the processes execute normally in the user mode until a system call interrupts this. Then the system call is executed on a priority basis in the kernel mode. After the execution of the system call, the control returns to the user mode and execution of user processes can be resumed.

In general, system calls are required in the following situations

- If a file system requires the creation or deletion of files. Reading and writing from files also require a system call.
- Creation and management of new processes.
- Network connections also require system calls. This includes sending and receiving packets.
- Access to a hardware devices such as a printer, scanner etc. requires a system call.

1.11 LIST DIFFERENT TYPES OF SYSTEM CALLS

There are mainly five types of system calls. These are explained in detail as follows “

Process Control : These system calls deal with processes such as process creation, process termination etc.

File Management : These system calls are responsible for file manipulation such as creating a file, reading a file, writing into a file etc.

Device Management : These system calls are responsible for device manipulation such as reading from device buffers, writing into device buffers etc.

Information Maintenance : These system calls handle information and its transfer between the operating system and the user program.

Communication : These system calls are useful for interprocess communication. They also deal with creating and deleting a communication connection.

Some of the examples of all the above types of system calls in Windows and Unix are given as follows

TYPES OF SYSTEM CALLS	WINDOWS	LINUX
Process Control	CreateProcess()ExitProcess()WaitForSingleObject()	fork()exit()wait()
File Management	CreateFile()ReadFile()WriteFile()CloseHandle()	open()read()write()close()
Device Management	SetConsoleMode()ReadConsole()WriteConsole()	ioctl()read()write()
Information Maintenance	GetCurrentProcessID()SetTimer()Sleep()	getpid()alarm()sleep()
Communication	CreatePipe()CreateFileMapping()MapViewOfFile()	pipe()shmget()mmap()

There are many different system calls as shown above. Details of some of those system calls are as follows -

open():

The open() system call is used to provide access to a file in a file system. This system call allocates resources to the file and provides a handle that the process uses to refer to the file. A file can be opened by multiple processes at the same time or be restricted to one process. It all depends on the file organisation and file system.

read():

The read() system call is used to access data from a file that is stored in the file system. The file to read can be identified by its file descriptor and it should be opened using open() before it can be read. In general, the read() system calls takes three arguments i.e. the file descriptor, buffer which stores read data and number of bytes to be read from the file.

write():

The write() system calls writes the data from a user buffer into a device such as a file. This system call is one of the ways to output data from a program. In general, the write system calls takes three arguments i.e. file descriptor, pointer to the buffer where data is stored and number of bytes to write from the buffer.

close():

The close() system call is used to terminate access to a file system. Using this system call means that the file is no longer required by the program and so the buffers are flushed, the file metadata is updated and the file resources are de-allocated.

1.12 DEFINE SINGLE USER, MULTI USER OPERATING SYSTEM STRUCTURES

There are four types of operating systems -

- Real-time operating system
- Single-User/Single-Tasking operating system
- Single-User/Multitasking operating system
- Multi-User/Multitasking operating system

REAL-TIME OPERATING SYSTEM

Real-time operating system is designed to run real-time applications. It can be both single- and multi-tasking. Examples include Abbasi, AMX RTOS, etc.



FIG 1.20 :

Advantages :

- It works very fast.
- It is time saving, as it need not be loaded from memory.
- Since it is very small, it occupies less space in memory.

Single - User / Single - Tasking OS : An operating system that allows a single user to perform only one task at a time is called a Single-User Single-Tasking Operating System. Functions like printing a document, downloading images, etc., can be performed only one at a time.

Examples include MS-DOS, Palm OS, etc.

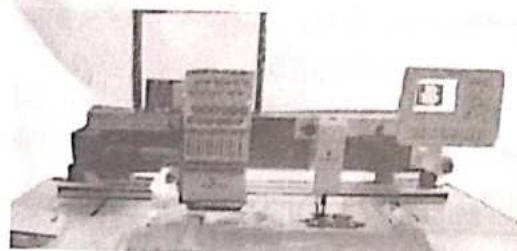


FIG 1.21 :

Advantages :

- This operating system occupies less space in memory.

Disadvantages :

- It can perform only a single task at a time.

Single - User/ Multitasking OS : An operating system that allows a single user to perform more than one task at a time is called Single-User Multitasking Operating System. Examples include Microsoft Windows and Macintosh OS.

*Xerox
by*

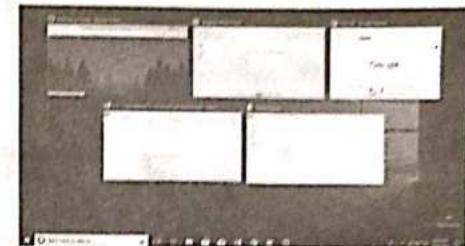


FIG 1.22 :

Advantages :

- It is time saving as it performs multiple tasks at a time yielding high productivity.

Disadvantages :

- This operating system is highly complex and occupies more space.

Multiuser / Multitasking OS Ultiuser/Multitasking OS

It is an operating system that permits several users to utilize the programs that are concurrently running on a single network server. The single network server is termed as "Terminal server". "Terminal client" is software that supports user sessions. Examples include UNIX, MVS, etc.

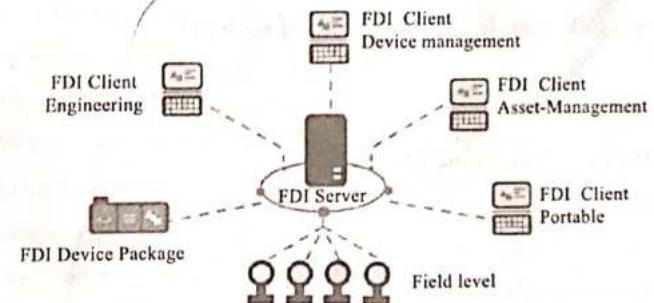


FIG 1.23 :

Advantages :

- It is highly productive as it performs multiple tasks at a time.
- It is time saving as we don't have to make changes in many desktops, instead can make changes only to the server.

Disadvantages :

- If the connection to the server is broken, user cannot perform any task on the client as it is connected to that server.

OBJECTIVE TYPE QUESTIONS

1. OS stands for
 - (a) Operating solve
 - (b) Open Source
 - (c) Open System
 - (d) Operating system

2. A Microsoft Windows is a(n)
 - (a) Operating system
 - (b) Graphic program
 - (c) Word Processing
 - (d) Database program

3. Which is not application software?
 - (a) Windows NT
 - (b) Page Maker
 - (c) WinWord XP
 - (d) Photoshop

4. The program compresses large files into a smaller file
 - (a) WinZip
 - (b) WinShrink
 - (c) WinStyle
 - (d) None of above

5. My Computer was introduced from
 - (a) Windows 3.1
 - (b) Windows 3.11
 - (c) Windows 95
 - (d) Windows 98

6. Which of the following is not an operating system?
 - (a) DOS
 - (b) Linux
 - (c) Windows
 - (d) Oracle

7. Linux is a(n) operating system
 - (a) Open source
 - (b) Microsoft
 - (c) Windows
 - (d) Mac

8. What is an operating system?
 - (a) interface between the hardware and application programs
 - (b) collection of programs that manages hardware resources
 - (c) system service provider to the application programs
 - (d) all of the mentioned

9. Which one of the following errors will be handle by the operating system?
 - (a) lack of paper in printer
 - (b) connection failure in the network
 - (c) power failure
 - (d) all of the mentioned

10. Which one of the following is not a real time operating system?
 - (a) RTLinux
 - (b) Palm OS
 - (c) QNX
 - (d) VxWorks

11. What are the services operating System provides to both the users and to the programs?
 - (a) File System manipulation
 - (b) Error Detection
 - (c) Program execution
 - (d) Resource Allocation

12. Which of the following few common services provided by an operating system?
 - (a) Protection
 - (b) Program execution
 - (c) I/O operations
 - (d) All of the above

13. Which of the following are examples of storage media?
 - (a) magnetic disk
 - (b) optical disk
 - (c) Both A and B
 - (d) None of the above

14. Which of the following is true about Program execution?
 - (a) Restrict to load a program into memory.
 - (b) Provides a mechanism for process synchronization.
 - (c) Do not provides a mechanism for process communication.
 - (d) Do not provides a mechanism for deadlock handling.

15. Which of the following is false about I/O Operation?
 - (a) Operating system does not provide the access to the I/O device
 - (b) I/O operation means read or write operation
 - (c) An I/O subsystem comprises of I/O devices
 - (d) None of the above

ANSWERS

1. (d)	2. (a)	3. (a)	4. (a)	5. (c)	6. (d)	7. (a)	8. (d)	9. (d)	10. (b)
11. (c)	12. (d)	13. (c)	14. (b)	15. (a)					

REVIEW QUESTIONS**Part-A**

1. Define operating system.
2. List the types of operating system.
3. What is spooling?
4. Define System call.
5. List the different types of system calls.

Part-B

1. Explain various functions of Operating System. (Oct/Nov. 2012, 2011)
2. Differentiate between distributed & real time operating system.
3. Describe about operating system services (Apr. 2009 ; Oct/Nov. 2010, 2008)
4. What is system call? Explain about system calls in detail. (Apr. 2009, 2008)
5. Describe about goals of operating system.
6. Write a short notes on
 - (a) Multi Processor system
 - (b) Time sharing
7. Explain about types of operating system. (Oct/Nov. 2013 ; Apr. 2009)
8. Difference between spooling and buffering.

CHAPTER**2****PROCESS MANAGEMENT***Chapter Outline*

-
- 2.0 INTRODUCTION
 - 2.1 PROCESS AND PROCES CONTROL BLOCK
 - 2.2 PROCEDSS STATE DIAGRAM
 - 2.3 PROCESS CREATION AND TERMINATION
 - 2.4 RELATIONS BETWEEN PROCESSES
 - 2.5 THREAD AND DESCRIBE MULTITHREADING
 - 2.6 SCHEDULING CONCEPT
 - 2.7 SCHEDULING QUEUES AND SCHEDULERS
 - 2.8 CPU SCHEDULING CRITERIA
 - 2.9 VARIOUS SCHEDULING ALGORITHMS
-

2.0 INTRODUCTION

Process management involves various tasks like creation, scheduling, termination of processes, and a deadlock. Process is a program that is under execution, which is an important part of modern-day operating systems.

The OS must allocate resources that enable processes to share and exchange information. It also protects the resources of each process from other methods and allows synchronization among processes.

It is the job of OS to manage all the running processes of the system. It handles operations by performing tasks like process scheduling and such as resource allocation.

2.1 DEFINE PROCESS AND PROCES CONTROL BLOCK

2.1.1 PROCESS

- ✓ A process is a program in execution which then forms the basis of all computation.
- ✓ The process is not as same as program code but a lot more than it.
- A process is an 'active' entity as opposed to the program which is considered to be a 'passive' entity. Attributes held by the process include hardware state, memory, CPU, etc.

2.1.2 PROCES CONTROL BLOCK

There is a Process Control Block for each process, enclosing all the information about the process. It is also known as the task control block.

It is a data structure, which contains the following :

- **Process State** : It can be running, waiting, etc.
- **Process ID** and the **parent process ID**.
- **CPU registers and Program Counter**. **Program Counter** holds the address of the next instruction to be executed for that process.
- **CPU Scheduling Information** : Such as priority information and pointers to scheduling queues.
- **Memory Management Information** : For example, page tables or segment tables.
- **Accounting Information** : The User and kernel CPU time consumed, account numbers, limits, etc.
- **I/O Status information**: Devices allocated, open file tables, etc.

Process ID
state
Pointer
Priority
Program counter
CPU registers
I/O information
Accounting information
etc...

FIG 2.1 :

2.2 PROCEDSS STATE DIAGRAM

Processes in the operating system can be in any of the following states:

- **NEW** : The process is being created.
- **READY** : The process is waiting to be assigned to a processor.
- **RUNNING** : Instructions are being executed.
- **WAITING** : The process is waiting for some event to occur(such as an I/O Completion or reception of a signal).
- **TERMINATED** : The process has finished execution.

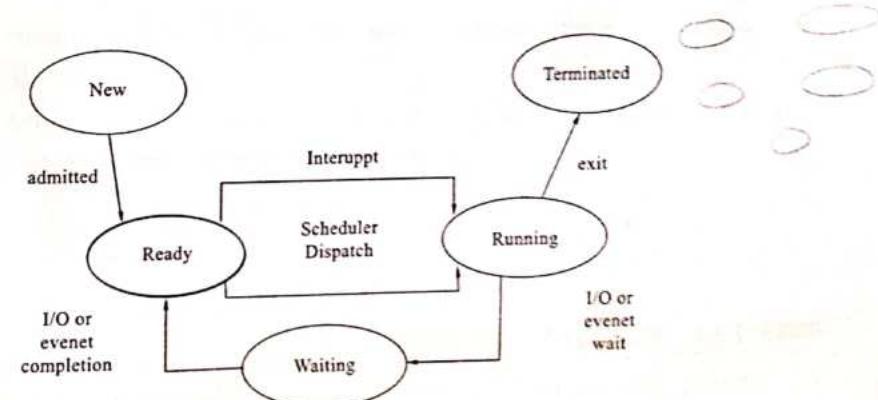


FIG 2.2 :

2.3 PROCESS CREATION AND TERMINATION

2.3.1 PROCESS CREATION

A process may be created in the system for different operations.

Some of the events that lead to process creation are as follows :

- User request for process creation
- System Initialization
- Batch Job Initialization
- Execution of a process creation system call by a running process

A process may be created by another process using `fork()`. The creating process is called the parent process and the created process is the child process.

A child process can have only one parent but a parent process may have many children. Both the parent and child processes have the same memory image, open files and environment strings. However, they have distinct address spaces.

A diagram that demonstrates process creation using `fork()` is as follows :

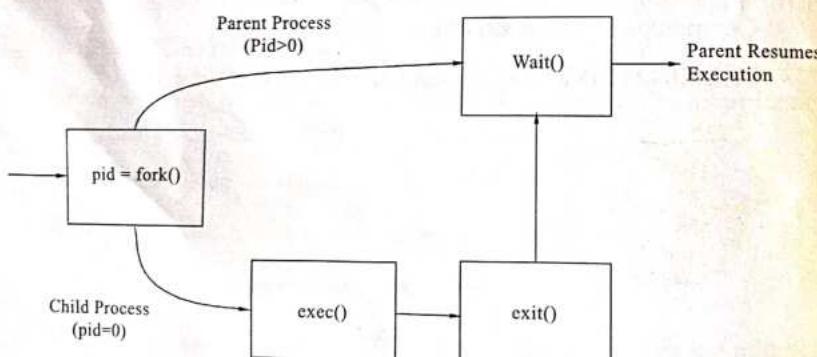


FIG 2.3 : Process Creation Using `for()`

2.3.2 PROCESS TERMINATION

Process termination occurs when the process is terminated. The `exit()` system call is used by most operating systems for process termination.

Some of the causes of process termination are as follows :

- A process may be terminated after its execution is naturally completed. This process leaves the processor and releases all its resources.

- A child process may be terminated if its parent process requests for its termination.
- A process can be terminated if it tries to use a resource that it is not allowed to. **For example :** A process can be terminated for trying to write into a read only file.
- If an I/O failure occurs for a process, it can be terminated. **For example :** If a process requires the printer and it is not working, then the process will be terminated.
- In most cases, if a parent process is terminated then its child processes are also terminated. This is done because the child process cannot exist without the parent process.
- If a process requires more memory than is currently available in the system, then it is terminated because of memory scarcity.

2.4 RELATIONS BETWEEN PROCESSES

Process memory is divided into four sections for efficient working :

- The **Text** section is made up of the compiled program code, read in from non-volatile storage when the program is launched.
- The **Data** section is made up of the global and static variables, allocated and initialized prior to executing the main.
- The **Heap** is used for the dynamic memory allocation and is managed via calls to new, delete, malloc, free, etc.
- The **Stack** is used for local variables. Space on the stack is reserved for local variables when they are declared.

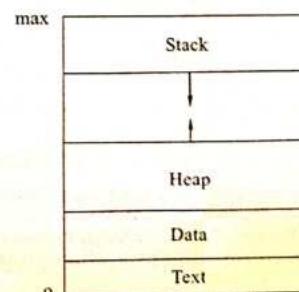


FIG 2.4 : Process in the Memory

2.5 THREAD AND DESCRIBE MULTITHREADING

2.5.1 THREAD

A thread is a flow of execution through the process code, with its own program counter that keeps track of which instruction to execute next, system registers which hold its current working variables, and a stack which contains the execution history.

A thread is also called a **lightweight process**. Threads provide a way to improve application performance through parallelism.

Threads represent a software approach to improving performance of operating system by reducing the overhead thread is equivalent to a classical process.

Each thread belongs to exactly one process and no thread can exist outside a process.

Each thread represents a separate flow of control. Threads have been successfully used in implementing network servers and web server.

They also provide a suitable foundation for parallel execution of applications on shared memory multiprocessors.

The following figure shows the working of a single-threaded and a multithreaded process.

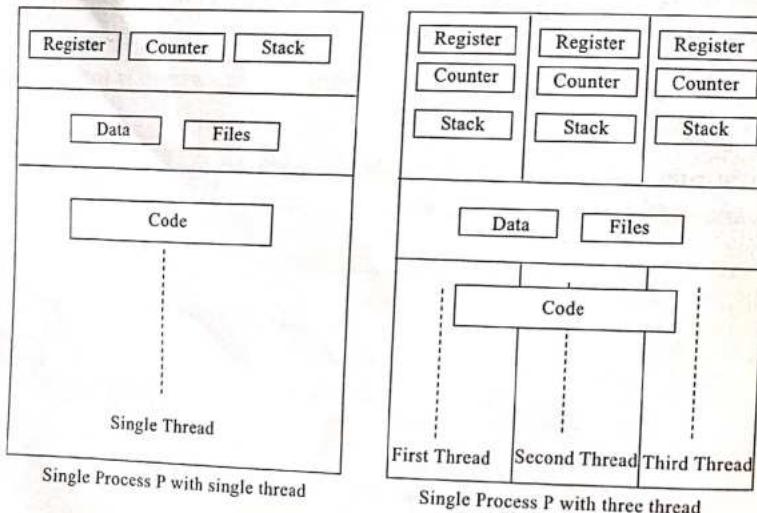


FIG 2.5 :

2.5.2 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.

User Level Threads : In this case, the thread management kernel is not aware of the existence of threads.

The **thread library** contains **code** for creating and destroying threads, for passing message and data between threads, for scheduling thread execution and for saving and restoring thread contexts. The application starts with a single thread.

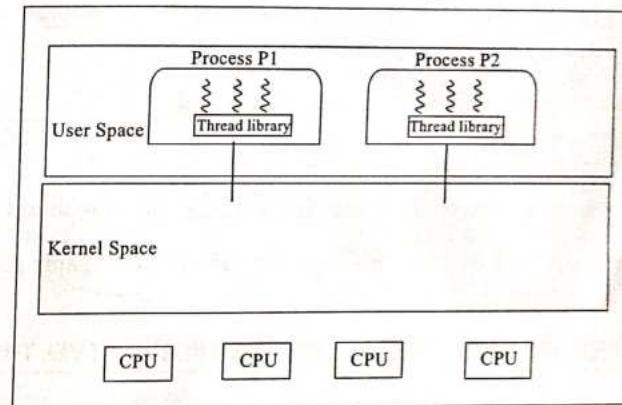


FIG 2.6 :

Advantages :

- Thread switching does not require **Kernel mode** privileges.
- **User level thread** can run on any operating system.
- Scheduling can be application specific in the user level thread.
- User level threads are fast to create and manage

Disadvantages :

- In a typical operating system, most system calls are blocking.
- Multithreaded application cannot take advantage of multiprocessing.

Kernel Level Threads : In this case, thread management is done by the Kernel. There is no thread management code in the application area.

Kernel threads are supported directly by the operating system. Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.

The Kernel maintains context information for the process as a whole and for individuals threads within the process. Scheduling by the Kernel is done on a thread basis.

The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages :

- Kernel can simultaneously schedule multiple threads from the same process or multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.
- Kernel routines themselves can be multithreaded.

Disadvantages :

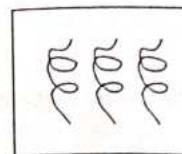
- Kernel threads are generally slower to create and manage than the user threads.
- Transfer of control from one thread to another within the same process requires a mode switch to the Kernel.

2.5.3 DIFFERENCE BETWEEN USER-LEVEL AND KERNEL-LEVEL THREAD

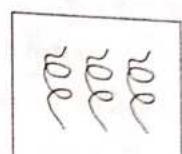
S.No.	User-level Threads	Kernel-level Thread
1.	User-level threads are faster to create and manage.	Kernel-level threads are slower to create and manage.
2.	Implementation is by a thread library at the user level.	Operating system supports creation of Kernel threads.
3.	User-level thread is generic and can run on any operating system.	Kernel-level thread is specific to the operating system.
4.	Multi-threaded applications cannot take advantage of multiprocessor.	Kernel routines themselves can be multithreaded.

2.5.4 MULTITHREADING MODEL

Multithreading allows the application to divide its task into individual threads. In multithreads, the same process or task can be done by the number of threads, or we can say that there is more than one thread to perform the task in multithreading. With the use of multithreading, multitasking can be achieved.



One Process
Multiple Threads



Multiple Process
Multiple
Threads per Process

FIG 2.7 :

The main drawback of single threading systems is that only one task can be performed at a time, so to overcome the drawback of this single threading, there is multithreading that allows multiple tasks to be performed.

For Example :

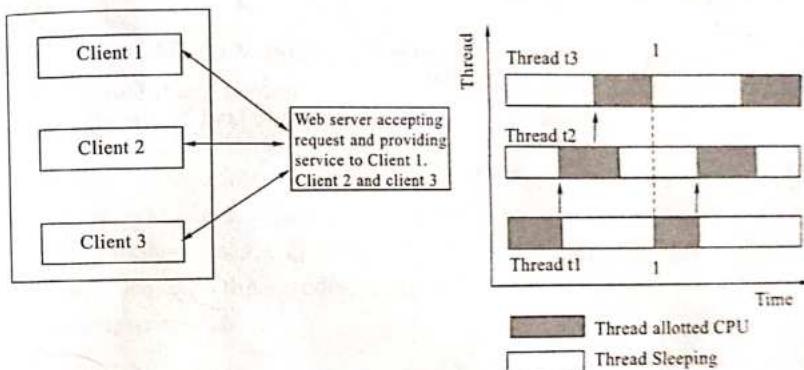


FIG 2.8 :

In the above example, client1, client2, and client3 are accessing the web server without any waiting. In multithreading, several tasks can run at the same time.

In an **operating system**, threads are divided into the user-level thread and the Kernel-level thread. User-level threads handle independently from the kernel and thereby managed without any kernel support. On the opposite hand, the operating system directly manages the kernel-level threads. Nevertheless, there must be a form of relationship between user-level and kernel-level threads.

There exists three established multithreading models classifying these relationships are :

- Many to one multithreading model

- One to one multithreading model
- Many to Many multithreading models

1. Many To One Multithreading Model : The many to one model maps many user level threads to one kernel thread.

This type of relationship facilitates an effective context-switching environment, easily implemented even on the simple kernel with no thread support.

The disadvantage of this model is that since there is only one kernel-level thread schedule at any given time, this model cannot take advantage of the hardware acceleration offered by multithreaded processes or multi-processor systems.

In this, all the thread management is done in the user space. If blocking comes, this model blocks the whole system.

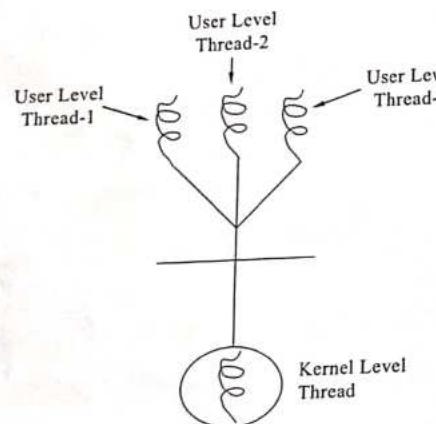


FIG 2.9 : Many-to-One Model

In the above figure, the many to one model associates all user-level threads to single kernel-level threads.

2. One to One Multithreading Model : The one-to-one model maps a single user-level thread to a single kernel-level thread. This type of relationship facilitates the running of multiple threads in parallel. However, this benefit comes with its drawback.

The generation of every new user thread must include creating a corresponding kernel thread causing an overhead, which can hinder the performance of the parent process. Windows series and Linux operating systems try to tackle this problem by limiting the growth of the thread count.

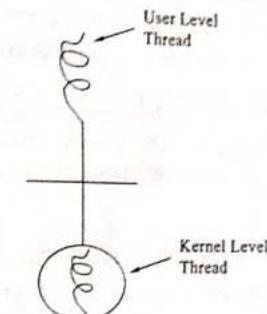


FIG 2.10 : One-to-One Model

In the above figure, one model associates that one user-level thread to a single kernel-level thread.

3. Many to Many Model Multithreading Model : In this type of model, there are several user-level threads and several kernel-level threads. The number of kernel threads created depends upon a particular application.

The developer can create as many threads at both levels but may not be the same. The many to many model is a compromise between the other two models. In this model, if any thread makes a blocking system call, the kernel can schedule another thread for execution. Also, with the introduction of multiple threads, complexity is not present as in the previous models.

Though this model allows the creation of multiple kernel threads, true concurrency cannot be achieved by this model. This is because the kernel can schedule only one process at a time.

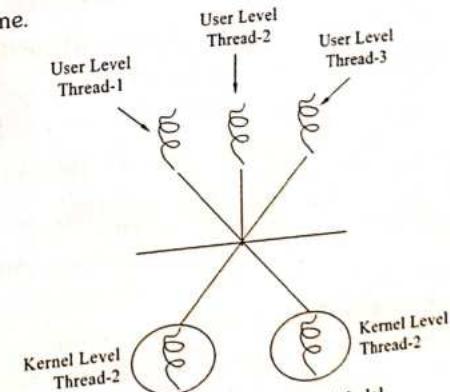


FIG 2.11 : Many-to-Many Model

Many to many versions of the multithreading model associate several user-level threads to the same or much less variety of kernel-level threads in the above figure.

2.6 SCHEDULING CONCEPT

The process scheduling is the activity of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.

Process scheduling is an essential part of Multiprogramming operating systems. Such operating systems allow more than one process to be loaded into the executable memory at a time and the loaded process shares the CPU using time multiplexing.

2.7 SCHEDULING QUEUES AND SCHEDULERS

2.7.1 PROCESSES SCHEDULING QUEUES

The OS maintains all PCBs in Process Scheduling Queues.

The OS maintains a separate queue for each of the process states and PCBs of all processes in the same execution state are placed in the same queue.

When the state of a process is changed, its PCB is unlinked from its current queue and moved to its new state queue.

The Operating System maintains the following important process scheduling queues :

- Job Queue : This queue keeps all the processes in the system.
- Ready Queue : This queue keeps a set of all processes residing in main memory, ready and waiting to execute. A new process is always put in this queue.
- Device Queues : The processes which are blocked due to unavailability of an I/O device constitute this queue.

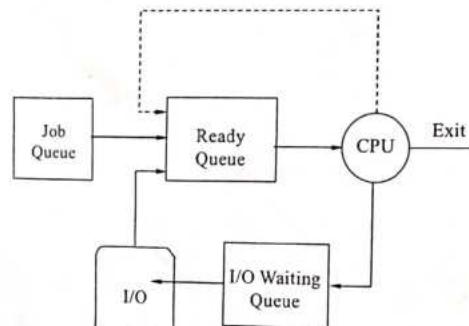


FIG 2.12 : Many-to-Many Model

The OS can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.). The OS scheduler determines how to move processes between the ready and run queues which can only have one entry per processor core on the system; in the above diagram, it has been merged with the CPU.

Two-State Process Model : Two-state process model refers to running and non-running states which are described below :

S.No.	State and Description
1.	Running : When a new process is created, it enters into the system as in the running state.
2.	Not Running : Processes that are not running are kept in queue, waiting for their turn to execute. Each entry in the queue is a pointer to a particular process. Queue is implemented by using linked list. Use of dispatcher is as follows. When a process is interrupted, that process is transferred in the waiting queue. If the process has completed or aborted, the process is discarded. In either case, the dispatcher then selects a process from the queue to execute.

2.7.2 SCHEDULERS

Schedulers are special system software which handle process scheduling in various ways. Their main task is to select the jobs to be submitted into the system and to decide which process to run.

Schedulers are of Three Types :

- Long-Term Scheduler.
- Short-Term Scheduler.
- Medium-Term Scheduler.

1. Long Term Scheduler :

- It is also called a job scheduler. A long-term scheduler determines which programs are admitted to the system for processing.
- It selects processes from the queue and loads them into memory for execution.
- Process loads into the memory for CPU scheduling.
- The primary objective of the job scheduler is to provide a balanced mix of jobs, such as I/O bound and processor bound.
- It also controls the degree of multiprogramming. If the degree of multiprogramming is stable, then the average rate of process creation must be equal to the average departure rate of processes leaving the system.

- On some systems, the long-term scheduler may not be available or minimal.
- Time-sharing operating systems have no long term scheduler. When a process changes the state from new to ready, then there is use of long-term scheduler.

2. Short Term Scheduler :

- It is also called as CPU scheduler. Its main objective is to increase system performance in accordance with the chosen set of criteria.
- It is the change of ready state to running state of the process. CPU scheduler selects a process among the processes that are ready to execute and allocates CPU to one of them.
- Short-term schedulers, also known as dispatchers, make the decision of which process to execute next. Short-term schedulers are faster than long-term schedulers.

3. Medium Term Scheduler :

- Medium-term scheduling is a part of **swapping**. It removes the processes from the memory. It reduces the degree of multiprogramming.
- The medium-term scheduler is in-charge of handling the swapped out-processes.
- A running process may become suspended if it makes an I/O request.
- A suspended process cannot make any progress towards completion. In this condition, to remove the process from memory and make space for other processes, the suspended process is moved to the secondary storage.
- This process is called **swapping**, and the process is said to be swapped out or rolled out. Swapping may be necessary to improve the process mix.

*5m
differently*

S.No.	Long-term Scheduler	Short-term Scheduler	Medium-term Scheduler
1.	It is a job scheduler	It is a CPU scheduler	It is a process swapping scheduler.
2.	Speed is lesser than short term scheduler	Speed is fastest among other two	Speed is in between both short and long term scheduler.
3.	It controls the degree of multi programming	It provides lesser control over degree of multiprogramming	It reduces the degree of multiprogramming.
4.	It is almost absent or minimal in time sharing system	It is also minimal in time sharing system	It is a part of Time sharing systems.
5.	It selects processes from pool and loads them into memory for execution	It selects those processes which are ready to execute	It can re-introduce the process into memory and execution can be continued.

WARNING

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

2.8 CPU SCHEDULING CRITERIA 5m

5m Different CPU scheduling algorithms have different properties and the choice of a particular algorithm depends on the various factors.

Many criteria have been suggested for comparing CPU scheduling algorithms.

The criteria include the following :

1. **CPU Utilisation:** The main objective of any CPU scheduling algorithm is to keep the CPU as busy as possible. Theoretically, CPU utilisation can range from 0 to 100 but in a real-time system, it varies from 40 to 90 percent depending on the load upon the system.
2. **Throughput :** A measure of the work done by the CPU is the number of processes being executed and completed per unit of time. This is called throughput. The throughput may vary depending upon the length or duration of the processes.
3. **Turnaround Time :** For a particular process, an important criterion is how long it takes to execute that process. The time elapsed from the time of submission of a process to the time of completion is known as the turnaround time. Turn-around time is the sum of times spent waiting to get into memory, waiting in the ready queue, executing in CPU, and waiting for I/O. The formula to calculate Turn Around Time = Compilation Time – Arrival Time.
4. **Waiting Time :** A scheduling algorithm does not affect the time required to complete the process once it starts execution. It only affects the waiting time of a process i.e. time spent by a process waiting in the ready queue. The formula for calculating Waiting Time = Turnaround Time – Burst Time.
5. **Response Time :** In an interactive system, turn-around time is not the best criteria. A process may produce some output fairly early and continue computing new results while previous results are being output to the user. Thus another criteria is the time taken from submission of the process of request until the first response is produced. This measure is called response time. The formula to calculate Response Time = CPU Allocation Time (when the CPU was allocated for the first) – Arrival Time
6. **Completion Time :** This is the time when the process completes its execution.

2.9 VARIOUS SCHEDULING ALGORITHMS

There are mainly two types of scheduling methods :

- **Preemptive Scheduling :** Preemptive scheduling is used when a process switches from running state to ready state or from the waiting state to the ready state.

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

WARNING

- Non-preemptive Scheduling : Non-Preemptive scheduling is used when a process terminates , or when a process switches from running state to waiting state.

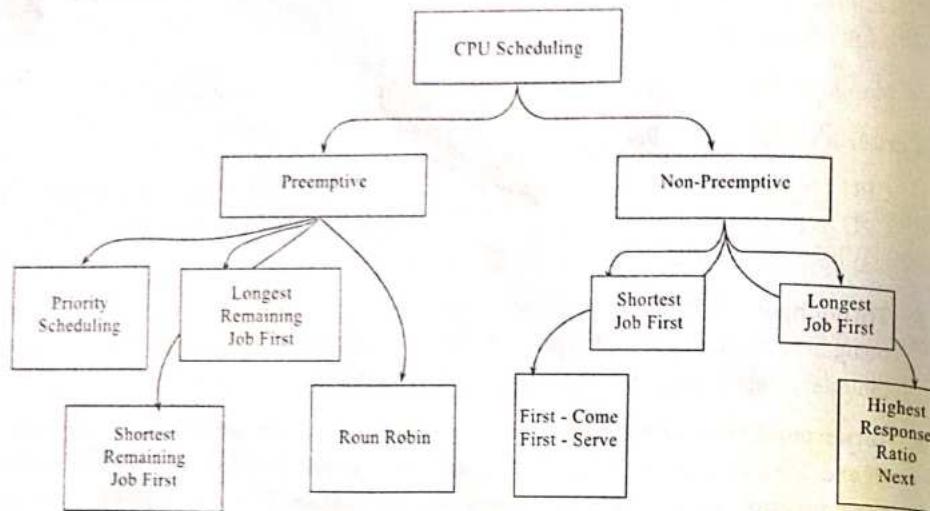


FIG 2.13 : Different Types of CPU Scheduling Algorithms

2.9.1 FIRST COME FIRST SERVE

FCFS considered to be the simplest of all operating system scheduling algorithms. First come first serve scheduling algorithm states that the process that requests the CPU first is allocated the CPU first and is implemented by using FIFO queue.

Characteristics of FCFS :

- FCFS supports non-preemptive and preemptive CPU scheduling algorithms.
- Tasks are always executed on a First-come, First-serve concept.
- FCFS is easy to implement and use.
- This algorithm is not much efficient in performance, and the wait time is quite high.

Advantages of FCFS :

- Easy to implement
- First come, first serve method

Disadvantages of FCFS :

- FCFS suffers from Convoy effect.

- The average waiting time is much higher than the other algorithms.
- FCFS is very simple and easy to implement and hence not much efficient.

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on *First come, First serve Scheduling*.

2.9.2 SHORTEST JOB FIRST (SJF)

Shortest Job First (SJF) is a scheduling process that selects the waiting process with the smallest execution time to execute next. This scheduling method may or may not be preemptive. Significantly reduces the average waiting time for other processes waiting to be executed. The full form of SJF is Shortest Job First.

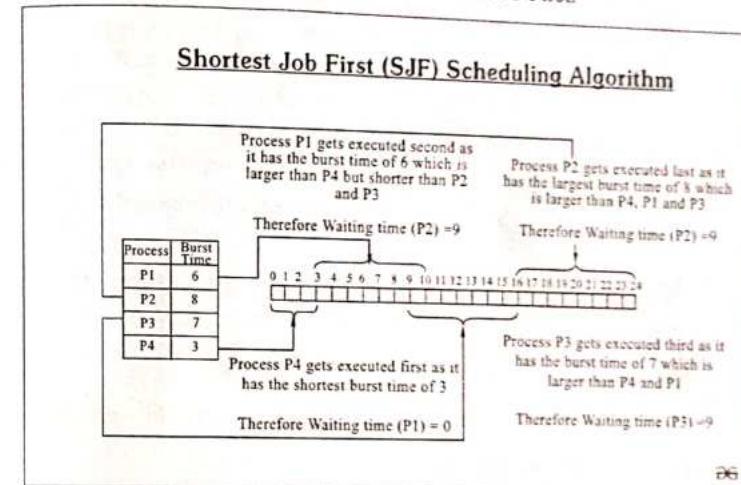


FIG 2.14 :

Characteristics of SJF :

- Shortest Job first has the advantage of having a minimum average waiting time among all *operating system scheduling algorithms*.
- It is associated with each task as a unit of time to complete.
- It may cause starvation if shorter processes keep coming. This problem can be solved using the concept of ageing.

Advantages of Shortest Job First :

- As SJF reduces the average waiting time thus, it is better than the first come first serve scheduling algorithm.
- SJF is generally used for long term scheduling

Disadvantages of SJF :

- One of the demerit SJF has is starvation.
- Many times it becomes complicated to predict the length of the upcoming CPU request

To learn about how to implement this CPU scheduling algorithm, please refer to our detailed article on *Shortest Job First*.

Characteristics of Round Robin :

- It's simple, easy to use, and starvation-free as all processes get the balanced CPU allocation.
- One of the most widely used methods in CPU scheduling as a core.
- It is considered preemptive as the processes are given to the CPU for a very limited time.

Advantages of Round Robin :

- Round robin seems to be fair as every process gets an equal share of CPU.
- The newly created process is added to the end of the ready queue.

2.9.5 MULTIPLE QUEUE SCHEDULING

Processes in the ready queue can be divided into different classes where each class has its own scheduling needs. For example, a common division is a **foreground (interactive)** process and a **background (batch)** process. These two classes have different scheduling needs. For this kind of situation **Multilevel Queue Scheduling** is used.

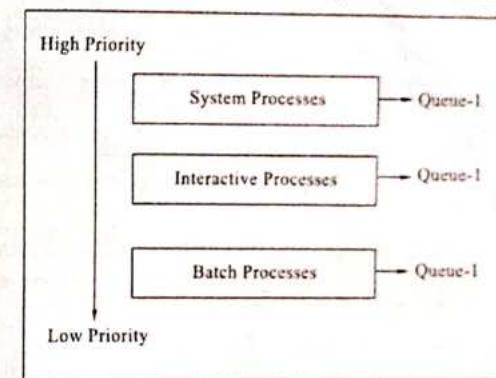


FIG 2.15 :

The description of the processes in the above diagram is as follows :

- **System Processes** : The CPU itself has its process to run, generally termed as System Process.
- **Interactive Processes** : An Interactive Process is a type of process in which there should be the same type of interaction.
- **Batch Processes** : Batch processing is generally a technique in the operating system that collects the programs and data together in the form of a batch before the processing starts.

2.9.4 ROUND ROBIN

Round Robin is a **CPU scheduling algorithm** where each process is cyclically assigned a fixed time slot. It is the **preemptive version of First come First Serve CPU Scheduling algorithm**. Round Robin CPU Algorithm generally focuses on Time Sharing technique.

Advantages of Multilevel Queue Scheduling :

- The main merit of the multilevel queue is that it has a low scheduling overhead.

Disadvantages of Multilevel Queue Scheduling :

- Starvation problem
- It is inflexible in nature

2.9.6 MULTILEVEL FEEDBACK QUEUE SCHEDULING

Multilevel Feedback Queue Scheduling (MLFQ) CPU Scheduling is like Multilevel Queue Scheduling but in this process can move between the queues. And thus, much more efficient than multilevel queue scheduling.

Characteristics of Multilevel Feedback Queue Scheduling :

- In a multilevel queue-scheduling algorithm, processes are permanently assigned to a queue on entry to the system, and processes are not allowed to move between queues.
- As the processes are permanently assigned to the queue, this setup has the advantage of low scheduling overhead,
- But on the other hand disadvantage of being inflexible.

Advantages of Multilevel Feedback Queue Scheduling :

- It is more flexible.
- It allows different processes to move between different queues.

Disadvantages of Multilevel Feedback Queue Scheduling :

- It also produces CPU overheads
- It is the most complex algorithm.

OBJECTIVE TYPE QUESTIONS

- The systems which allow only one process execution at a time, are called _____
 - (a) Uniprogramming systems
 - (b) Uniprocessing systems
 - (c) Unitasking systems
 - (d) None of the mentioned
- In operating system, each process has its own _____
 - (a) Address space and global variables
 - (b) Open files
 - (c) Pending alarms, signals and signal handlers
 - (d) All of the mentioned
- A process can be terminated due to _____
 - (a) Normal exit
 - (b) Fatal error
 - (c) Killed by another process
 - (d) All of the above
- What is the ready state of a process?
 - (a) When process is scheduled to run after some execution
 - (b) When process is unable to run until some task has been completed
 - (c) When process is using the CPU
 - (d) None of the mentioned
- OS access the services of operating system, the interface is provided by the _____
 - (a) System calls
 - (b) API
 - (c) Library
 - (d) Assembly instructions
- Which one of the following is not a real time operating system?
 - (a) Vx Works
 - (b) QNX
 - (c) RTLinux
 - (d) Palm OS
- Which of the following cannot be scheduled by the kernel?
 - (a) Process
 - (b) User-level thread
 - (c) Kernel-level thread
 - (d) None of the above

8. Scheduling algorithm In multilevel feedback

- (a) Processes are not classified into groups
- (b) A process can move to a different classified ready queue
- (c) Classification of the ready queue is permanent
- (d) None of the above

9. The process can be classified into many groups in

- (a) Shortest job scheduling algorithm
- (b) Multilevel queue scheduling algorithm
- (c) Round-robin scheduling algorithm
- (d) Priority scheduling algorithm

10. Preemptive Shortest Job First scheduling is sometimes called _____

- (a) Fast SJF scheduling
- (b) EDF scheduling – Earliest Deadline First
- (c) HRRN scheduling – Highest Response Ratio Next
- (d) SRTN scheduling – Shortest Remaining Time Next

11. Which module gives control of the CPU to the process selected by the short-term scheduler?

- (a) Dispatcher
- (b) Interrupt
- (c) Scheduler
- (d) None of the above

12. Which of the following do not belong to queues for processes?

- (a) Job Queue
- (b) PCB queue
- (c) Device Queue
- (d) Ready Queue

13. When the process issues an I/O request _____

- (a) It is placed in an I/O queue
- (b) It is placed in a waiting queue
- (c) It is placed in the ready queue
- (d) It is placed in the Job queue

ANSWERS

1. (b)	2. (d)	3. (d)	4. (a)	5. (a)	6. (d)	7. (b)	8. (b)	9. (b)	10. (d)
11. (a)	12. (b)	13. (a)							

Part-A

1. Define Process.
2. List the CPU scheduling algorithms.
3. Define scheduling concept.
4. What is a thread?.
5. What is Process Control Block?

Part-B

1. What is a process scheduler? State the characteristics of a good process scheduler?
2. What is Shortest Remaining Time, SRT scheduling?
3. Find out which algorithm among FCFS, SJF and Round Robin with quantum 10, would give the minimum average time for a given workload.
4. Describe the various types of scheduling algorithms in detail?
(Outline 2012-2014)
(Apr-May 2009, 2008)
5. Write short notes on scheduling queues.
6. Compare user threads & kernel threads.
7. What are the various scheduling criteria for CPU scheduling?
(Mar/Apr 2014, Apr/May 2012, April 2008)
8. Explain the difference between long term and short term and medium term schedulers.
(Oct/Nov 2013, Apr 2009)
9. What is a process? Draw and explain process state diagram.
(Mar/Apr 2013, Apr/May 2012)
10. What do you mean by PCB? Where is it used? What are its contents? Explain

CHAPTER

5

VIRTUAL MEMORY MANAGEMENT

Chapter Outline

- 5.0 INTRODUCTION
- 5.1 VIRTUAL MEMORY TECHNIQUES
- 5.2 DEMAND PAGING
- 5.3 PAGE REPLACEMENT
- 5.4 PAGE REPLACEMENT ALGORITHM
- 5.5 CONCEPT OF THRASHING
- 5.6 TECHNIQUES TO HANDLE

5.0 INTRODUCTION

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

In this scheme, User can load the bigger size processes than the available main memory by having the illusion that the memory is available to load the process.

Instead of loading one big process in the main memory, the Operating System loads the different parts of more than one process in the main memory.

By doing this, the degree of multiprogramming will be increased and therefore, the CPU utilization will also be increased.

5.1 DEFINE VIRTUAL MEMORY TECHNIQUES

The main visible advantage of this scheme is that programs can be larger than physical memory. Virtual memory serves two purposes. First, it allows us to extend the use of physical memory by using disk. Second, it allows us to have memory protection, because each virtual address is translated to a physical address.

Following are the situations, when entire program is not required to be loaded fully in main memory.

- User written error handling routines are used only when an error occurred in the data or computation.
- Certain options and features of a program may be used rarely.
- Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
- The ability to execute a program that is only partially in memory would counter many benefits.
- Less number of I/O would be needed to load or swap each user program into memory.
- A program would no longer be constrained by the amount of physical memory that is available.
- Each user program could take less physical memory, more programs could be run the same time, with a corresponding increase in CPU utilization and throughput.

Modern microprocessors intended for general-purpose use, a memory management unit, or MMU, is built into the hardware.

The MMU's job is to translate virtual addresses into physical addresses. A basic example is given below :

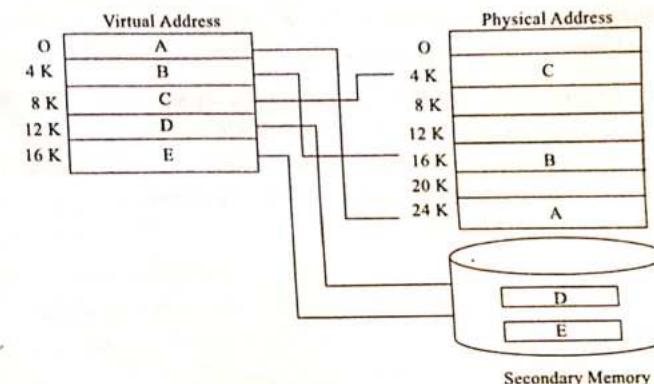


FIG 5.1 :

Virtual memory is commonly implemented by demand paging. It can also be implemented in a segmentation system. Demand segmentation can also be used to provide virtual memory.

5.2 DEMAND PAGING

A demand paging system is quite similar to a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance.

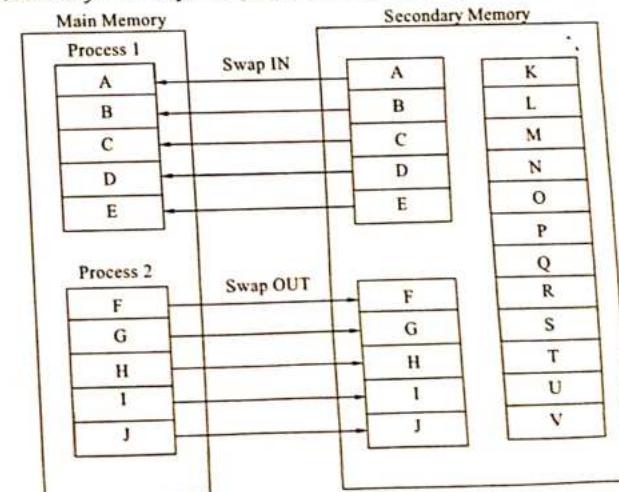


FIG 5.2 :

When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory. Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.

While executing a program, if the program references a page which is not available in the main memory because it was swapped out a little ago, the processor treats this invalid memory reference as a **page fault** and transfers control from the program to the operating system to demand the page back into the memory.

Advantages :

- Large virtual memory.
- More efficient use of memory.
- There is no limit on degree of multiprogramming.

Disadvantages :

- Number of tables and the amount of processor overhead for handling page interrupts are greater than in the case of the simple paged management techniques.

5.3 PAGE REPLACEMENT

1. Page replacement algorithms are the techniques using which an Operating System decides which memory pages to swap out, write to disk when a page of memory needs to be allocated.
2. Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.
3. When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion.
4. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

5.4 PAGE REPLACEMENT ALGORITHM

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself.

There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

Reference String : The string of memory references is called reference string. Reference strings are generated artificially or by tracing a given system and recording the address of each memory reference. The latter choice produces a large number of data, where we note two things.

- For a given page size, we need to consider only the page number, not the entire address.
- If we have a reference to a page p , then any immediately following references to page p will never cause a page fault. Page p will be in memory after the first reference; the immediately following references will not fault.
- For example, consider the following sequence of addresses "123, 215, 600, 1234, 76, 96".
- If page size is 100, then the reference string is 1,2,6,12,0,0

5.4.1 FIRST IN FIRST OUT (FIFO) ALGORITHM

- Oldest page in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages from the tail and add new pages at the head.

Reference String : 0, 2, 1, 6, 4, 0, 1, 0, 3, 1, 2, 1

Misses : x x x x x x x x x

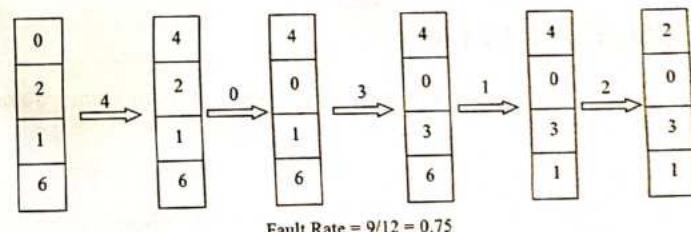


FIG 5.3 :

5.4.2 OPTIMAL PAGE ALGORITHM

- An optimal page-replacement algorithm has the lowest page-fault rate of all algorithms. An optimal page-replacement algorithm exists, and has been called OPT or MIN.
- Replace the page that will not be used for the longest period of time. Use the time when a page is to be used.

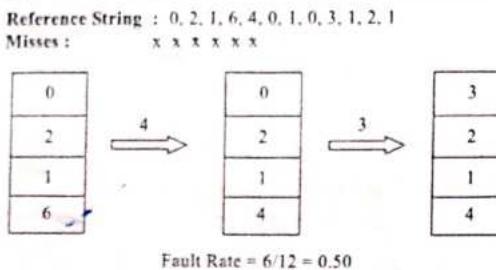


FIG 5.4 :

5.4.3 LEAST RECENTLY USED (LRU) ALGORITHM

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.

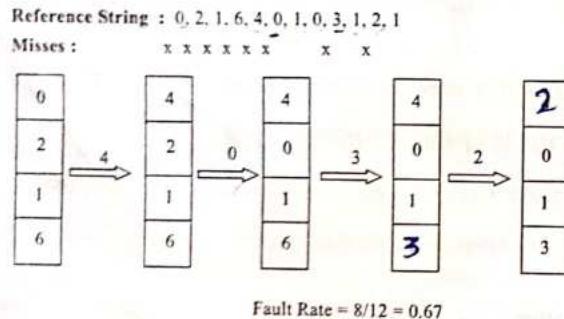


FIG 5.5 :

5.5 CONCEPT OF THRASHING

- At any given time, only few pages of any process are in main memory and therefore more processes can be maintained in memory.

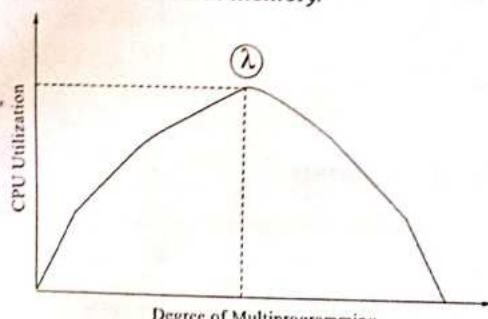


FIG 5.6 :

- Furthermore time is saved because unused pages are not swapped in and out of memory. However, the OS must be clever about how it manages this scheme.
- In the steady state practically, all of main memory will be occupied with process's pages, so that the processor and OS has direct access to as many processes as possible. Thus when the OS brings one page in, it must throw another out.
- If it throws out a page just before it is used, then it will just have to get that page again almost immediately. Too much of this leads to a condition called **Thrashing**.
- The system spends most of its time swapping pages rather than executing instructions. So a good page replacement algorithm is required.

In the given diagram, initial degree of multi programming upto some extent of point(λ), the CPU utilization is very high and the system resources are utilized 100%. But if we further increase the degree of multi programming the CPU utilization will drastically fall down and the system will spent more time only in the page replacement and the time taken to complete the execution of the process will increase. This situation in the system is called as thrashing.

5.5.1 CAUSES OF THRASHING

- High Degree of Multiprogramming** : If the number of processes keeps on increasing in the memory than number of frames allocated to each process will be decreased. So, less number of frames will be available to each process. Due to this, page fault will occur more frequently and more CPU time will be wasted in just swapping in and out of pages and the utilization will keep on decreasing.

For Example : Let free frames = 400

Case-1 : Number of process = 100

Then, each process will get 4 frames.

Case-2 : Number of process = 400

Each process will get 1 frame.

Case-2 is a condition of thrashing, as the number of processes are increased, frames per process are decreased. Hence CPU time will be consumed in just swapping pages.

- Lacks of Frames** : If a process has less number of frames then less pages of that process will be able to reside in memory and hence more frequent swapping in and out will be required. This may lead to thrashing. Hence sufficient amount of frames must be allocated to each process in order to prevent thrashing.

5.5.2 RECOVERY OF THRASHING

- Do not allow the system to go into thrashing by instructing the long term scheduler not to bring the processes into memory after the threshold.
- If the system is already in thrashing then instruct the mid term scheduler to suspend some of the processes so that we can recover the system from thrashing.

5.6 TECHNIQUES TO HANDLE

1. Locality Model : A locality is a set of pages that are actively used together.

The locality model states that as a process executes, it moves from one locality to another. A program is generally composed of several different localities which may overlap.

For example, when a function is called, it defines a new locality where memory references are made to the instructions of the function call, its local and global variables, etc. Similarly, when the function is exited, the process leaves this locality.

2. Working Set Model : This model is based on the above-stated concept of the Locality Model.

The basic principle states that if we allocate enough frames to a process to accommodate its current locality, it will only fault whenever it moves to some new locality. But if the allocated frames are lesser than the size of the current locality, the process is bound to thrash.

According to this model, based on parameter A, the working set is defined as the set of pages in the most recent 'A' page references. Hence, all the actively used pages would always end up being a part of the working set.

The accuracy of the working set is dependent on the value of parameter A. If A is too large, then working sets may overlap. On the other hand, for smaller values of A, the locality might not be covered entirely.

If D is the total demand for frames and is the working set size for process i, now, if 'm' is the number of frames available in the memory, there are 2 possibilities:

- (i) $D > m$ i.e., total demand exceeds the number of frames, then thrashing will occur as some processes would not get enough frames.
- (ii) $D \leq m$, then there would be no thrashing.

3. Page Fault Frequency : A more direct approach to handling thrashing is the one that uses the Page-Fault Frequency concept.

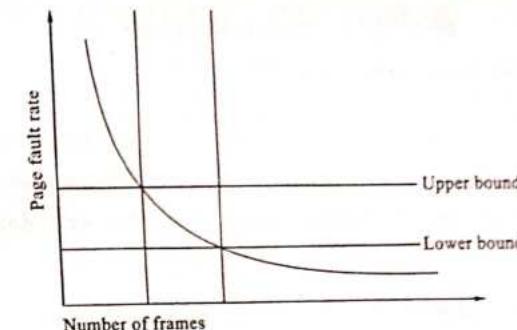


FIG 5.7 :

The problem associated with Thrashing is the high page fault rate and thus, the concept here is to control the page fault rate.

If the page fault rate is too high, it indicates that the process has too few frames allocated to it. On the contrary, a low page fault rate indicates that the process has too many frames.

Upper and lower limits can be established on the desired page fault rate as shown in the diagram.

If the page fault rate falls below the lower limit, frames can be removed from the process. Similarly, if the page fault rate exceeds the upper limit, more frames can be allocated to the process.

In other words, the graphical state of the system should be kept limited to the rectangular region formed in the given diagram.

Here too, if the page fault rate is high with no free frames, then some of the processes can be suspended and frames allocated to them can be reallocated to other processes. The suspended processes can then be restarted later.

OBJECTIVE TYPE QUESTION

1. What a virtual-memory miss is called?
 - (a) Hit miss
 - (b) Page hit
 - (c) Page miss
 - (d) Page fault

2. A process is copied into the main memory from the secondary memory
 - (a) Swapping
 - (b) Paging
 - (c) Segmentation
 - (d) Demand paging

3. Process is trash
 - (a) It spends more time paging than executing
 - (b) Ut spends less time paging than executing
 - (c) Page fault occurs
 - (d) Swapping can not take place

4. In which Swap space exists
 - (a) CPU
 - (b) Primary memory
 - (c) secondary memory
 - (d) None of the above

5. A page fault occurs when?
 - (a) A page gives inconsistent data
 - (b) A page cannot be accessed due to its absence from memory
 - (c) A page is invisible
 - (d) All of the above

6. When a page fault occurs, the state of the interrupted process is _____
 - (a) Disrupted
 - (b) Invalid
 - (c) Saved
 - (d) None of the above

7. When a process begins execution with no pages in memory?
 - (a) Process execution becomes impossible
 - (b) A page fault occurs for every page brought into memory
 - (c) Process causes system crash
 - (d) None of the above

8. A process is said to be in a ready state if
 - (a) A process is newly created and is waiting for the CPU.
 - (b) A process is unable to run and wait until some task has been completed
 - (c) A process is using the CPU.
 - (d) None of the above.

9. How many ways can a process be terminated?
 - (a) Normal exit
 - (b) Fatal error
 - (c) Killed by another process
 - (d) All of the above.

10. Virtual address generated by the CPU while a program is running is called as?
 - (a) Physical address
 - (b) Logical address
 - (c) MAC address
 - (d) None of the above

11. A program is generally composed of several different localities, which _____ overlap.
 - (a) May
 - (b) Must
 - (c) Do not
 - (d) Must not

12. The accuracy of the working set depends on the selection of _____
 - (a) Working set model
 - (b) Working set size
 - (c) Memory size
 - (d) Number of pages in memory

13. If working set window is too small _____
 - (a) It will not encompass entire locality
 - (b) It may overlap several localities
 - (c) It will cause memory problems
 - (d) None of the above

14. Thrashing _____ the CPU utilization.
 - (a) Increases
 - (b) Keeps constant
 - (c) Decreases
 - (d) None of the above

ANSWERS

1. (a)	2. (b)	3. (c)	4. (a)	5. (b)	6. (a)	7. (a)	8. (c)	9. (c)	10. (b)
11. (a)	12. (a)	13. (a)	14. (a)	15. (d)					

REVIEW QUESTIONS**Part-A**

1. What is thrashing?
2. Define Virtual memory
3. What is demand paging?
4. List page replacement algorithms.
5. What is page fault frequency?

Part-B

1. Explain about paging concept. (Apr. 2008)
2. Describe the demand paging. (Apr/May. 2011)
3. Describe FIFO as a page replacement algorithm. (Mar/Apr. 2014 ; Apr/May. 2011)
4. Write about FOFO and LRU page replacement algorithms. With examples. (Apr/May. 2012)
5. Explain about
 - (i) Working set model
 - (ii) Page fault frequency
6. Explain briefly about page replacement. (Apr. 2007)
7. Explain the advantages of demand paging. (Mar/Apr. 2004)
8. Write about optimal page replacement algorithm.
9. Explain briefly about the concept of thrashing.
10. Difference between FIFO & LRU.

CHAPTER**6****DISK SCHEDULING AND FILE MANAGEMENT***Chapter Outline*

- 6.0 INTRODUCTION
- 6.1 VARIOUS DISK PERFORMANCE PARAMETERS
- 6.2 DISK SCHEDULING POLICIES
- 6.3 FILE MANAGEMENT
- 6.4 VARIOUS ACCESS METHODS
- 6.6 DIRECTORY STRUCTURE ORGANIZATION

6.0 INTRODUCTION

The Disk Scheduling Algorithm in OS is used to manage input and output requests to the disk. Disk Scheduling is important as multiple requests are coming to disk simultaneously and it is also known as the "**Request Scheduling Algorithm**".

A file management system is used for file maintenance (or management) operations. It is a type of software that manages data files in a computer system.

A file management system has limited capabilities and is designed to manage individual or group files, such as special office documents and records.

6.1 VARIOUS DISK PERFORMANCE PARAMETERS

- Disk Partitioning :** Management of disk partitions affects the overall performance of the disk. Creating too many partitions may cause a decrease in overall disk performance in instances where data is constantly being accessed across the various partitions. Disk partitioning increases overall performance when data types are segregated in different partitions. For example, OS, data, and games reside in respective partitions. The magnetic head moves within a partition while loading a particular program and doesn't traverse across different partitions.
- Drive Form Factor :** This is one parameter that differentiates one hard disk from the other. It refers to the actual size of the drive. **Drive form factor** specifies the disk's total storage capacity and the corresponding number of platters used within it and determines the disk's transfer rate. **Transfer rate** refers to the speed at which data is written to or read from the disk.
- Spindle Speed :** The speed of the spindle on which the disks rotate determines the speed of the hard drive in RPMs (revolutions per minute). A high spindle speed means the drive's platters rotate at high speeds, which translates into faster data access and retrieval times.
- Platter Diameter :** Even though spindle speed is an important factor in data transfer rates, it isn't the only factor. The diameter of the platter also affects the drive's data transfer rates. This is because the speed of the spindle does not change but the outer perimeters of the discs covers a greater distance than the inner perimeters. This means the diameter will affect the rate at which the magnetic head accesses different areas of the platter. Platters with a large diameter have higher transfer rates at the same spindle speed than platters with a small diameter.
- Access Times :** Data may be stored in any area on a given platter. For any read or write request, the magnetic head has to re-position itself to the appropriate platter and its location for the given task. Access times consist of various parameters.

- (i) **Write Seek Time :** Time taken by the magnetic head to locate available space on a platter to perform a write operation.
- (ii) **Read Seek Time :** Time taken by the magnetic head to locate data to be read.
- (iii) **Full Stroke Time :** Time taken by the magnetic head to move from one platter to the next.
- (iv) **Track to Track :** Time taken by the magnetic head to move from one sequential track to the next on a given platter
- 6. **Buffer Memory :** Buffer memory is a dedicated section of memory reserved for quicker access when data needs to be moved from one area of drive to the other. Today, this reserved space can have a capacity of 16MB. The larger the size of the buffer, the faster the data access.

6.2 DISK SCHEDULING POLICIES

- As we know, a process needs two type of time, CPU time and IO time. For I/O, it requests the Operating system to access the disk.
- However, the operating system must be fare enough to satisfy each request and at the same time, operating system must maintain the efficiency and speed of process execution.
- The technique that operating system uses to determine the request which is to be satisfied next is called disk scheduling.

Let's discuss some important terms related to disk scheduling.

- **Seek Time :** Seek time is the time taken in locating the disk arm to a specified track where the read/write request will be satisfied.
 - **Rotational Latency :** It is the time taken by the desired sector to rotate itself to the position from where it can access the R/W heads.
 - **Transfer Time :** It is the time taken to transfer the data.
 - **Disk Access Time :** Disk access time is given as,
- $$\text{Disk Access Time} = \text{Rotational Latency} + \text{Seek Time} + \text{Transfer Time}$$
- **Disk Response Time :** It is the average of time spent by each request waiting for the IO operation.
 - **Purpose Of Disk Scheduling :** The main purpose of disk scheduling algorithm is to select a disk request from the queue of IO requests and decide the schedule when this request will be processed.

DISK SCHEDULING ALGORITHMS**6.2.1 FIFO (FCFS)**

It is the simplest Disk Scheduling algorithm. It services the IO requests in the order in which they arrive. There is no starvation in this algorithm, every request is serviced.

Disadvantages :

- The scheme does not optimize the seek time.
- The request may come from different processes therefore there is the possibility of inappropriate movement of the head.

Example : Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 50, 89, 52, 61, 87, 25

Head pointer starting at 50 and moving in left direction. Find the number of head movements in cylinders using FCFS scheduling.

Solution :

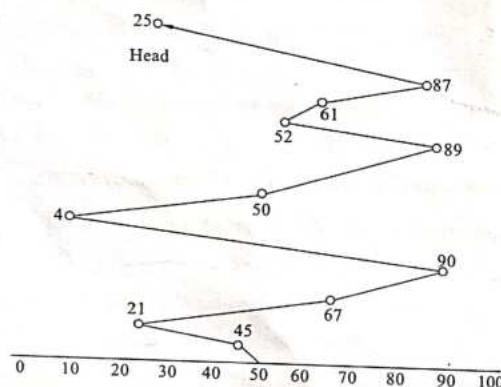


FIG 6.1 :

Number of cylinders moved by the head

$$\begin{aligned}
 &= (50 - 45) + (45 - 21) + (67 - 21) + (90 - 67) + (90 - 4) + (50 - 4) \\
 &\quad + (89 - 50) + (61 - 52) + (87 - 61) + (87 - 25) \\
 &= 5 + 24 + 46 + 23 + 86 + 46 + 49 + 9 + 26 + 62 \\
 &= 376
 \end{aligned}$$

6.2.2 SSTF SCHEDULING ALGORITHM

Shortest seek time first (SSTF) algorithm selects the disk I/O request which requires the least disk arm movement from its current position regardless of the direction. It reduces the total seek time as compared to FCFS.

It allows the head to move to the closest track in the service queue.

Disadvantages :

- It may cause starvation for some requests.
- Switching direction on the frequent basis slows the working of algorithm.
- It is not the most optimal algorithm.

Example : Consider the following disk request sequence for a disk with 100 tracks 45, 21, 67, 90, 4, 89, 52, 61, 87, 25

45, 21, 67, 90, 4, 89, 52, 61, 87, 25

Head pointer starting at 50. Find the number of head movements in cylinders using SSTF scheduling.

Solution :

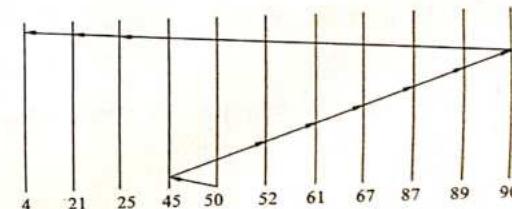


FIG 6.2 :

$$\text{Number of cylinders} = 5 + 7 + 9 + 6 + 20 + 2 + 1 + 65 + 4 + 17 = 136$$

6.2.3 SCAN ALGORITHM

It is also called as Elevator Algorithm. In this algorithm, the disk arm moves into a particular direction till the end, satisfying all the requests coming in its path, and then it turns back and moves in the reverse direction satisfying requests coming in its path.

It works in the way an elevator works, elevator moves in a direction completely till the last floor of that direction and then turns back.

Example : Consider the following disk request sequence for a disk with 100 tracks

98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using SCAN scheduling.

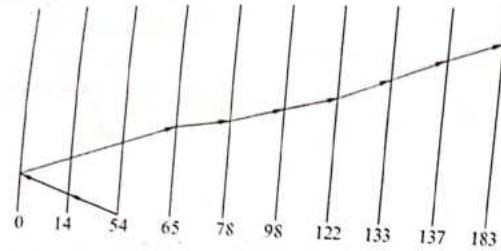


FIG 6.3 :

$$\text{Number of Cylinders} = 40 + 14 + 65 + 13 + 20 + 24 + 11 + 4 + 46 = 237$$

6.2.4 C-SCAN ALGORITHM

In C-SCAN algorithm, the arm of the disk moves in a particular direction servicing requests until it reaches the last cylinder, then it jumps to the last cylinder of the opposite direction without servicing any request then it turns back and start moving in that direction servicing the remaining requests.

Example : Consider the following disk request sequence for a disk with 100 tracks
98, 137, 122, 183, 14, 133, 65, 78

Head pointer starting at 54 and moving in left direction. Find the number of head movements in cylinders using C-SCAN scheduling.

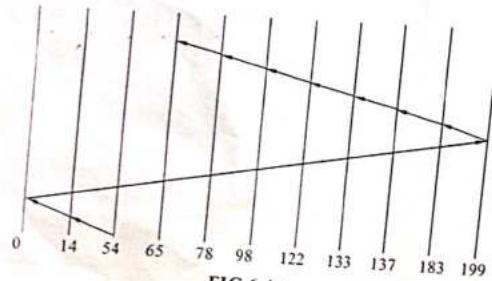


FIG 6.4 :

$$\text{No. of cylinders crossed} = 40 + 14 + 199 + 16 + 46 + 4 + 11 + 24 + 20 + 13 = 387$$

6.3 FILE MANAGEMENT

A file management system is used for file maintenance (or management) operations. It is a type of software that manages data files in a computer system.

WARNING

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

A file management system has limited capabilities and is designed to manage individual or group files, such as special office documents and records. It may display report details, like owner, creation date, state of completion and similar features useful in an office environment.

A file management system is also known as a *file manager*.

6.4 VARIOUS ACCESS METHODS

There are several files that will be stored in computer memory. When an application needs a file, the operating system should search in the computer memory and has to access the required files. There are several ways the operating system can access the information in the files called *File Access Methods*. The file access methods provide operations like reading from or writing to a file stored in the computer memory.

The various file access methods are :

- Sequential Access Method.
- Direct Access Method.
- Indexed Access Method.
- Indexed Sequential Access Method.

Sequential Access Method : Among all the access methods, it is considered the simplest method. As the name itself suggests it is the sequential (or series) processing of information present in a file. Due to its simplicity most of the compilers, editors, etc., use this method.

Processing is carried out with the use of two operations namely, read and write. Read operation is responsible for reading the next portion of the file and after a successful read of the record, the pointer proceeds automatically to the next record which tracks the I/O location. Write operation is responsible for writing at the end of the file and shifts the pointer towards the end of the newly added record.

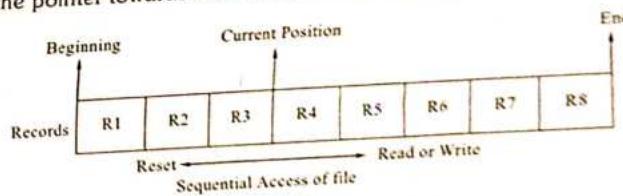


FIG 6.5 :

In this type of access, while processing the records sequentially, some of the records can be skipped in both directions (either forward or backward) and can also be reset or rewound to the head of the file (beginning).

WARNING

XEROX / PHOTOCOPYING OF THIS BOOK IS ILLEGAL

Advantages of Sequential Access Method :

- This method of file access is easy to implement.
- It provides fast access to the next record using lexicographic order.

Disadvantages of Sequential Access Method :

- This type of file access method is slow if the file record to be accessed next is not present next to the current record.
- Inserting a new record may require moving a large proportion of the file.

Direct Access Method : This access method is also called real-time access where the records can be read irrespective of their sequence. This means they can be accessed as a file that is accessed from a disk where each record carries a sequence number. For example, block 40 can be accessed first followed by block 10 and then block 30, and so on. This eliminates the need for sequential read or write operations.

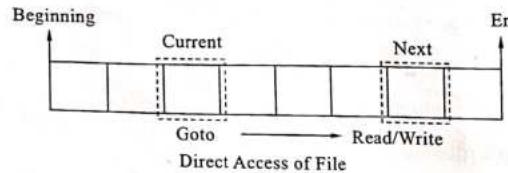


FIG 6.6 :

A major example of this type of file access is the database where the corresponding block of the query is accessed directly for instant response. This type of access saves a lot of time when a large amount of data is present. In such cases, hash functions and index methods are used to search for a particular block.

The read next and write next operations of sequential access are modified to read n and write n, where 'n' is the block number. A more promising approach is to use direct access to map the position of the file and sequential access for performing read next operation in that block.

This can be done by using relative block numbers which are nothing but index related to the access of blocks. For example, relative block numbers 0, 1, 2, ... can be allocated to block numbers 72, 1423, 20, etc. This approach is adopted by some of the operating systems while others use either sequential or direct access.

Advantages of Direct Access Method :

- The files can be immediately accessed decreasing the average access time.
- In the direct access method, in order to access a block, there is no need of traversing all the blocks present before it.

Indexed Access Method : This method is typically an advancement in the direct access method which is the consideration of index. A particular record is accessed by browsing through the index and the file is accessed directly with the use of pointers or addresses present in the index as shown below.

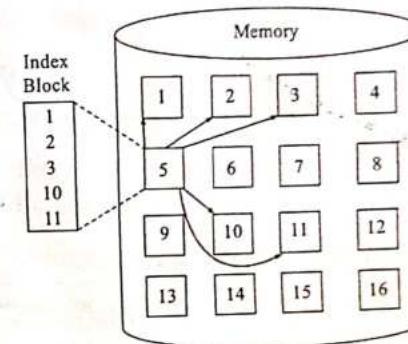


FIG 6.7 : Index Access of File

To understand the concept consider a book store where the database contains a 12-digit ISBN and a four-digit product price. If the disk can carry 2048 (2kb) of bytes per block then 128 records of 16 bytes (12 for ISBN and 4 for price) can be stored in a single block. This results in a file carrying 128000 records to be reduced to 1000 blocks to be considered in the index each entry carrying 10 digits. To find the price of a book binary search can be performed over the index with which the block carrying that book can be identified.

A drawback of this method is, it is considered ineffective in the case of a larger database with very large files which results in making the index too large.

Indexed Sequential Access Method : To overcome the drawback associated with indexed access, this method is used where an index of an index is created. Primary index points to the secondary index and the secondary index points to the actual data items as shown below.

An example of such a method is ISAM (Indexed-Sequential Access Method) of IBM which carries two types of indexes. They are a master index and secondary index.

Master index carries pointers to the secondary index whereas, the secondary carries blocks that point directly to the disk blocks. Two binary searches are performed to access a data item. The first one is performed on a master index and the second one on the secondary index. This type of method is a two direct access read method.

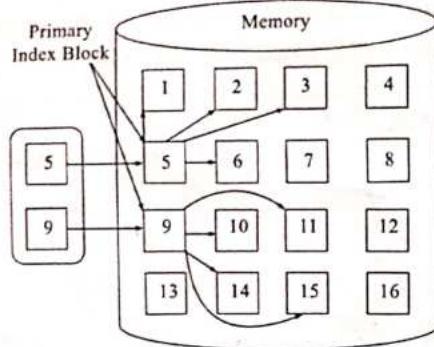


FIG 6.8 : Indexed Sequential Access of File

6.6 DIRECTORY STRUCTURE ORGANIZATION

What is a Directory : A Directory is the collection of the correlated files on the disk. In simple words, a directory is like a container which contains file and folder. In a directory, we can store the complete file attributes or some attributes of the file. A directory can be comprised of various files. With the help of the directory, we can maintain the information related to the files.

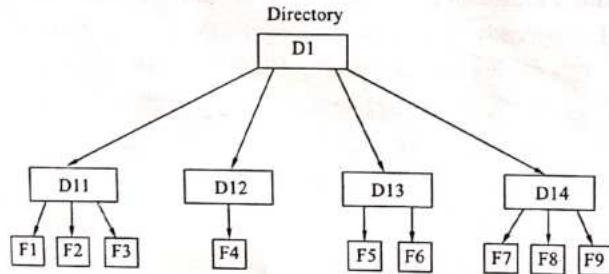


FIG 6.9 :

To take the advantages of various file systems on the different operating systems, we can divide the hard disk into multiple partitions, which are of different sizes. Partitions are known as minidisks or volumes.

There should be at least one directory that must be present in each partition. Through it, we can list all the files of the partition. In the directory for each file, there is a directory entry, which is maintained, and in that directory entry, all the information related to the file is stored.

There are various types of information which are stored in a directory :

1. Name
2. Type
3. Location
4. Size

5. Position
6. Protection
7. Usage
8. Mounting
1. **Name :** Name is the name of the directory, which is visible to the user.
2. **Type :** Type of a directory means what type of directory is present such as single-level directory, two-level directory, tree-structured directory, and Acyclic graph directory.
3. **Location :** Location is the location of the device where the header of a file is located.
4. **Size :** Size means number of words/blocks/bytes in the file.
5. **Position :** Position means the position of the next-read pointer and the next-write pointer.
6. **Protection :** Protection means access control on the read/write/delete/execute.
7. **Usage :** Usage means the time of creation, modification, and access, etc.
8. **Mounting :** Mounting means if the root of a file system is grafted into the existing tree of other file systems.

6.6.1 OPERATIONS ON DIRECTORY

The various types of operations on the directory are :

- | | |
|--------------|---------------------|
| 1. Creating | 2. Deleting |
| 3. Searching | 4. List a directory |
| 5. Renaming | 6. Link |
| 7. Unlink | |
1. **Creating :** In this operation, a directory is created. The name of the directory should be unique.
 2. **Deleting :** If there is a file that we don't need, then we can delete that file from the directory. We can also remove the whole directory if the directory is not required. An empty directory can also be deleted. An empty directory is a directory that only consists of dot and dot-dot.
 3. **Searching :** Searching operation means, for a specific file or another directory, we can search a directory.
 4. **List a directory :** In this operation, we can retrieve all the files list in the directory. And we can also retrieve the content of the directory entry for every file present in the list. If in the directory, we want to read the list of all files, then first, it should be opened, and afterwards we read the directory, it is a must to close the directory so that the internal table space can be free up.

6.. TYPES OF DIRECTORY STRUCTURE

There are various types of directory structure :

1. Single-Level Directory.
2. Two-Level Directory.
3. Tree-Structured Directory.
4. Acyclic Graph Directory.
5. General-Graph Directory.

1. Single-level Directory : Single-Level Directory is the easiest directory structure. There is only one directory in a single-level directory, and that directory is called a root directory. In a single-level directory, all the files are present in one directory that makes it easy to understand. In this, under the root directory, the user cannot create the subdirectories.

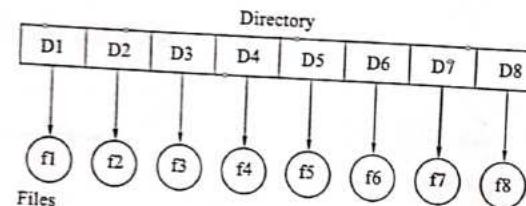


FIG 6.10 :

Advantages of Single-level Directory :

1. The implementation of a single-level directory is so easy.
2. In a single-level directory, if all the files have a small size, then due to this, the searching of the files will be easy.
3. In a single-level directory, the operations such as searching, creation, deletion, and updating can be performed.

Disadvantages of Single-level Directory :

1. If the size of the directory is large in Single-Level Directory, then the searching will be tough.
2. In a single-level directory, we cannot group the similar type of files.
3. Another disadvantage of a single-level directory is that there is a possibility of collision because the two files cannot have the same name.
4. The task of choosing the unique file name is a little bit complex.

2. Two-level Directory : Two-Level Directory is another type of directory structure. In this, it is possible to create an individual directory for each of the users. There is one master node in the two-level directory that includes an individual directory for every user. At the second level of the directory, there is a different directory present for each of the users. Without permission, no user can enter into the other user's directory.

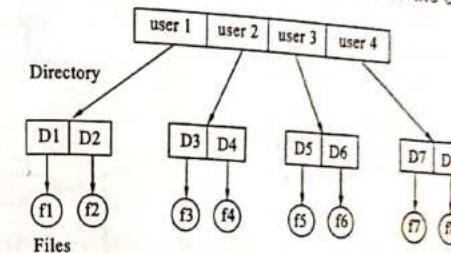


FIG 6.11 :

Characteristics of Two-level Directory :

1. In a two-level directory, there may be same file names of different users.
2. There is a pathname of each file such as /User-name/directory-name/
3. In a two-level directory, we cannot group the files which are having the same name into a single directory for a specific user.
4. In a two-level directory, searching is more effective because there is only one user's list, which is required to be traversed.

Advantages of Two-level Directory :

1. In the two-level directory, various users have the same file name and also directory name.
2. Because of using the user-grouping and pathname, searching of files are quite easy.

Disadvantages of Two-level Directory :

1. In a two-level directory, one user cannot share the file with another user.
2. Another disadvantage with the two-level directory is it is not scalable.
3. **Tree-structured Directory :** A Tree-structured directory is another type of directory structure in which the directory entry may be a sub-directory or a file. The tree structure reduces the limitations of the two-level directory. We can group the same type of files into one directory.

In a tree-structured directory, there is an own directory of each user, and any user is not allowed to enter into the directory of another user. Although the user can read the data of root, the user cannot modify or write it. The system administrator only has full access to the root directory. In this, searching is quite effective and we use the current working concept. We can access the file by using two kinds of paths, either absolute or relative.

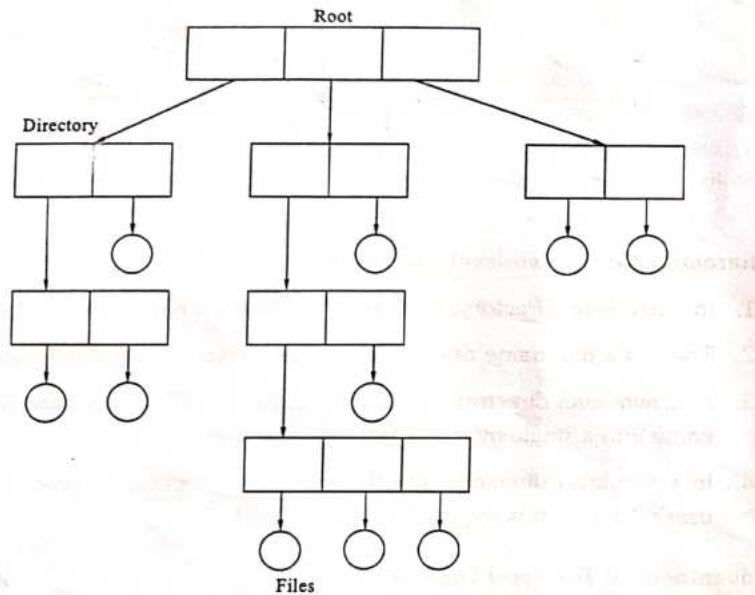


FIG 6.12 :

Advantages of Tree-structured Directory :

- The tree-structured directory is very scalable.
- In the tree-structures directory, the chances of collision are less.
- In the tree-structure directory, the searching is quite easy because, in this, we can use both types of paths, which are the absolute path and relative path.

Disadvantages of Tree-structure Directory :

- In the tree-structure directory, the files cannot be shared.
- Tree-structure directory is not efficient because, in this, if we want to access a file, then it may go under multiple directories.

- Another disadvantage of the tree-structure directory is that each file does not fit into the hierachal model. We have to save the files into various directories.
- 4. Acyclic-graph Directory :** In the tree-structure directory, the same files cannot exist in the multiple directories, so sharing the files is the main problem in the tree-structure files. In the acyclic-graph directory, we can provide the sharing of or subdirectory. We can share those files among the two directory entries.

With the help of aliases, and links, we can create this type of directory graph. We may also have a different path for the same file. Links may be of two kinds, which are hard link (physical) and symbolic (logical).

If we delete the files in acyclic graph structures, then

1. In the hard link (physical) case, we can remove the actual files only if all the references to the file are deleted.
2. In the symbolic link (logical) case, we just delete the file, and there is only a dangling point that is left.

Advantages of Acyclic-graph Directory :

- In the acyclic-graph directory, the sharing of files is possible.
- In the acyclic-graph directory, because of different-different paths, searching is easy.

Disadvantages of Acyclic-graph Directory :

- If the files are shared through linking, there may be a problem in the case of deleting.
- If we are using soft link, then in this case, if the file is deleted then there is only a dangling pointer which is left.
- If we are using hardlink, in this case, when we delete a file, then we also have to remove all the reference connected with it.

- 5. General-graph Directory :** The General-Graph directory is another vital type of directory structure. In this type of directory, within a directory we can create cycle in the directory where we can derive the various directory with the help of more than one parent directory.

The main issue in the general-graph directory is to calculate the total space or size, taken by the directories and the files.

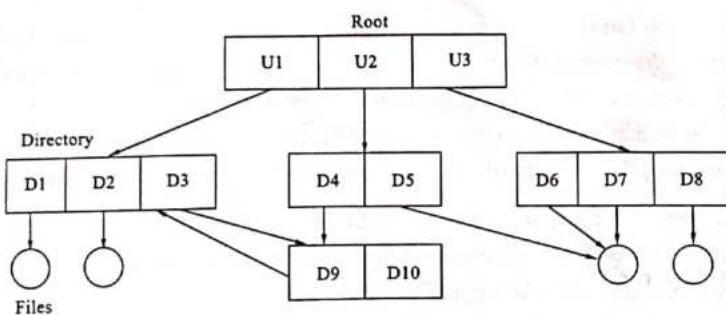


FIG 6.13 :

Advantages of General-graph Directory :

1. The General-Graph directory is more flexible than the other directory structure.
2. Cycles are allowed in the general-graph directory.

Disadvantages of General-graph Directory :

1. In general-graph directory, garbage collection is required.
2. General-graph directory is more costly, among other directory structures.

1 12
3 14
5 16
7
8

OBJECTIVE TYPE QUESTION

1. In _____ information is recorded magnetically on platters.

(a) magnetic disks	(b) electrical disks
(c) assemblies	(d) cylinders
2. The heads of the magnetic disk are attached to a _____ that moves all the heads as a unit.

(a) spindle	(b) disk arm
(c) track	(d) none of the mentioned
3. The set of tracks that are at one arm position make up a _____

(a) magnetic disks	(b) electrical disks
(c) assemblies	(d) cylinders
4. The time taken to move the disk arm to the desired cylinder is called the _____

(a) positioning time	(b) random access time
(c) seek time	(d) rotational latency
5. The time taken for the desired sector to rotate to the disk head is called _____

(a) positioning time	(b) random access time
(c) seek time	(d) rotational latency
6. The users are allowed to read information in directories.

(a) True	(b) False
----------	-----------
7. A filename without path information.

(a) File name	(b) Complete filename
(c) Directory name	(d) Relative filename
8. Files in which users store information?

(a) Info files	(b) Ordinary files
(c) Special files	(d) Complex files
9. Data cannot be written to secondary storage unless written within a _____

(a) file	(b) swap space
(c) directory	(d) text format
10. File attributes consist of _____

(a) name	(b) type
(c) identifier	(d) all of the mentioned