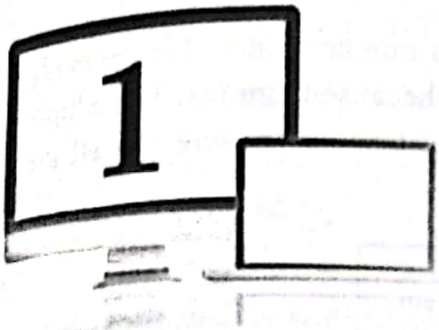


OVERVIEW OF OPERATING SYSTEM

Definition of Operating Systems, Types of Operating Systems, Operating System Services, User operating system interface, System Calls, Types of System Calls, System Programs, Operating System Structure, Virtual Machine, Benefits of Virtual Machine

1.1. Operating System

An operating system is a system software that is an interface between a computer user and computer hardware. An operating system is a software program which performs all the basic tasks like file management, memory management, process management, handling input and output and controlling peripheral devices such as disk drive and printers.

An operating system is similar to a government. Like a government, it performs no useful function by itself. It simply provides an environment within which other program can do useful work.

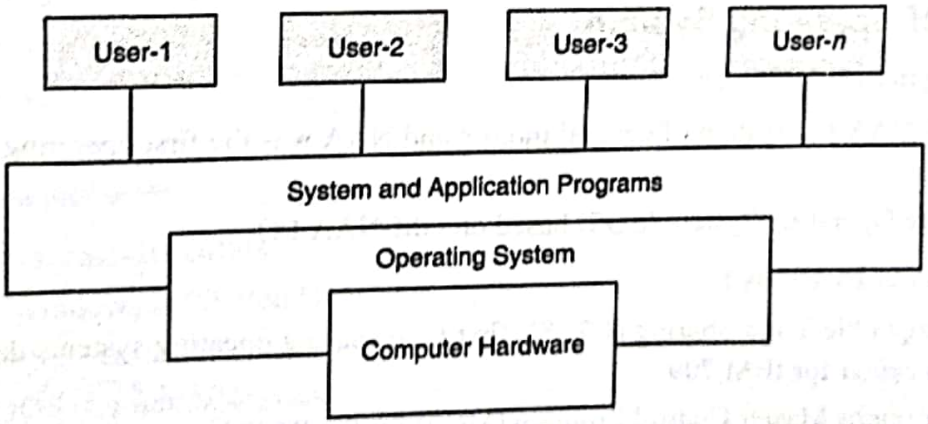


Fig. 1

Operating system have two viewpoints that of the user and that of the system.

1. User view (Resource utilization) : Many computer users sit in front of a *PC*, consisting of a monitor, keyboard, mouse and system unit. Such a system is designed for one user to monopolize its resources. The different types of user view experiences can be explained as follows :

(a) If the user is using personal computer, there is no need for the opeating system to worry about resource utilization. This is because the personal computer uses all the resoure available and there is no sharing.

(b) If the user is using a system connected to a mainframe or a minicomputer. The operating system is largly concerned with resource utilization. This is because there may be multiple terminals connected to the mainframe and the operating system makes sure that all the resources such as CPU, memory, I/O devices, etc.

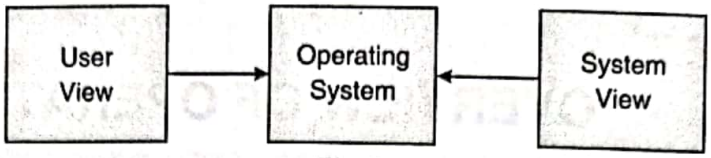


Fig. 2

(c) If the user is sitting on a workstation connected to other workstation through networks, then the operating system needs to focus on both individual usage of resources and sharing through the networks.

2. System view (Resource allocator : The different types of system view for operating system can be explained as follows :.

(a) The system views the operating system as a resource allocator. There are many resources such as CPU time, memory space, file storage space, I/O devices, etc. that are required by processes for execution. It is the duty of the operating system to allocate these resources.

(b) The operating system can also work as a control program. It manages all the processes and I/O devices so that the computer system works smoothly and there are no errors.

(c) Computer were required to easily solve user problems so operating systems were developed to easily communicate with the hardware.

1.2. History of Operating System

Operating Systems IN 1950s :

1956 : GM-NAA I/O system of general motors and NAA was the first operating system for the IBM704 computer.

1959 : Share Operating System (SOS) based on GM-NAA I/O

Operating systems In 1960s :

1961 : Compatible Time-Sharing (CTSS), first time sharing opeating systems developed at the MIT computation center for IBM 709

1961 : Burroughs Master Control Program (MCP) for the B5000 system.

1964 : IBM system 1360-batch processing system for IBM mainframe.

1968 : The multiprogramming system that supported multitasking.

1969 : Multics (Multiplexed Information and Computing Service) time-sharing operating system based on the concept of a single-level memory.

1969 : UNIX a family of multitasking, multiuser computer operating systems at the bell labs research center by ken thompson. Dennis ritchie and others.

Operating systems -In 1970s :

1970 : DOS-11 by digital Equipment corparation for Digital PDP-11 minicomputer

1972 : PRIMOS by Prime Computer for its minicomputer systems

1973 : Xerox Alto First Computer designed to support on OS based on GUI

1974 : Multi-Programming Executive, a mainframe computer real time operating system made by Hewlett Packard.

1977 : Berkeley Unix based on the source code of the original Unix developed at Bell Labs. Widely adopted by workstation. Descendants, such as free BSD, Open BSD. Net BSD or Dragon Fly BSD.

Operating system-In 1980s

1981 : MS-DOS on Operating System for X-86 based PC developed by Microsoft. Rebranding as IBM PC DOS

1984 : Classic Mac OS developed for the Macintosh Family of Personal Computers by Apple Inc. from 1984 to 2001 starting with system 1 and ending with MAC OS.9.

1985 : Windows 1.0 by Microsoft for Apple's January 1984. The first mass-product personal computer with a graphical user interface (GUI)

1987 : Minix (from mini-unix) is a POSIX-Compliant Unix like operating system based on a microkernel architecture.

Operating Systems-In 1990s :

1990 : Microsoft Windows 3.0

1991 : GNU/LINUX

1992 : Windows 3.1

1993 : Windows NT

1995 : Windows 95

1998 : Window 98

Operating Systems-In 2000s :

2000 : Windows 2000, Red Hat Linux

2001 : MAC OS X, Windows XP.

2007 : iOS (for iphones, ipads)

2008 : Android OS (Mobile phone came with the Android Operating System)

2009 : Windows 7

Operating Systems-In 2010s :

2012 : Windows 8, Haiku R1, Alpha 4

2013 : Debion

2015 : Windows 10, Z/OS

Classification of Operating System

Operating systems can be classified as follows :

1. Single User : Just allows one user to use the program at one time. Example-DOS (Disk operating system)

2. Multi User : In multi user operating system two or more users to use their programs at the same time. Examples UNIX, LINUX

3. Single Processor : A single processor operating system contains only one processor, so only one process can be executed at a time and then the process is selected from the ready queue. Example IBM mainframe (IBM System/360)

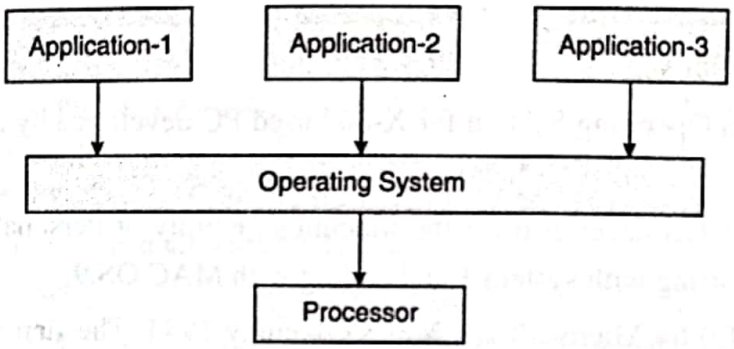


Fig. 3

4. Multiprocessor : Multiprocessor Operating System allows the multiple processors and these processors are connected with physical memory, computer buses and peripheral devices. Main objective of using multiprocessor operating system is to consume high computing power and increase the execution speed of system. Examples Windows NT, 2000, XP and UNIX.

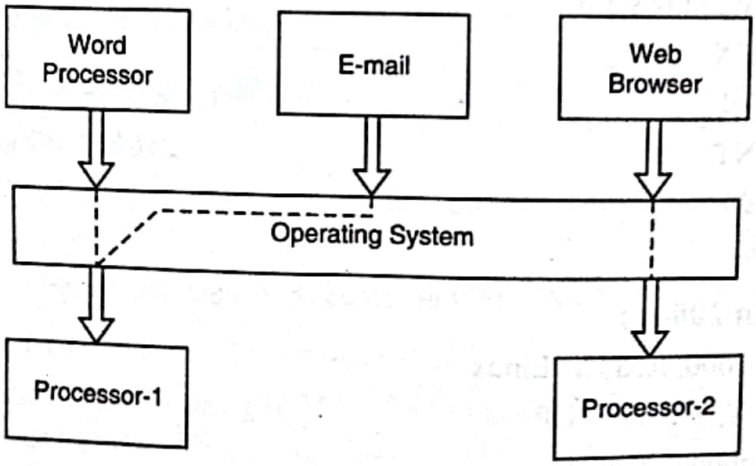


Fig. 4

The multi processor systems are of two types.

(a) Asymmetric Multiprocessing : In asymmetric multiprocessing operating system each processor is assigned a specific task. A master processor controls the system and other processors either look to the master for instruction or have predefined tasks.

(b) Symmetric Multiprocessing : In symmetric multiprocessing operating system each processor performs all tasks.

5. Clustered System : Another type of multiple-CPU system is the clustered system. A cluster is created when two or more computer systems are merged. In cluster system, computer share common storage and the system work together.

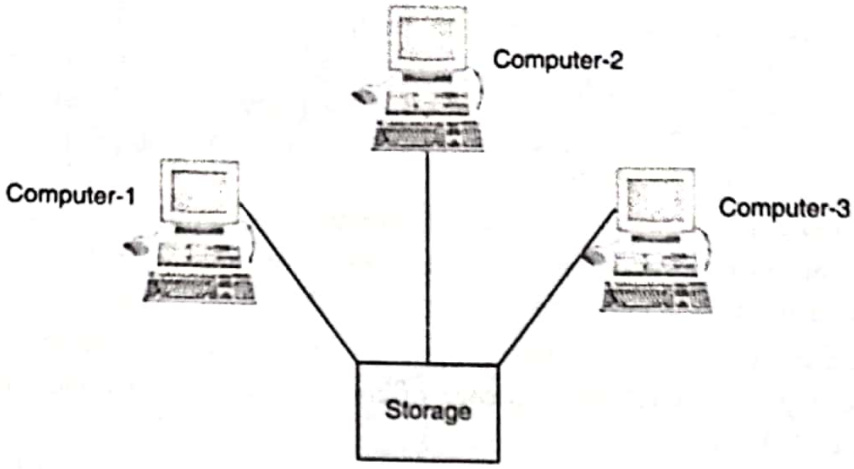


Fig. 5

There are two types of cluster systems.

(a) Symmetric cluster : In this type of clustering all the nodes run application and monitor other nodes at the same time.

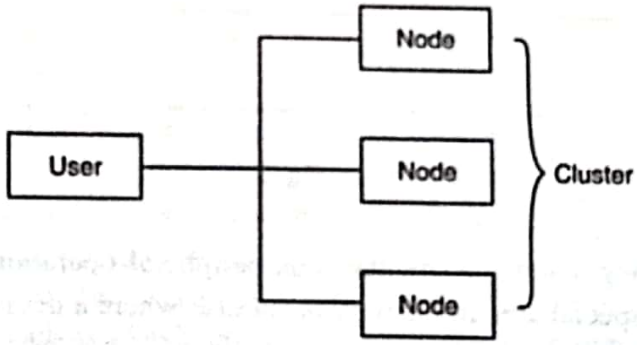


Fig. 6

(b) Asymmetric cluster : In this type of clustering, one of the nodes is in a hot stand by mode, while rest all nodes run different applications.

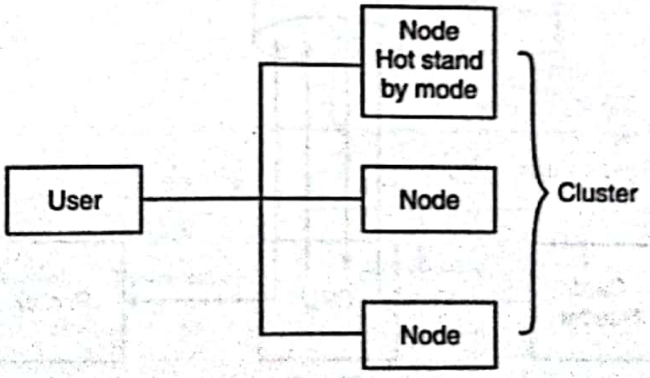


Fig. 7

Types of operating systems

Some of the widely used operating systems are as follows :

1. Batch operating system.
2. Multiprogramming operating system.
3. Multitasking/time sharing operating system.
4. Real time operating system.
5. Distributed operating system.

1. Batch operating system : This types of operating system does not interact with the computer directly. There is an operator which takes similar job, have same requirement and group them into batches. It is responsibility of operator to sort the jobs with similar needs.

Batch operating system is one where programs and data are collected together in a batch before processing starts. A job is predefined sequence of commands, programs and data that are combined into a single unit called job.

When a job completes execution, its memory is released and the output for the job gets copied into an output spool for later printing.

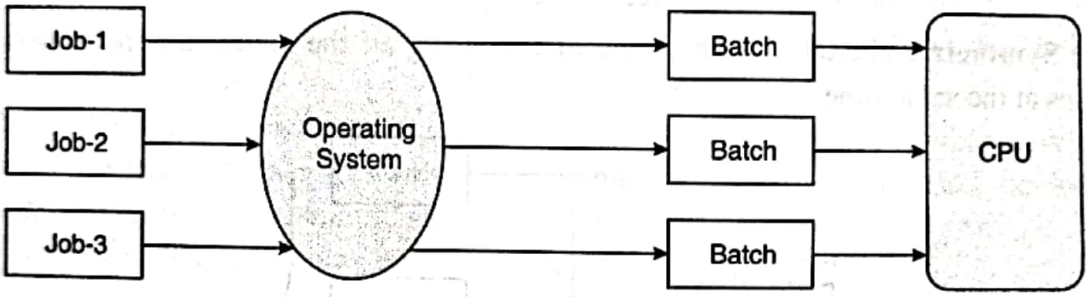


Fig. 8

Spooling : Spooling stand for simultaneous peripheral operation on line. Spooling refers to putting jobs in buffer, a special area in memory or on disk where a device can access them when it is ready.

The buffer provides a waiting station where data can rest while the slower device catches up.

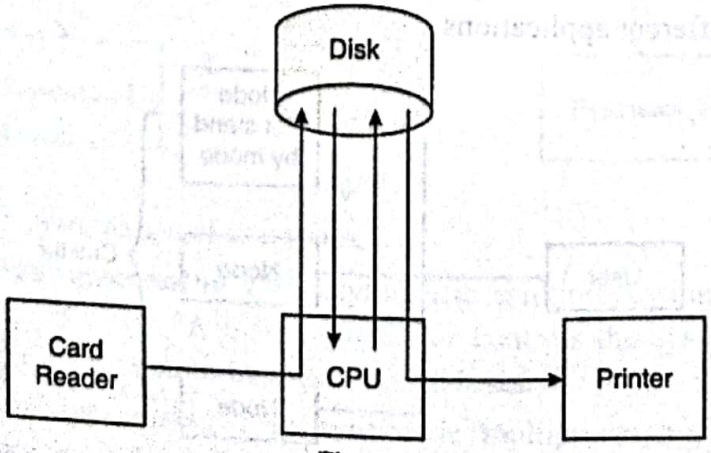


Fig. 9

2. Multiprogramming operating system : Multiprogramming is also the ability of an system to execute more than one program on a single processor machine. More than one task program/job/process can reside into the main memory at one point of time.

Multiprogramming increase CPU utilization by organizing jobs (code and data) so that the CPU always has one to execute.

The operating system keeps several jobs in memory simultaneously. This set of jobs can be a subset of job kept in the job pool which contains all jobs that enter the system. Since the number of jobs that can be kept simultaneously in memory is usually smaller than the number of job that can be kept in the job pool. The operating system picks and begins to execute one of the job in memory.

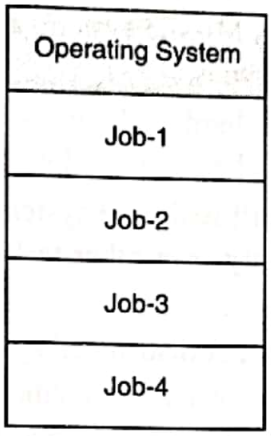
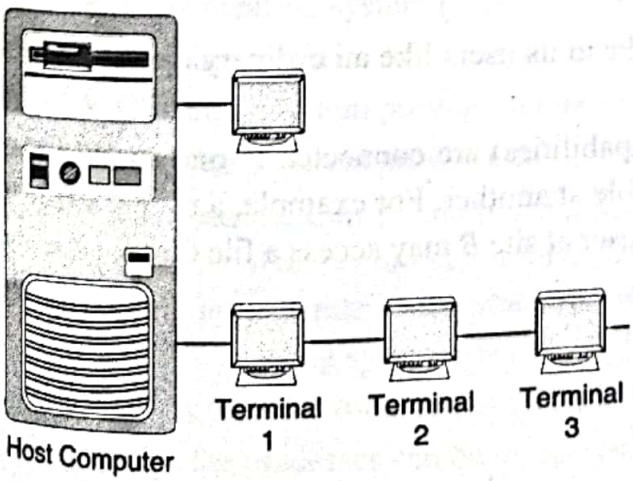


Fig. 10

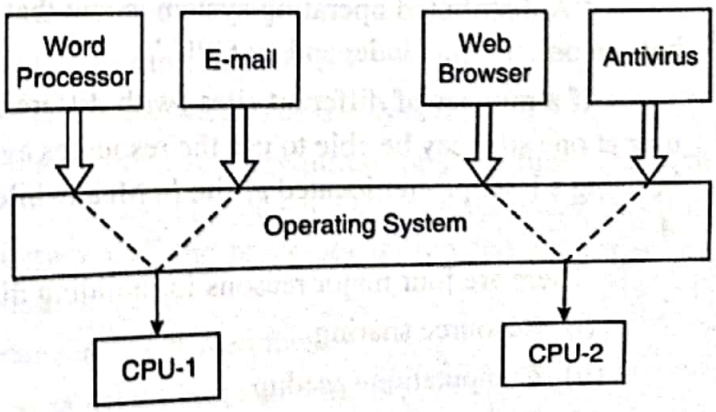
3. Multitasking/Time sharing operating system : Multiprogrammed system provides an environment in which the various system resources (for example, CPU, memory and peripheral devices) are utilized effectively, but they do not provide for user interaction with the system.

In time sharing (or multitasking), the CPU executes multiple jobs by switching among them, but the switches occur so frequently that the users can interact with each program while it is running.

Time sharing requires an interactive computer system which provides direct communication between the user and the system. The user gives instructions to the operating system or to program directly, using a input device such as a keyboard or mouse and waits for immediate results on an output device.



Time Sharing



Multi-tasking

Fig. 11

A time sharing operating system allows many users to share the computer simultaneously.

Multitasking allows multiple tasks or processes to use a computer system at a time. In multitasking the concurrent execution of multiple task or processes over a certain period of time.

Basically time sharing operating system and multitasking operating system increase the utilization of CPU.

4. Real time operating system : A real time system is used when rigid time requirements have been placed on the operation of a processor or the flow of data.

A real time system has well-defined, fixed time constraints. Processing must be done with the defined constraints or the system will fail. A real time system functions correctly only if it return the correct result within its time constraints. Missile system, air traffic control system are example of real time system real time systemf is of two types.

(a) Hard real time system : A hard real time system has the most stringent requirements, guaranteeing that critical real time tasks be completed within their deadlines.

(b) Soft real time system : A soft real time system is less restrictive, simply providing that a critical real time task will receive priority over other tasks and that it will retain that priority until it completes.

5. Distributed operating system : A distributed system is a collection of processors that do not share memory. Each processor has its own local memory. The processors communicate with one another through various communication networks, such as high speed buses or telephone line.

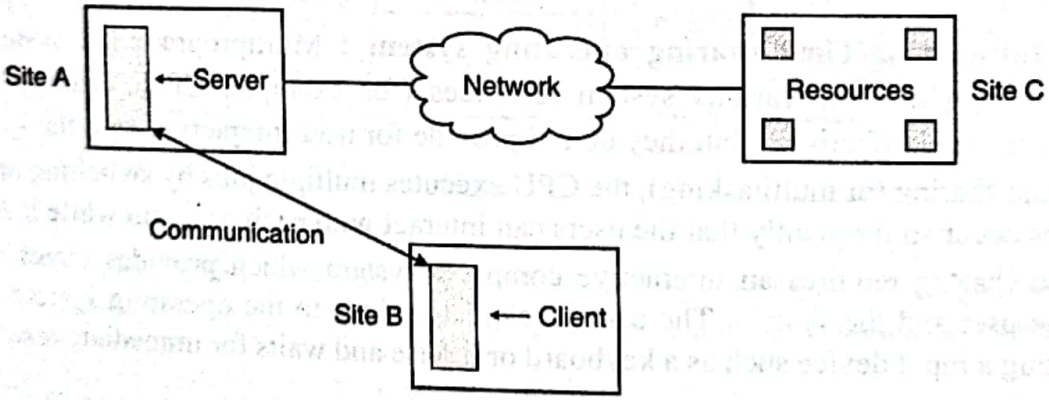


Fig. 12

"A distributed operating system is one that looks to its users like an ordinary operating system but run on multiple, independent CPU."

If a number of different sites (with different capabilities) are connected to one another, then a user at one site may be able to use the resources available at another. For example, a user at site *A* may be using a laser printer located at site *B*. Meanwhile, a user at site *B* may access a file that resides at site *A*.

There are four major reasons for building distributed system.

(i) Resource sharing
(ii) Computation speedup
(iii) Reliability
(iv) Communication

Operating System Services

The common services provided by the operating system is listed below.

1. Program execution
2. I/O operation
3. File system manipulation
4. Communication
5. Error detection
6. Resource allocation
7. Protection

1. Program execution : The operating system must be able to load a program into the memory for its execution. The program must complete its execution either normally or indicating error. Following are the major activities of on operating system with respect to program management.

- Loads a program into memory
- Executes the program
- Handles program's execution
- Provide a mechanism for process synchronization
- Provide a mechanism for process communication
- Provide a mechanism for dead lock handling

2. I/O operation : Program may require any I/O device while running. So operating system must provide the required I/O.

- 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. 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.
- 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 : Data transfer between two processes is required for some time. The both processes are on the one computer or on different computers but connected through computer network. The operating system manages communication between all the processes. 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.
- Communication may be implemented by two methods, either by shared memory or by message passing.

5. Error detection : Error 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 detection and error handling.

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

6. Resource management : If there are more than one user or jobs running at the same time, then resources must be allocated to each of them. Operating system manages different types of resources. Following are the major activities of an operating system with respect to resource management.

- The operating system manages all kinds of resources using schedulers.
- CPU scheduling alogrithms are used for better utilization of CPU.

7. Protection : In protection, all the access to the resources is controlled. In multiprocess environment, it is possible that, one process to interface with the other, or with the operating system, so protection is required.

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

- The operating system that all access to system resources is controlled.
- The operating system ensures that external I/O devices are protected from invalid access attempts.

1.3. User Operating System Interface

A user interface refers to a part of an operating system, program, or device that allows user to enter and receive information. There are two fundamental approaches for users to interface with the operating system. One technique is to provide a command-line interface or command interpreter that allows users to directly enter commands that are to be performed by the operating system. The second approach allows the user to interface with the operating system via a graphical user interface or GUI.

1. Command line interface : Some operating system can still be used with a text based user interface. In this case the commands are entered as text. Many of the commands given at this level manipulate files; create, delete, list, print, copy, execute and so on. The MS-DOS and UNIX Shells operate in this way.

rm file text.

To display the text-based command Prompt in Windows, open the start menu and type cmd. Press Enter on the keyboard to launch the Command Prompt in a separate Window. With the command prompt, you can type your command from keyboard.

2. Graphical user interface : A second strategy for interfacing with the operating system is through a user friendly graphical user interface or GUI. In GUI the mouse is moved to position its pointer on images, or icons, on the screen (the desktop) that represents programs, files, directories and system functions.

GUI is an interface that allows users to interact with different electronic devices using icons and others visual indicators. In today's time, graphical user interfaces are used in many devices such as mobiles, MP3 players, gaming devices, smart phones, etc.

System calls

System calls provide an interface to the services made available by an operating system. 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.

Kernel : A kernel is the central part of an operating system. It manages the operations of the computer and the hardware memory, and CPU time.

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

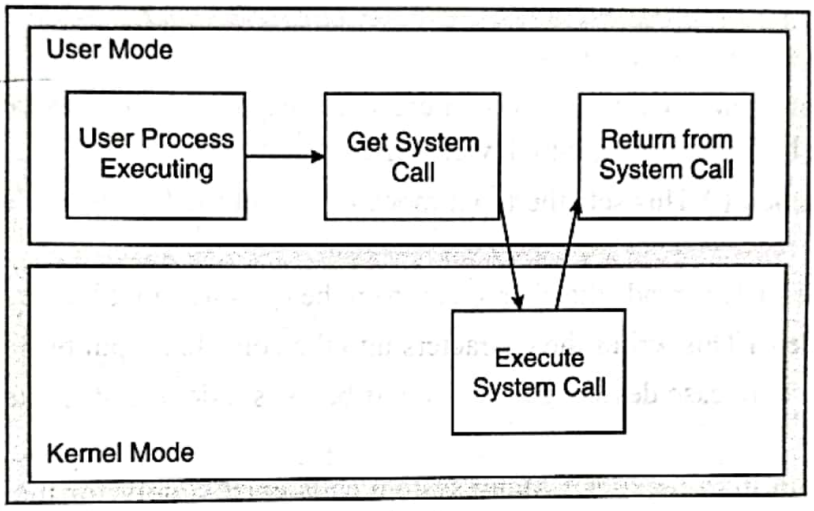


Fig. 13

In general, system calls are required in the following situations :

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

Types of system calls

There are mainly five types of system calls these are explained as follows.

1. Process control : A process is a basic entity in the system. The process in the system need to be created, deleted and aborted. These many operations are required on the process for their management. These are some example.

Fork ( ) : → Create a process.

Exit ( ) : → Terminate a process.

Kill ( ) : → Terminate a process abnormally.

Nice ( ) : → Increase a priority of a process.

2. File management : Creation, deletion, opening, closing, reading and writing are some general operations on files Similarly for organizing files, there is a directory system and thereby system managing them.

Create ( ) To create a new file

Open ( ) To open a file

Close ( ) To close a file

Read ( ) To read a file

Write ( ) To write a file

L seek ( ) Change the position of the read write pointer

Link ( ) Give another name to a file

Unlink ( ) Delete a file in a directory

Mk dir ( ) Create a new directory.

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

Set console mode ( ) This sets the input mode or output mode, console's input buffer or output buffer respectively.

Read console ( ) This reads the characters from the console input buffer.

Write console ( ) This writes the characters into the console output buffer.

Request device, release device, get device attributes, set device attributes are also operation of device management.

4. Information maintenanes : Many system calls exist simply for the purpose of transfering information between the user program and the OS.

get time ( )	**set date ( )**	**get process ( )**
set time ( )	**get system data ( )**	**set process ( )**
get date ( )	**set system data ( )**	

5. Communications : There is a need for communication among the processes in the system. General operation of communication are opening and closing the connection, sending and receiving messages, reading and writing messages and so on.

Msg snd ( )	sending a message
Msg rcv ( )	receiving a message

These system calls may be related to the communication between processes either on the same machine or between processes on different nodes of a network. Thus inter process communication is provided by the operating system through these communication related system calls.

System Programs

System programs provide a convenient environment for program development and execution. System program can be defined as act of building systems software using system programming languages. Some of system programs are user interfaces to system calls.

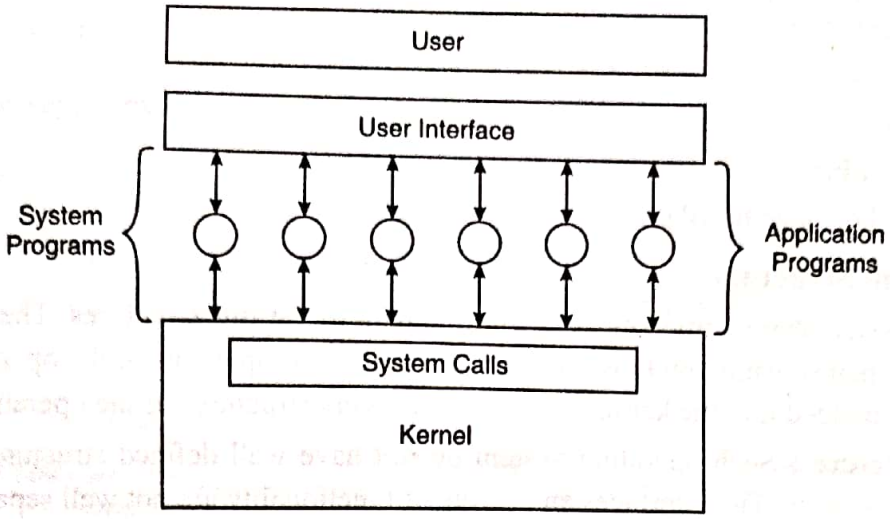


Fig. 14

System programs can be divided into these categories :.

1. File management : A file is a collection of specific information stored in memory of computer system. File management is defined as process of manipulating file in computer system, its management includes process of creating, modifying and deleting files. It helps to stores files in separate folders known as directories. File management generally manipulate files and directories.

2. Status information : Information like date, time amount of available memory, or disk space is asked by some users. Others providing detailed performance, logging and debugging information which is more complex. All this information is formatted and displayed on output devices and print the output to the terminal.

3. File modification : Several text editors may be available to create and modify the content of files stored on disk or other storage devices.

4. Programming-language support : Compilers, assemblers, debuggers and interpreters for common programming languages (such as, C, C ++, Java, visual basic) are often provided to the user with the operating system.

5. Program loading and execution : When a program is ready after assembling and compilation, it must be loaded into memory for execution. A loader is part of an operating system that is responsible for loading programs and libraries. Loaders, relocatable loaders, linkage editors and overlay loaders are provided by system.

6. Communications : These programs provide the mechanism for createing virtual connections among processes, users, and computer systems. User can send messages to other user on their screen, user can send email, browsing on web pages, remote login, transformation of files from one user to another.

Some examples of system programs in operating system are :

- Windows 10
- MAC OSX
- Ubuntu
- Linux

- Unix
- Android
- Anti virus
- Disk formatting
- Computer language translators.

Operating System Structure

Operating system can be implemented with the help of various structures. The structure of the operating system depends mainly on how the various common components of the operating system are interconnected and melded into the kernel. We have following structures of the operating system.

Simple structure : Such operating system do not have well defined structure and are small, simple and limited systems. The interfaces and levels of functionality are not well separated. MS-DOS is an example of such operating system. In MS-DOS application programs are able to access the basic I/O routines.

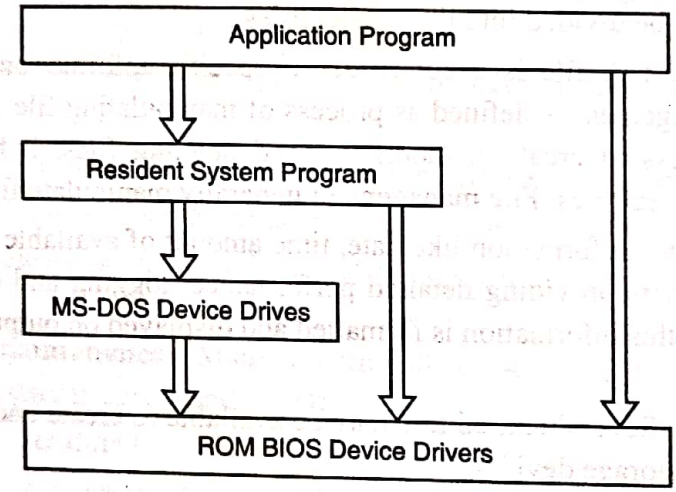


Fig. 15

Layered structure : In this structure the OS is broken into number of layers (levels). The bottom layer (Layer O) is the hardware and the top most layer (Layer N) is the user interface. These layers use the functions of the lower level layers only.

The main advantage of the layered approach is simplicity of construction and debugging. The layers are selected so that each uses functions (operations) and services of only lower level layers. UNIX is an example of this structure.

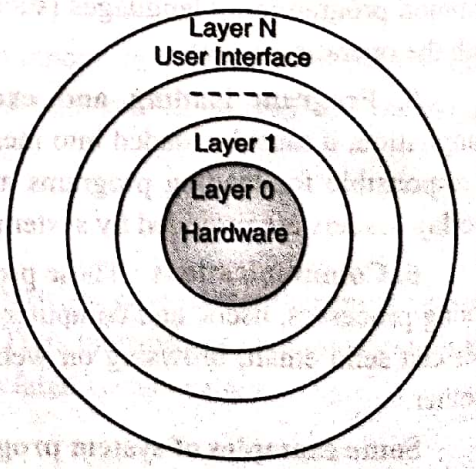


Fig. 16

Microkernels : This structure designs the operating system by removing all non-essential components from the kernel and implementing them as system and user programs. this result in a smaller kernel called the micro-kernel.

Advantages of this structure are that all new servies need to be added to user space and does not require the kernel to be modified. MAC OS is an example of this type of OS.

Modular structure : It is considered as the best approach for an OS. It involves designing of a modular kernel. The kernel has only set or core components and other services are added as dynamically loadable modules to the kernel either during run time or boot time. For example the solaris operating system structure, shown in figure.

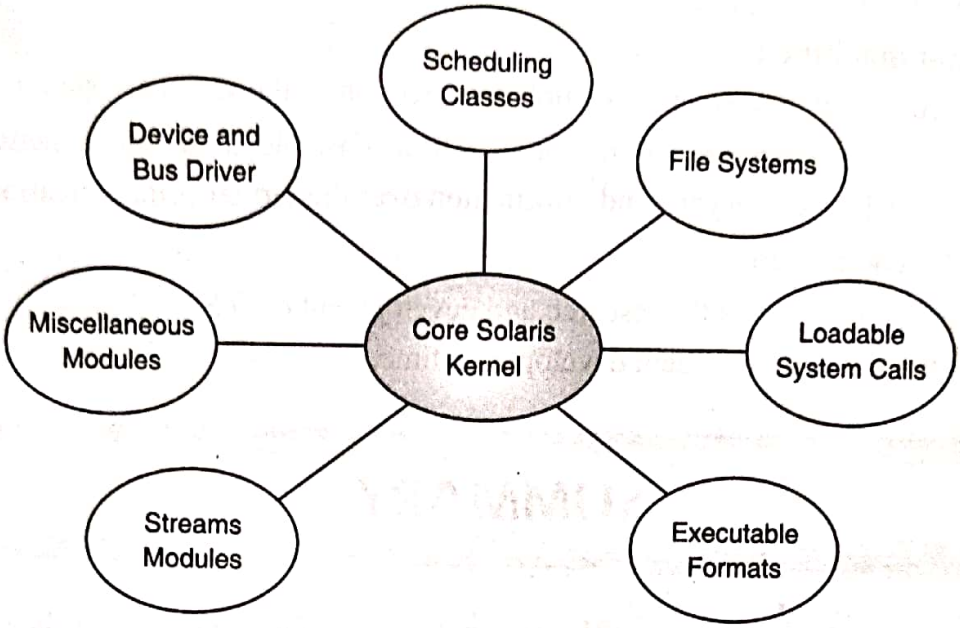
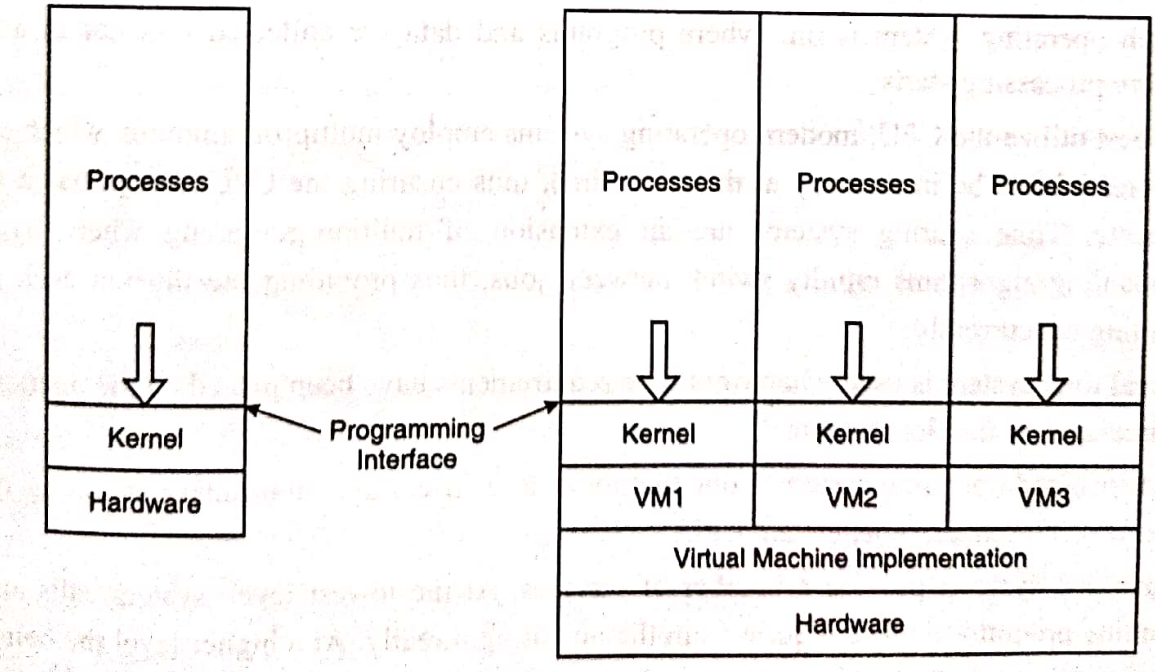


Fig. 17

Virtual Machines

Virtual machines are based on computer architectures and provide functionality of a physical computer. Their implementations may involve specialized hardware, software.



(a) Non-Virtual Machine

(b) Virtual Machine

Fig. 18

The fundamental idea behind, a virtual machine is to abstract the hardware of a single computer (the CPU, memory, disk drive, network interface cards and so forth) into several different execution environments, thereby creating the illusion (misunderstanding) that each separate execution environment is running its own private computer.

Benefits of virtual machine :

1. No protection problems (there is complete protection of the various system resources).
2. There is no direct sharing of resources. (It is possible to define a network of virtual machines, each of which can send information over the virtual communication network).
3. It provides good security.
4. Virtual machine supports the research and development of O.S.
5. It solves the problem of system development time.

SUMMARY

An operating system is system software that manages the computer hardware as well as providing an environment for application program to run.

Multiprocessor operating system allows the multiple processors and these processors are connected with physical memory, computer buses and peripheral devices. Main objective of using multiprocessor operating system is to consume high computing power an increase the execution speed of system.

Batch operating system is one where programs and data are collected together in a batch before processing starts.

To best utilize the CPU, modern operating systems employ multiprogramming, which allows several jobs to be in memory at the same time, thus ensuring the CPU always has a job to execute. Time sharing systems are an extension of multiprogramming where by CPU scheduling algorithms rapidly switch between jobs, thus providing the illusion each job is running concurrently.

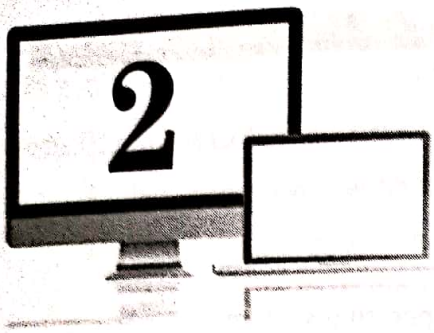
A real time system is used when rigid time requirements have been placed on the operation of a processor or the flow of data.

A distributed operating system is one that looks to its users like on ordinary operating system but run on multiple, independent CPU.

Operating systems provide a number of services. At the lowest level, system calls allow a running program to make request from the operating directly. At a higher level the command interpreter or shell provides a mechanism for a user to issue a request without writing a program.

EXERCISE

1. Write down the advantage of batch processing system.
2. What are the major function of operating system?
3. Explain the main features of an real time operating system.
4. Draw the layered structure of an operating system.
5. What is an operating system? Discuss the main services of the operating system.
6. Discuss the differences between a time sharing system and real time system.
7. Discuss the objectives of the multiprocessor system.
8. Explain multitasking.
9. What do you understand by system call? How is a system call made? How is a system call handled by the system? Choose suitable examples for explanation.
10. What are the operating system services?
11. What do you understand by system programs?
12. What are the virtual machine and also explain its benefits.



2

PROCESS MANAGEMENT

Process concept, Process State, Process Control Block, Scheduling Queues, Scheduler, Job Scheduler, Process Scheduler, Context Switch, Operations on Processes, Interprocess Communication, Shared Memory Systems, MessagePassing Systems, CPU Scheduler, Scheduling Criteria, Scheduling Algorithms, Preemptive and Non Preemptive, First come first serve (FCFS), Shortest Job first (SJF), Round Robin (RR), Multiprocessor scheduling, Process Synchronization.

Process management involves various tasks like creation, scheduling, termination of processes, and a dead lock. Processes is a program that is under execution, which is an important part of modern day operating system.

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

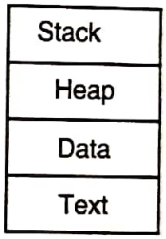


Fig. 2.1 : Process architecture

Process : Process is the execution of a program that performs the actions specified in that program. It can be defined as an execution unit where 0 program runs. The operating system helps you to create, schedule, and terminates the processes which is used by CPU.

Stack : The stack stores temporary data like function parameters, returns addresses and local variables.

Heap : Allocate memory, which may be processed during its run time.

Data : It contains the variable.

Text : Text section includes the current activity, which is represented by the value of the program counter.

Process State

The state of a process is defined in part by the current activity of that process. Each process may be in one of the following states.

New : The process is being created.

Running : Instructions are being executed.

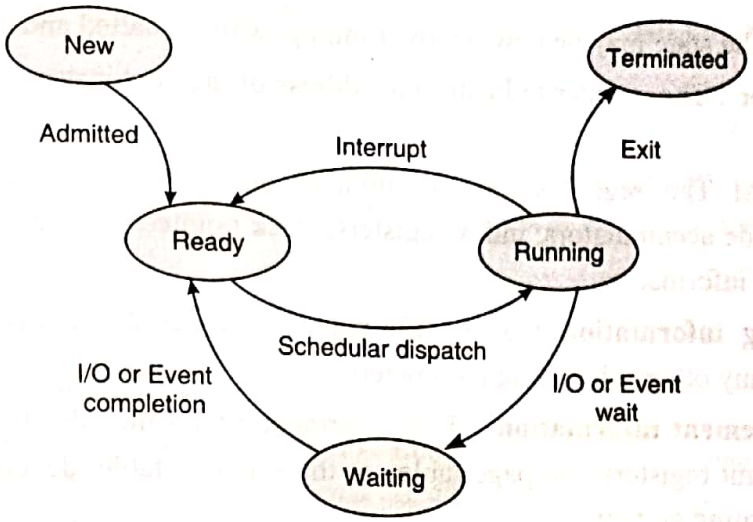


Fig. 2.2 : Diagram of Process state.

Waiting : The process is waiting for some event to occur (such as an I/O completion or reception of a signal)

Ready : The process is waiting to be assigned to a processor.

Terminated : The process has finished execution.

Process Control Block

A process control block (PCB) is a data structure used by computer operating systems to store all the information about a process. It is also called a task control block.

- When a process is created, the operating system creates a corresponding process control block.
- Information in a process control block is updated during the translation of process states.
- When the process terminates, its PCB is returned to the pool from which new PCBs are drawn.
- Each process has a single PCB

Role : The role of the PCBs is central in process management. They are accessed and/or modified by most utilities, particularly those involved with scheduling and resource management.

Structure :

Process state
Process number
Program counter
Registers
Memory limits
List of open files
...

Process state : The state may be new, ready, running, waiting halted and so on.

Program counter : The counter indicates the address of the next instruction to be executed for this process.

CPU Registers : The registers vary in number and type, depending on the computer architecture. They include accumulators, index registers, stack pointers and general purpose registers, plus any condition code information.

CPU Scheduling information : This information includes a process priority pointers to scheduling queues and any other scheduling parameters.

Memory-management information : This information may include such information as the value of the base and limt registers, the page tables or the segment table, depending on the memory system used by the operating system.

Accounting information : This information includes the amount of CPU and real time used, time limits, account numbers, job or process numbers and so on.

I/O status information : This information includes the list of I/O devices allocated to the process, a list of open files and so on.

2.2. Scheduling Queues

The operating sytem maintains all process control blocks in process scheduling queues. The operating system maintains a separate queue for each of the process states and process control block of all processes in the same execution state are placed in the same queue. When the state of a process is changed, its process control block is unlinked from its current queue and moved to its new state queue.

The operating system maintains the following important process scheduling queue.

Job queue : This queue keeps all the processes in the system.

Ready queue : This queue keeps a set of all processes residing in mainmemory, ready and waiting to execute. A new process is always put in this queue.

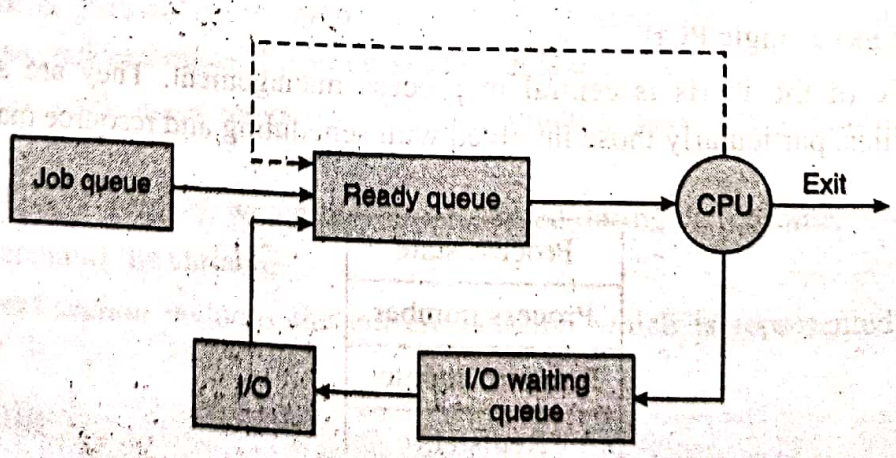


Fig. 2.3

Device queues : The processes which are blocked due to unavailability of an I/O device constitute this queue.

The operating system can use different policies to manage each queue (FIFO, Round Robin, Priority, etc.).

2.3. Scheduler

Scheduler 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. Scheduler are of three types.

- Long-term scheduler (Job scheduler)
- Short-term scheduler (CPU scheduler)
- Medium-term scheduler.

1. Long-term scheduler (Job scheduler) : Long-term scheduler is also called job scheduler. A long term scheduler determines which programs are admitted to the system for processing. It selects process from the queue and loads them into memory for execution. Process loads into the memory for CPU scheduling.

2. Short-term scheduler (CPU scheduler) : Short term scheduler is also called 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 select a process among the processes that are ready to execute and allocates CPU to one of them.

3. Medium term scheduler : Medium term scheduler is a part of swapping. It removes the processes from the memory. It reduces the degree of multiprogramming. The medium term scheduler is incharge of handling the swapped out processes.

2.4. Context Switch

A context switch is the mechanism to store and restore the state or context of a CPU in process control block so that a process execution can be resumed from the same point at a later time. Using this technique, a context switcher enables multiple processes to share a single CPU. Context switching is an essential part of a multitasking operating system features.

When the scheduler switches the CUP from excecuting one process to execute another, the state from the current running process is stored in to process control block.

Operations on Processes

There are many operations that can be performed on processes. Systems must provide a mechanism for process creation and termination.

Process creation : A process may create several new processes, via a create-process system call, during the course of execution. The creating process is called a parent process, and the new processes are called the children of that process. Each of these new processes may in turn create other processes, forming a tree of processes.

When a process creates a new process two possibilities exist in term of execution.

1. The parent continues to execute concurrently with its children.
2. The parent waits until some or all of its children have terminated.

There are also two possibilities in terms of the address space of the new process.

1. The child process is a duplicate of the parent process (It has the same program and data as the parent).
2. The child process has a new program loaded in to it.

Process termination : After the process has completed the execution of its last instruction it is terminated. The resources held by a process are released after it is terminated.

A child process can be terminated by its parent process if its task is no longer relevant.

A parent may terminate the execution of one of its children for a variety of reasons, such as these :

1. The child has exceeded its usage of some of the resources that it has been allocated.
2. The task assigned to the child is no longer required.
3. The parent is exiting, and the operating system does not allow a child to continue if its parent terminates.

2.5. Interprocess Communication

Inter process communication is the mechanism provided by the operating system that allows processes to communicate with each other.

The operating system may be either independent processes or cooperating processes. A process is independent if it can not affect or be affected by the other process executing in the system. Any process that does not share data with any other process is independent. A process is cooperating if it can affect or be affected by the other processes executing in the system. Clearly any process that shares data with other processes is a co-operating process.

There are several reasons for providing an environment that allows process cooperation.

Information sharing : Since several users may be interested in the same piece of information, we must provide an environment to allow concurrent access to such information.

Computation speedup : If we want a particular task to run faster, we must break it into subtasks, each of which will be executing in parallel with the others.

Modularity : We may want to construct the system in a modular fashion, dividing the system function into separate processes or threads.

Convenience : Even an individual user may work on many tasks at the same time.

Cooperating processes require an interprocess communication mechanism that will allow them to exchange data and information. There are two fundamental models of interprocess communication.

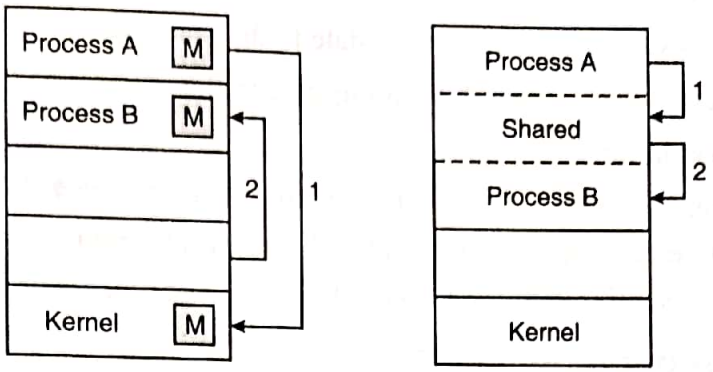
1. Shared memory
2. Message passing

Shared memory system : Shared memory is the memory that can be simultaneously accessed by multiple processes. This is done so that the processes can communicate with each other.

Interprocess communications using shared memory requires communicating processes to establish a region of shared memory.

To illustrate the concept of cooperating processes, let's consider the producer-consumer problem, which is a common paradigm for cooperating processes. A producer process produces

information that is consumed by a consumer process. For example, a compiler may produce assembly code, which is consumed by an assembler. The assember, in turn may produce object modules, which are consumed by the loader.



(a) Message passing (b) Shared memory

Fig. 2.4

Message-passing systems : Message passing provide a mechanism to allow processes to communicate and to synchronize their action without sharing the same adderss space and is particularly useful in a distributed environment, where the communicating processes may reside on different computers connected by a network. For example, a chat program used on the world wide Web could be designed so that chat participants communicate with one another by exchanging messages.

A message-passing facility provides at least two operations; send (message) and receive (message). Here are several methods for logically implementing the send ( )/Receive ( ) operations.

- Direct or indirect communication
- Synchronous or asynchronous communication
- Automatic or explicit buffering

Scheduling Criteria

Many criteria have been suggested for comparing CPU scheduling algorithm-which characteristics are used for comparison can make a substantial difference in which algorithm is judged to be best. The criteria include the following.

1. CPU utilization : CPU utilization is the average function of time, during which the processor is busy. CPU utilization can range from 0 to 100 per cent.

2. Through put : If the CPU is busy executing processes, then work is being done. One measure of work is the number of processes that are completed per time unit, called through put.

3. Turn around time : The interval from the time of submission of a process to the time of completion is the turnaround time.

4. Waiting time : Waiting time is the sum of the period spent waiting in the ready queue.

5. Response time : Response time is the time from the submission of a request untill the first response is produced.

Preemptive and non preemptive scheduling

CPU scheduling decisions may take place under the following four circumstances.

1. When a process switches from the running states to the waiting state.
2. When a process switches from running state to the ready state.
3. When a process switches from the waiting state to the ready state.
4. When a process terminates.

When scheduling takes place only under circumstances 1 and 4. We say that the scheduling scheme is nonpreemptive (Example : The FCFS scheduling algorithm is nonpreemptive) otherwise scheduling scheme is preemptive. (Round Robin scheduling algorithm is preemptive).

2.6. Scheduling Algorithms

There are many different CPU scheduling algorithms.

1. First come first seved scheduling (FCFS) : FCFS is the simplest scheduling method. CPU is allocated to the process in the order of arrival. The process that requests the CPU first is allocated the CPU first.

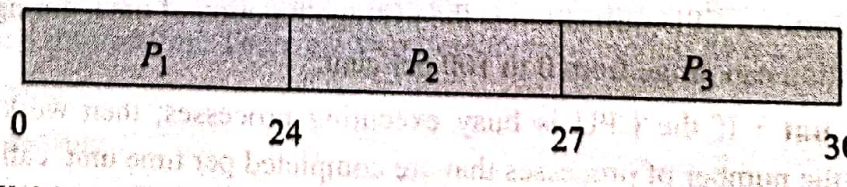
FCFS is also called FIFO. FCFS is nonpreemptive scheduling algorithm implementation of the FCFS policy is easily managed with a FIFO queue.

Consider the following set of processes that arrive at time 0, with length of the CPU burst given in milliseconds.

Process	Burst time
P_1	24
P_2	3
P_3	3

If the processes arrive in the order of $P_1P_2P_3$ and are served in FCFS order. Calculate the average waiting time and average turn around time.

Gantt chart :



Waiting time

$P_1 = 0$

$P_2 = 24$

$P_3 = 27$

Rurn around time

$P_1 = 24$

$P_2 = 27$

$P_3 = 30$

Average waiting time $= \frac{0+24+27}{3}$

$= \frac{51}{3}$

= 17 Milliseconds

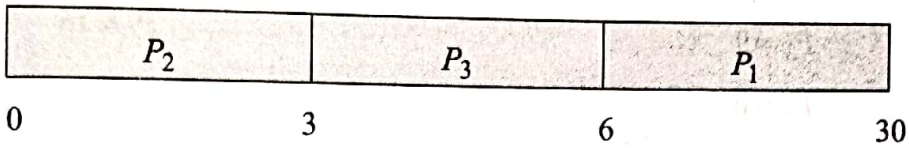
Average turn around time

$= \frac{24+27+30}{3}$

$= \frac{81}{3}$ **= 27 Milliseconds**

If the processes arrive in the order of $P_2 P_3 P_1$ and are served in *FCFS* order. Calculate the average waiting time and average turn around time.

Gantt chart :



Waiting time

$P_1 = 6$

$P_2 = 0$

$P_3 = 3$

Average waiting time $= \frac{6+0+3}{3}$

$= \frac{9}{3}$

= 3 Millisecond

Turn around time

$P_1 = 30$

$P_2 = 3$

$P_3 = 6$

Average turn around time $= \frac{30+3+6}{3}$

$= \frac{39}{3}$

= 13 Milliseconds

2. Shortest-Job-first Scheduling : Shortest job first scheduling is a preemptive or non preemptive algorithm. In the shortest job first scheduling algorithm the job having shortest or less burst time will get the CPU first. It is the best approch to minimize the waiting time.

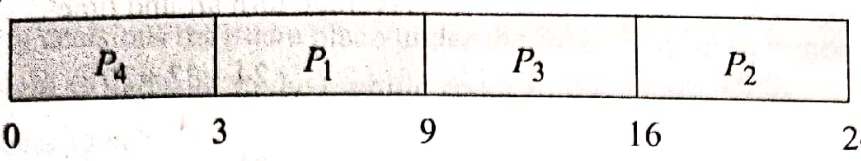
SJF improves the through put of the process by ensuring that the shorter jobs are executed first, thus the possibility of less turn around time.

As an example of SJF scheduling, consider the following set of processes, with the length of the CPU burst given in milliseconds.

Process	Burst time
P_1	6
P_2	8
P_3	7
P_4	3

Arrival time of the process is 0 and processes arrive in the order of P_1, P_2, P_3 and P_4. The gantt chart, waiting time and average turn around time is given below.

Gantt chart :



Waiting time

$P_1 = 3$

$P_2 = 16$

$P_3 = 9$

$P_4 = 0$

Average waiting time

$$= \frac{3+16+9+0}{4}$$

$$= \frac{28}{4}$$

= 7 Milliseconds

Turn around time

$P_1 = 9$

$P_2 = 24$

$P_3 = 16$

$P_4 = 3$

Average turn around time

$$= \frac{9+24+16+3}{4}$$

$$= \frac{52}{4}$$

= 13 Milliseconds

3. Priority scheduling : CPU is allocated to the highest priority of the process from ready queue. Each process has a priority number. If two or more processes have the same priority, then FCFS algorithm is applied for solving.

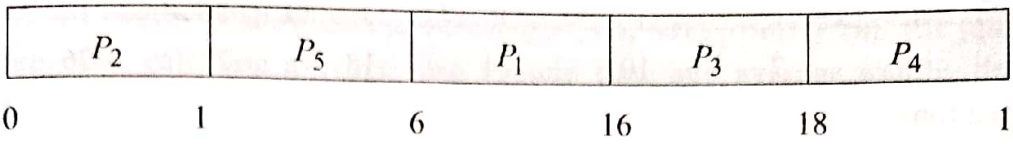
Priority scheduling is preemptive or non preemptive. Priority of the process can be defined either internally or externally. In priority scheduling we assume that low number represent high priority.

As an example, consider the following set of processes, assumed to have arrived at time 0, in the order of P_1, P_2, P_3, P_4 and P_5, with the length of the CPU burst given in milliseconds.

Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	4
P_4	1	5
P_5	5	2

The Gantt chart average waiting time and average turn around time is given below.

Gantt chart :



Waiting time

$P_1 = 6$

$P_2 = 0$

$P_3 = 16$

$P_4 = 18$

$P_5 = 1$

Average waiting time $= \frac{6+0+16+18+1}{5}$

$= \frac{41}{5}$

= 8.2 milliseconds

Turn around time

$P_1 = 16$

$P_2 = 1$

$P_3 = 18$

$P_4 = 19$

$P_5 = 6$

Average turn around time

$= \frac{16+1+18+19+6}{5}$

$= \frac{60}{5}$ **= 12 milliseconds**

4. Round Robin scheduling : Round Robin scheduling (RR) is similar to FCFS scheduling, but preemption is added to switch between processes. A small unit of time, called a time quantum or time slice, is defined.

A time quantum is generally from 10 to 100 milliseconds. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum.

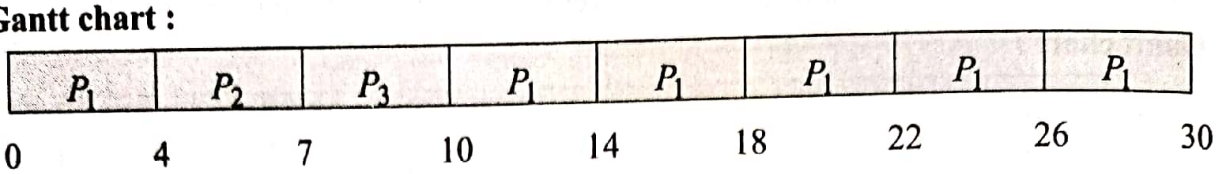
The CPU scheduler picks the first process from ready queue, sets a timer to interrupt after 1 time quantum and dispatches the process.

Consider the following set of processes that arrive at time 0, with the length of the CPU durst given in milliseconds.

Process	Burst time
P_1	24
P_2	3
P_3	3

If we use a time quantum of **4 milliseconds,** then process P_1 gets the first 4 milliseconds. Since it requires another 20 milliseconds, it is preempted after the first time quantum and the CPU is given to the next process in the queue, process P_2. Since process P_2 does not need 4 milliseconds it quits before its time quantum expires. The CPU is then given to the next process, process P_3. Once each process has received 1 time quantum, the CPU is returned to process P_1 for an additional time quantum.

Gantt chart :



Waiting time

$$P_1 = 0 + 3 + 3 = 6$$
$$P_2 = 4$$
$$P_3 = 7$$

Average waiting time

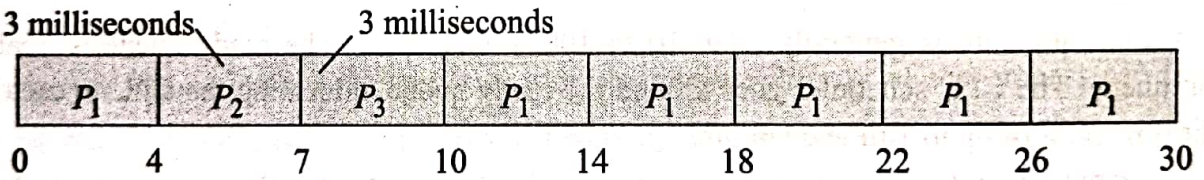
$$= \frac{6+4+7}{3}$$
$$= \frac{17}{3}$$
$$= \mathbf{5.66 \text{ milliseconds}}$$

Turn around time :

$$P_1 = 30$$
$$P_2 = 7$$
$$P_3 = 10$$

$$\text{Average turn around time} = \frac{30+7+10}{3} = \frac{47}{3}$$

$$= \mathbf{15.66 \text{ Milliseconds}}$$



Process P_1 wait for processing when process P_2 and process P_3 are in processing. Process P_2 3 milliseconds and process P_3 is also 3 milliseconds are in processing. Total process P_2 and P_3 are in processing for 6 milliseconds so waiting time for P_1 procssing is 6 milliseconds.

Example : Consider following process, with the burst time given in milliseconds.

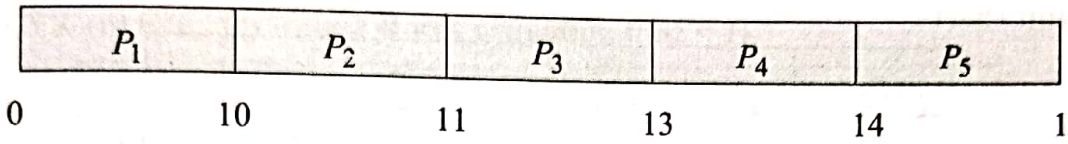
Process	Burst time	Priority
P_1	10	3
P_2	1	1
P_3	2	3
P_4	1	4
P_5	5	2

Processes are arrived in P_1, P_2, P_3, P_4, P_5 order of all at time 0.

Draw Gantt charts to show execution using FCFS, SJF, priority and RR (quantum time = 1) scheduling. Also calculate average turn around time and average waiting time for each scheduling algorithms.

Solution : (a) FCFS

Gantt chart :



Waiting time	Turn around time
$P_1 = 0$	$P_1 = 10$
$P_2 = 10$	$P_2 = 11$
$P_3 = 11$	$P_3 = 13$
$P_4 = 13$	$P_4 = 14$
$P_5 = 14$	$P_5 = 19$

Average waiting time

$$= \frac{0+10+11+13+14}{5}$$

$$= \frac{48}{5} = 9.6 \text{ Milliseconds}$$

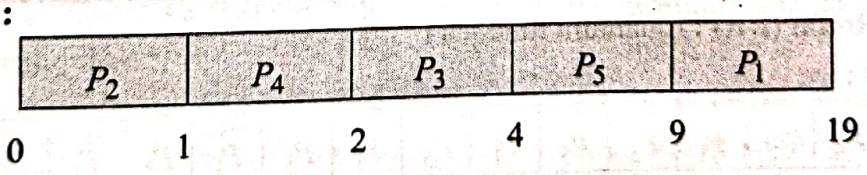
Average turn around time

$$= \frac{10+11+13+14+19}{5}$$

$$= \frac{67}{5} = 13.4 \text{ Milliseconds}$$

(B) SJF :

Gantt chart :



Waiting time	Turn around time
$P_1 = 9$	$P_1 = 19$
$P_2 = 0$	$P_2 = 1$
$P_3 = 2$	$P_3 = 4$
$P_4 = 1$	$P_4 = 2$
$P_5 = 4$	$P_5 = 9$

Average waiting time

$$= \frac{9+0+2+1+4}{5}$$

$$= \frac{16}{5} = 3.2 \text{ Milliseconds}$$

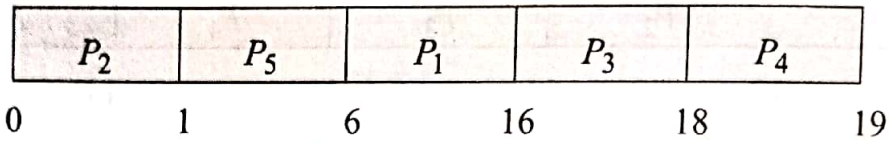
Average turn around time

$$= \frac{19+1+4+2+9}{5}$$

$$= \frac{35}{5} = 7 \text{ Milliseconds}$$

(c) Priority :

Gantt chart :



Waiting time

$P_1 = 6$

$P_2 = 0$

$P_3 = 16$

$P_4 = 18$

$P_5 = 1$

Turn around time

$P_1 = 16$

$P_2 = 1$

$P_3 = 18$

$P_4 = 19$

$P_5 = 6$

Average waiting time

$$= \frac{6+0+16+18+1}{5}$$

$$= \frac{41}{5}$$

$= 8.2$ **Milliseconds**

Average turn around time

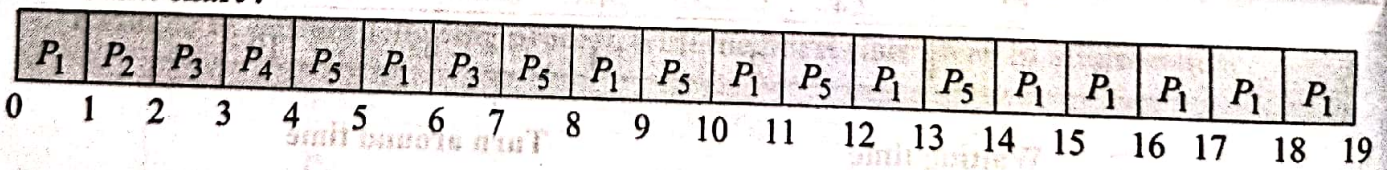
$$= \frac{16+1+18+19+6}{5}$$

$$= \frac{60}{5}$$

$= 12$ **Milliseconds**

(d) Round Robin (RR) : (quantum time = 1)

Gantt chart :



Waiting time

$P_1 = 0+1+2+1+5 = 9$

$P_2 = 1$

$P_3 = 2+1+1+1 = 5$

$P_4 = 3$

$P_5 = 4+4+1 = 9$

Turn around time

$P_1 = 19$

$P_2 = 2$

$P_3 = 7$

$P_4 = 4$

$P_5 = 14$

Average waiting time

$$= \frac{9+1+5+3+9}{5}$$

$$= \frac{27}{5} = 5.4 \text{ Milliseconds}$$

Average turn around time

$$= \frac{19+2+7+4+14}{5}$$

$$= \frac{46}{5} = 9.2 \text{ Milliseconds}$$

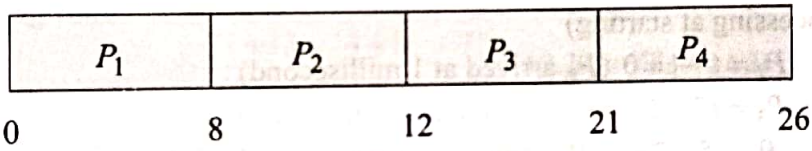
Example : Calculate average turn around time and average waiting time for the following algorithm.

(a) FCFS (b) SJF (c) Round Robin (quantum time = 1)

Process	Arrival time	Burst time
P_1	0	8
P_2	1	4
P_3	2	9
P_4	3	5

Solution : (a) FCFS

Gantt chart :



Waiting time

$$P_1 = 0$$
$$P_2 = 8-1 = 7$$
$$P_3 = 12-2 = 10$$
$$P_4 = 21-3 = 18$$

- P_2 is arrived at 1 millisecond so waiting time of P_2 starts after 1 millisecond.
- P_3 is arrived at 2 millisecond so waiting time of P_3 starts after 2 milliseconds.
- P_4 is arrived at 3 milliseconds so waiting time of P_4 starts after 3 milliseconds.

Average waiting time

$$= \frac{0+7+10+18}{4}$$

$$= \frac{35}{4} = 8.75 \text{ Milliseconds}$$

Turn around time :

$$P_1 = 8$$
$$P_2 = 12-1 = 11$$
$$P_3 = 21-2 = 19$$
$$P_4 = 26-3 = 23$$

Average turn around time

$$= \frac{8+11+19+23}{4}$$

$$= \frac{61}{4}$$

= 15.25 milliseconds

T.A.T. = Burst time + waiting time

OR

T.A.T. = Completion time – arrival time

$P_1 = 8+0 = 8$

OR

$8-0=8$

$P_2 = 4+7 = 11$

OR

$12-1=11$

$P_3 = 9+10 = 19$

OR

$21-2=19$

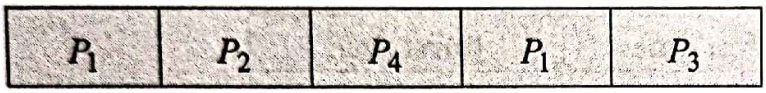
$P_4 = 18+5 = 23$

OR

$26-3=23$

(b) SJF :

Gantt chart :



0 1 5 10 17 26

Process P_1 arrived at 0 millisecond so CPU assigned P_1 for processing but after 1 millisecond process P_2 is arrived which smallest process than process P_1 so P_2 assigned by CPU for processing after 1 millisecond.

Waiting time

$P_1 = 10-1=9$ (Because P_1 is 1 millisecond is in processing at starting)

$P_2 = 1-1=0$ (P_2 arrived at 1 millisecond)

$P_3 = 17-2=15$ (P_3 arrived at 2 milliseconds)

$P_4 = 5-3=2$ (P_4 arrived at 3 milliseconds)

Average waiting time

$$= \frac{9+0+15+2}{4} = \frac{26}{4} = \textbf{6.5 milliseconds}$$

Turn arround time :

$P_1 = 17-0=17$

$P_2 = 5-1=4$

$P_3 = 26-2=24$

$P_4 = 10-3=7$

Average turn around time

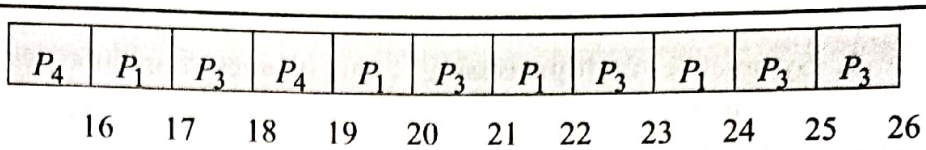
$$= \frac{17+4+24+7}{4}$$

$$= \frac{52}{4} = \textbf{13 milliseconds}$$

(c) Round Robin : (Quantum time = 1)

Gant chart

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15



Waiting time :

$P_1 = 0 + 4 + 7 + 5 = 16$ (When 4 milliseconds P_2, 7 milliseconds P_3 and 5 milliseconds P_4 are in processing then P_1 is wait)

$P_2 = 3 + 3 + 3 = 9$ (3 milliseconds P_1, 3 milliseconds P_3, 3 milliseconds P_4)

$P_3 = 7 + 3 + 5 = 15$ (7 m.s P_1, 3 m.s P_2, 5 m.s P_4)

$P_4 = 4 + 3 + 4 = 11$ (4 m.s P_1, 3 m.s P_2, 4 m.s P_3)

Average waiting time

$$= \frac{16 + 9 + 15 + 11}{4}$$

$$= \frac{51}{4} = \textbf{12.75 milliseconds}$$

Turn around time

$P_1 = 24 - 0 = 24$

$P_2 = 14 - 1 = 13$

$P_3 = 26 - 2 = 24$

$P_4 = 19 - 3 = 16$

Average waiting time

$$= \frac{24 + 13 + 24 + 16}{4} = \frac{77}{4} = \textbf{19.24 milliseconds}$$

2.2. Multiprocessor Scheduling

A multiprocessor system consists of several processors which share memory. In the multiprocessor, there is more than one processor in the system. The reason we use multiprocessor is that sometimes load on the processor is very high but input output on other function is not required. In the multiprocessor system all the processors operate under the single operating system.

In this, the user does not know in which processor rheir process work. A process is divided in to several small processes and they work independently on the different processors.

In the multiprocessor scheduling, there are multiple CPU's which share the load so that various processes run simultaneously. in general as compared to single processor scheduling. In the multiprocessor scheduling, there are many processors and they are identical and we can run any process at any time.

The multiple CPU's in the system are in the close communication which shases a common bus memory and other periphesal devices. These systems are mainly used in satellite, weather forecasting etc.

Approaches to multiprocessor scheduling : There are two types.

1. Asymmetric multiprocessing : One approach to CPU scheduling in a multiprocessor system has all scheduling decisions, I/O processing and other system activities handled by a single processor the master server. The other processor execute only user code. This asymmetric multiprocessing is simple because only one processor accesses the system data structures, reducing the need for data sharing.

2. Symmetric multiprocessing : In symmetric multiprocessing each processor is self scheduling. All processes may be in common ready queue, or each processor may have its own private queue of ready processor.

Process synchronization

On the basis of syncronization, processes are categorized as on of the following two types.

Independent process : Execution of one process does not affect the execution of other processes.

Cooperative process : Execution of one process affects the execution of other processes.

Process synchronization problem arises in the case of co-operative process also because resources are shared in co-operative processes.

Race condition : When more than one processes are executing the same code or accessing the same memory of any shared variable, in that condition there is a possibility that the output or the value of the shared variable is wrong so for that all the processes doing the race to say that my output is correct this condition known as a race condition.

Race conditions in critical section can be avoided if the critical section is treated as an atomic instruction.

Critical section problem : Critical section is a code segment that can be accessed by only one process at a time. Critical section contains shared variable which need to be synchronized to maintain consistency of data variables.

```
do {
        entry section
            critical section
        exit section
            Remainder section
}
While (true);
```

Any solution to the critical section problem must satisfy three requirements.

1. Mutual exclusion : If a process is executing in its critical section, then no other process is allowed to execute in critical section.

2. Progress : If no process is executing in the critical section and other processes are waiting outside the critical section, then only those processes that are not executing in their remainder section can participate in deciding which will enter in the critical section next and selection can not postponed indefinitely.

3. Bounded Waiting : A bound must exist on the number of times that other processes are allowed to enter their critical sections after a process has made a request to enter its critical section.

SUMMARY

- Process management involves various tasks like creation, scheduling, termination of process and a dead lock. processes is a program that is under execution, which is an important part of modern day operating system.
- Process control block (PCB) is a data structure used by computer operating system to store all the information about a process.
- Schedular 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.
- Interprocess communication is the mechanism provided by the operating system that allows processes to communicate with each other.
- First come first served scheduling is the simplest scheduling method. CPU allocated to the process in the order of arrival. The process that requests the CPU first is allocated the CPU first.
- In shortest job first scheduling,the job having shortest or less burst time will get CPU first.
- In priority scheduling CPU allocated to the highest priority of the process from ready queue. Each process has a priority number.
- RR scheduling is similar to FCFS but preemption is added to switch between processes. A small unit of time, called a time quantum or time slice is defined.
- In multiprocessor scheduling, there are multiple CPU's which share the load so that various processes run simultaneously.

EXERCISE

1. Draw the process state diagram and explain the state.
2. Explain the need for process control block.
3. What is schedular and also explain its types.
4. Explain the operation on processes.
5. What is difference between preemptive and non preemptive scheduling?
6. Discuss about multiprocessor scheduling in brief.
7. Discuss the CPU secheduling criteria.

8. Draw gantt chart and calculate average turn around time and average waiting time for the following algorithm.

(a) FCFS (b) SJF (c) Priority (d) RR (Quantum time = 2)

Process	Burst time	Arrival time	Priority
A	3	0	1
B	5	1	2
C	2	2	3
D	5	3	2

9. Explain the interprocess communication.

10. Discuss the context switch.

3 DEADLOCKS

Deadlock, Conditions for Dead lock, Methods for handling deadlocks, Dead Prevention, Deadlock Avoidance, Deadlock detection, Recovery from deadlock.

3.1. Introduction

In a multiprogramming environment, several processes may complete for a finite number of resources. A process request resources and if the resources are not available at that time,the process enters a waiting state. Sometimes, a waiting process is never again able to change state, because the resources it has requested are held by other waiting process. This situation is called a deadlock.

A deadlock is situation where a group of processes are permanently blocked as a result of each process having acquired a subset of the resource needed for its completion and waiting for release of the remaining.

Sequence of deadlock :

1. Request : If the request can not be granted immediately (for example, if the resource is being used by another process) then the requesting process must wait untill it can acquire the resource.

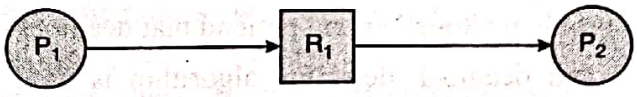


Fig. 3.1

Process P_1 request to resource R_1 and resource R_1 is being used by process P_2 so request of process P_1 can not be granted immediately.

2. Use : The process can operate on the resource (for example, if the resource is a printer, the process can print on the printer)

3. Release : After completion of program process releases the resource.

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

1. Mutual exclusion : At least one resource must be held in a nonsharable mode, that is only one process at a time can use the resource.

Deadlock will occur when many processes want to use nonsharable resource but only one process may use a nonsharable resource at a time.

If there exists no nonsharable resource, then deadlock will never occur.

Printer is an example of a nonsharable resource that can be used by only one process at a time.

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

3. No preemption : Resources can not be preempted that is, a resource can be released only voluntarily (not forcefully) by the process holding it, after that process has completed its task.

4. Circular wait : A set $\{P_0, P_1 \ldots P_n\}$ of waiting process must exist such that P_0 is waiting for a resource held by P_1, P_1 is waiting for a resource held by P_2 ....., P_{n-1} is waiting for a resource held by P_n, and P_n is waiting for a resource held by P_0.

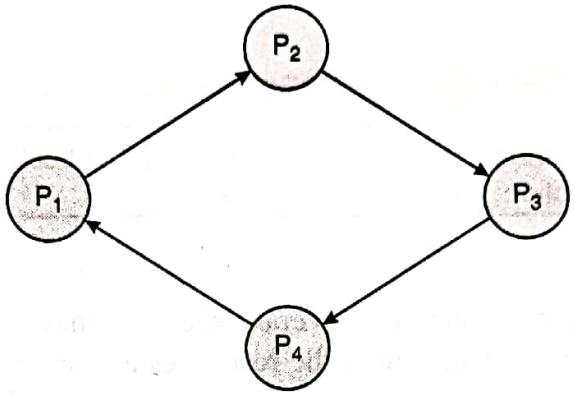


Fig. 3.2 : Circular wait

Methods for handling deadlocks

Generally speaking, we can deal with the deadlock problem in one of three ways.

- We can use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.
- We can allow the system to enter a deadlock, detect it and recover.
- We can ignore the problem altogether and pretend that deadlocks never occur in the system.

Deadlock prevention and deadlock detection algorithm is used for ignoring the deadlock. Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions cannot hold. These methods prevent deadlocks by constraining how requests for resources can be made.

Deadlock avoidance require that the operating system be given in advance, additional information concerning which resources, a process will request and use during its life time. If a system does not employ either a deadlock prevention or a deadlock avoidance algorithm, then a deadlock situation may occur.

3.2. Deadlock Prevention

In this method, the system will prevent any deadlock condition to happen. The system will make sure that at least one of the four conditions of the deadlock will be violated. Since we are preventing any one of four conditions to happen by applying some techniques. These techniques can be very costly so, you should apply deadlock prevention in only that situation which has a drastic change in the system if deadlock happens.

Let's see how we can avoid the four conditions of deadlock using the deadlock prevention technique.

Mutual exclusion : Mutual exclusion from the resource point of view is the fact that a resource can never be used by more than one process simultaneously which is fair enough but that is the main reason hebind the deadlock. If a resource could have been used by more than one process at the same time then the process would have never been waiting for any reasource. Read only files are a good example of a sharable resource. If several process attempt to open a read only file at the same time, they can be granted simultaneous access the file. A process never needs to wait for a sharable resource.

Hold and wait : Hold and wait arises when a process holds some resources and is waiting for some other resources that are being held by some other waiting process. To avoid this, the process can acquire all the resources that it needs, before starting its execution and after that, it starts its execution. In this way the process need not wait for some resources during its execution.

No preemption : This is a technique in which a process can't forcefully take the resource of other processes. But it we found some resource due to which, deadlock is happening in the system, then we can forcefully preempt that resource from the process that is holding that resource. By doing this, we can remove the deadlock.

- In general, sequential I/O devices can not be preempted.
- Preemption is possible for certain types of resources such as CPU and main memory.

Circular wait : Circular wait is a condition in which the first process is waiting for the resource held by the second process, the second process is waiting for the resource held by the third process and so on. Every process is waiting for each other to release the resource. This is called a circular wait.

To avoid this, we can list the number of resources required by a process and we assign some number or priority to each resource (in our case, we are using R_1, R_2, R_3 and so on). Now the process will take the resources in the ascending order. For example, if the process P_1 and P_2, requires resource R_1 and R_2, then initially, both the process will demand the resource R_1 and only one of them will get resource R_1 at that time and the other process have to wait for its turn. So in this way both the process will not be waiting for each other. One of them will be executing and the other will wait for its turn. So there is no circular wait here.

Deadlock Avoidance

Different deadlock prevention approach put different type of restrictions on the processes and resources because of which system becomes slow and resource utilization and reduced system through put.

To avoiding deadlocks we require additional information about how resources are to be requested. Which resources a process will request and use during its lifetime maximum number resources of each type that it may need.

With this additional knowledge, the operating system can decide for each request whether process should wait or not.

(a) Safe state : A state is safe if the system can allocate resource to each process in some order and still avoid a deadlock. A system is in a safe state only if there exists a safe sequence. A sequence of processes $< P_1, P_2, \ldots P_n >$ is a safe sequence for the current allocation state if, for each P_i, the resource

requests that P_i can still make can be satisfied by the currently available resources plus the resources held by all P_j with $j < i$. In this situation, if the resources that P_i needs are not immediately available, the P_i can wait untill all P_j have finished. When they have finished, P_i can obtain all of its needed resources, complete its designated task, return its allocated resources and terminate. When P_i terminates, P_{i+1} can obtain its needed resources and so on. If no such sequence exists, then the system state is said to be unsafe.

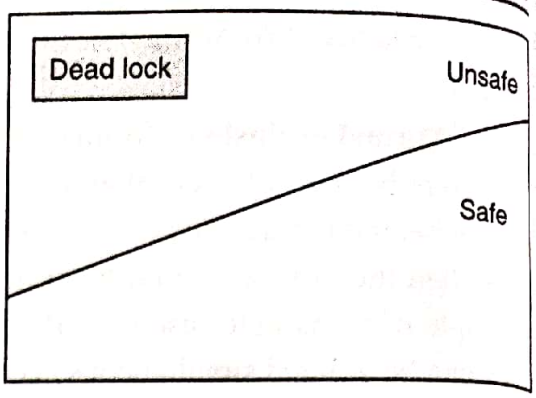


Fig. 3.3

- Safe state have never deadlock
- Unsafe state may be deadlock or not
- Deadlock is a subset of unsafe state.

For example : A system consists of three processes P_1, P_2 and P_3 and one resource R_1. Number of units for R_1 is 12.

- Process P_1 requires 10 R_1 resource.
- Process P_2 requires 4 R_1 resource.
- Process P_3 requires 9 R_1 resource.

At time t_0 operating system allocated resource R_1 to the all three processes as follows.

- Process P_1 is holding 5 resource of R_1
- Process P_2 is holding 2 resource of R_1
- Process P_3 is holding 2 resource of R_1

Total allocated resource is 9 and 3 resource R_1 is free.

Processes	Maximum Request (R_1)	Current allocation (R_1)	Available resource (R_1)
P_1	10	5	3
P_2	4	2	
P_3	9	2	

At time t_0, the system is in a safe state. The safe sequence for this is $< P_2, P_1, P_3 >$. This sequence satisfies the safety condition. Safe sequence is calculated as follows.

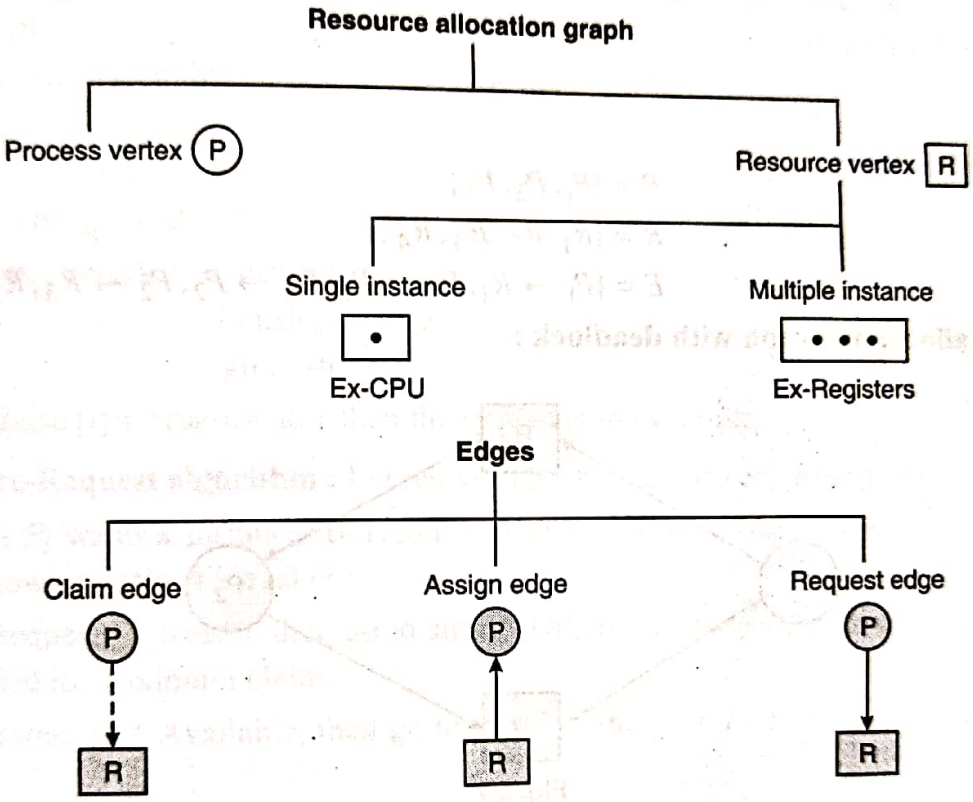
- Process P_2 can immediately be allocated all its resource (R_1) and then return to the system.
- Available resource (R_1) becomes 5 after returning process P_2.
- Process P_1 can get all its R_1 resource and return them to the system. Need of process P_1 is 5 (R_1 resource) and in system R_1 is available with 5 resources.
- Available resource (R_1) then becomes 10 resources.
- Finally process P_3 could get all its R_1 resource and return them to the system.
- At last, system have all 12 R_1 resources available.
- A system may go from safe state to the unsafe state.

Deadlock avoidance has the advantage that it is not necessary to preempt and roll back processes as in deadlock detection. It is less restrictive than deadlock prevention.

(b) Resource allocation graph : The resoruce allocation graph is the pictorial representation of the state of a system. The resource allocation graph is the complete information about all the process which are holding some resoruces or waiting for some resources.

It also contains the information about all the instances of all the resources whether they are available or being used by the processes.

In resource allocation graph the process is represented by a circle while the resource is represented by a rectangle. Let's see the types of vertices and edges in detail.



Claim edge : Not currently requirement but may be need future or release any resources then use claim edge.

Resoruce allocation graph with claim edge :

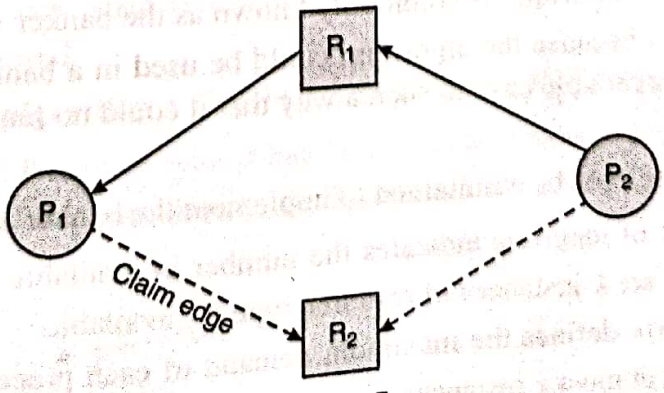


Fig. 3.5

Resource allocation graph simple :

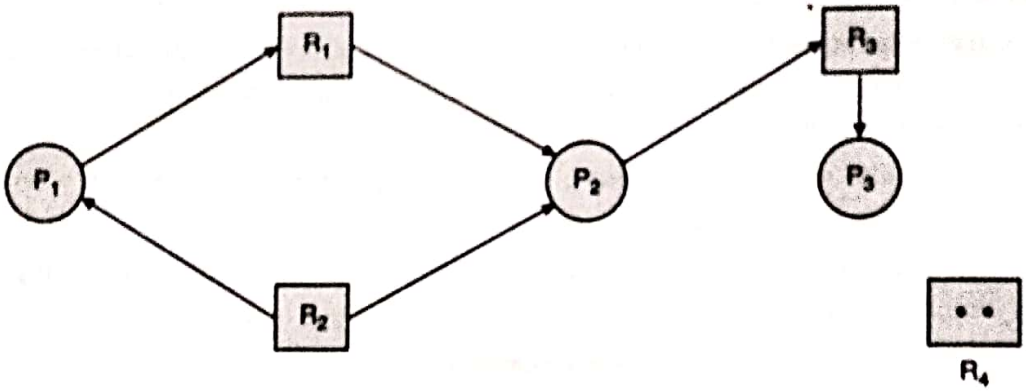


Fig. 3.6

Here $P = \{P_1, P_2, P_3\}$

$R = \{R_1, R_2, R_3, R_4\}$

$E = \{P_1 \rightarrow R_1, R_1 \rightarrow P_2, R_2 \rightarrow P_2, P_2 \rightarrow R_3, R_3 \rightarrow P_3\}$

Resource allocation graph with deadlock :

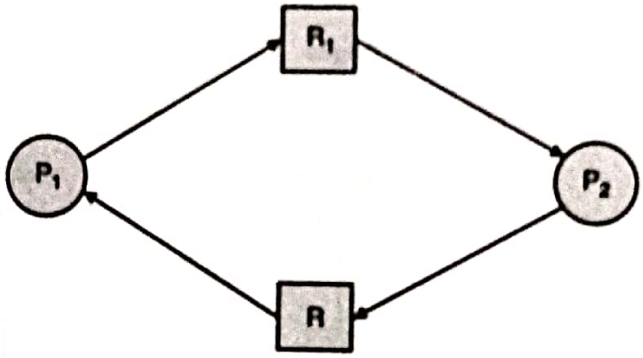


Fig. 3.7

(c) Banker's algorithm : The resource-allocation-graph algorithm is not applicable to a resource allocation system with multiple instances of each resource type. The deadlock avoidance algorithm that we describe next is applicable to such a system but is less effecient than the resource allocation graph scheme. This algorithm is commonly known as the banker's algorithm.

The name was chosen because the algorithm could be used in a banking system to ensure that the bank never allocated its available cash in such a way that it could no longer satisfy the needs of all its customers.

Several data structures must be maintained to implement the banker's algorithm.

Available : A vector of length m indicates the number of available resources of each type. If available $[j]$ equals k, there are k instances of resource type R_j available.

Max : An $n \times m$ matrix defines the maximum demand of each process. If max $[i][j]$ equals k, then process P_i may request at most k instances of resource type R_j.

Allocation : An $n \times m$ matrix defines the number of resources of each type currently allocated to each process if allocation $[i][j]$ equals k, then process P_i is currently allocated k instances of resource type R_j.

Need : An $n \times m$ matrix indicates the remaining resource need of each process. If need $[i][j]$ equals k then process P_i may need k more instances of resource type R_j to complete its task.

$$\text{Need}[i][j] = \text{Max}[i][j] - \text{Allocation}[i][j]$$

(a) Safety algorithm : Safety algorithm is used to find the state of the system. That is, system may be in safe state or unsafe state. Method for this is as follows.

Step 1 : Let work and finish be vector of length m and n respectively initialise

$$\text{Work} = \text{Available and finish}[i] = \text{False for } i = 1,2,3,4 \ldots\ldots n.$$

Step 2 : Find and i such that both

(a) Finish $[i]$ = False

(b) Need $[i] \leq$ Work

if no such i exist. go to step 4.

Step 3 : Work = Work + Allocation i

Finish $[i]$ = true

go to step 2

Step 4 : If finish $[i]$ = True for all i, then the system is in safe state.

(b) Resource-Request algorithm : Let request i be the request vector for process P_i. If request $i[j] = k$ then process P_i wants k instances of resource type R_j. When a request for resource is made by process P_i, the following actions are taken.

Step 1 : If request $i \leq$ Need i, then go to step 2. Otherwise, raise an error conditions, since the process has exceeded its maximum claim.

Step 2 : If request $i \leq$ Available, then go to step 3. Otherwise P_i must wait, since the resource are not available.

Step 3 : Available = Available − Request $_i$

$$\text{Allocation}_i = \text{Allocation}_i + \text{Request}_i$$

$$\text{Need}_i = \text{Need}_i - \text{Request}_i$$

If the resulting resource allocation state is safe, the transaction is completed and process P_i is allocated its resources. If the new state is unsafe, then P_i must wait for the request i and the old resource allocation state is restored.

Example of Banker's algorithm : Consider a system with five process P_0, P_1, P_2, P_3 and P_4 and three resource A, B and C. Resource type A has 10 instances, resource type B has 5 intances, and resource type C has 7 instances. Suppose that at time T_0 the following snapshot of the system has been taken.

Process	Allocation			Max			Available		
	A	*B*	*C*	*A*	*B*	*C*	*A*	*B*	*C*
P_0	0	1	0	7	5	3	3	3	2
P_1	2	0	0	3	2	2			
P_2	3	0	2	9	0	2			
P_3	2	1	1	2	2	2			
P_4	0	0	2	4	3	3			

Solution : Need = Max – Allocation

Need

	A	*B*	*C*
P_0	7	4	3
P_1	1	2	2
P_2	6	0	0
P_3	0	1	1
P_4	4	3	1

Need = max – Allocation

A B C A B C A B C

$P_0 = 753 - 010 = 743$

$P_1 = 322 - 200 = 122$

$P_2 = 902 - 302 = 600$

$P_3 = 222 - 211 = 011$

$P_4 = 433 - 002 = 431$

Safe sequence : Safe sequence is calculated as follows.

1. Need of each process is compared with available. If Need $i \leq$ available i, then the resour are allocated to that process and process will release the resource.
2. If need is greater than available, next process need is taken for comparison.
3. In the above example, need of process P_0 is (7, 4, 3) and available is (3, 3, 2).

$$\text{Need} \geq \text{available} \rightarrow \text{False}$$

So, system will move for next process.

4. Need for process P_1 is (1, 2, 2) and available (3, 3, 2) so

$$\text{need} \leq (3, 3, 2) = \text{True.}$$

then

$$\text{fiish}[i] = \text{True.}$$

Request P_1 is granted and processes P_1 is release the resource to the system.

$$\text{Work} = \text{Work} + \text{Allocation}$$

$$\text{Work} = (3,3,2) + (2,0,0)$$

$$= 5,3,2$$

In safety algorithm initialise Work = Available

5. Next process P_2 need (6,0,0) is compared with new available (5,3,2)

$$\text{Need} > \text{Available} = \text{False}$$

$$(6,0,0) > (5,3,2)$$

6. Process P_3 need (0, 1, 1) is compared with available (5,3,2).

$$\text{Need} < \text{available}$$

$$(0,1,1) < (5,3,2) = \text{True}$$

$$\text{Work} = \text{Work} + \text{Allocation} \qquad \text{(Work = Available)}$$

OR

$$\text{Available} = \text{Available} + \text{Allocation}$$

$$= 532 + 211$$

$$= 743 \text{ (New available)}$$

7. Then process P_4 need (4, 3, 1) is compared with available (7 4 3)

$$\text{Need} < \text{Available}$$

$$(431) < (743) = \text{True}$$

$$\text{Available} = \text{Available} + \text{Allocation}$$

$$= 743 + 002$$

$$= 745 \text{ (New available)}$$

8. One cycle is completed. Again system takes all remaining process in sequence. So process P_0 need (7 4 3) is compared with new available (7, 4, 5)

$$\text{Need} < \text{Available} = \text{True}$$

$$(7\,4\,3) < (7\,4\,5) = \text{True}$$

$$\text{Available} = \text{Available} + \text{Allocation.}$$

$$= 745 + 010$$

$$= 755 \text{ (New available)}$$

9. Process P_2 need is (600) is compared with new available (7 5 5).

$$\text{Need} < \text{Available} = \text{True}$$

$$(6\,0\,0) < (7\,5\,5) = \text{True}$$

$$\text{Available} = \text{Available} + \text{allocation}$$

$$= 755 + 302$$

$$= (10,5,7) \qquad \text{(New available)}$$

So safe sequence is $< P_1 P_3 P_4 P_0 P_2 >$

Example 2. Consider the following snapshot of a system :

Process	Allocation				Max				Available			
	A	*B*	*C*	*D*	*A*	*B*	*C*	*D*	*A*	*B*	*C*	*D*
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	5	3	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

Answer the following questions using the banker's algorithm.

(a) What is the content of the matrix need?

(b) Is the system in a safe state?

Solution :

(a) Need matrix

Need = Max – Allocation

	A	*B*	*C*	*D*
P_0	0	0	0	0
P_1	0	5	3	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

(b) Safe state :

1. Need of process P_0 is (0,0,0,0) and available (1, 5, 2, 0)

Need < Available = True

(0, 0, 0, 0) < (1, 5, 2, 0) = True

Then finish [*i*] = True

Request P_0 is granted and process P_0 is release the resource to the system.

Work = Work + Allocation

OR

Available = Available + Allocation

= (1,5,2,0) + (0,0,1,2)

= (1,5,3,2) New available.

2. Next process P_1 need (0,5,3,0) is compared with new available (1, 5, 3, 2)

Need < available = True

(0, 5, 3, 0) < (1, 5 3 2) = True

Available = Available + Allocation
= 1 5 3 2 + 1 0 0 0
= 2 5 3 2 (New available)

3. Process P_2 need (1 0 0 2) is compared with new available (2 5 3 2)
Need < available
(1 0 0 2) < (2 5 3 2) = True
Available = Available + Allocation
= 2 5 32 + 1 3 5 4
= (3 8 8 6) New available

4. Process P_3 need (0 0 2 0) is compared new available (3 8 8 6)
Need < available
(0 0 2 0) < (3 8 8 6) = True
Available = Available + Allocation
= 3 8 8 6 + 0 6 3 2
= (3 14 11 8) New available

5. Then process P_4 need (0 6 4 2) is compared new available (3, 14, 11, 8)
Need < Available
(0 6 4 2) < (3, 14, 11, 8) = True
Available = Available + Allocation
= (3, 14, 11, 8) + (0, 0, 1, 4)
= (3, 14, 12, 12) (New available)

So system is in a safe state, safe sequence is < $P_0P_1P_2P_3P_4$ >

3.4. deadlock Detection

A deadlock occurence can be detected by the resource schedular. A resource schedular helps operating system to keep track of all the resources which are allocated to different processes.

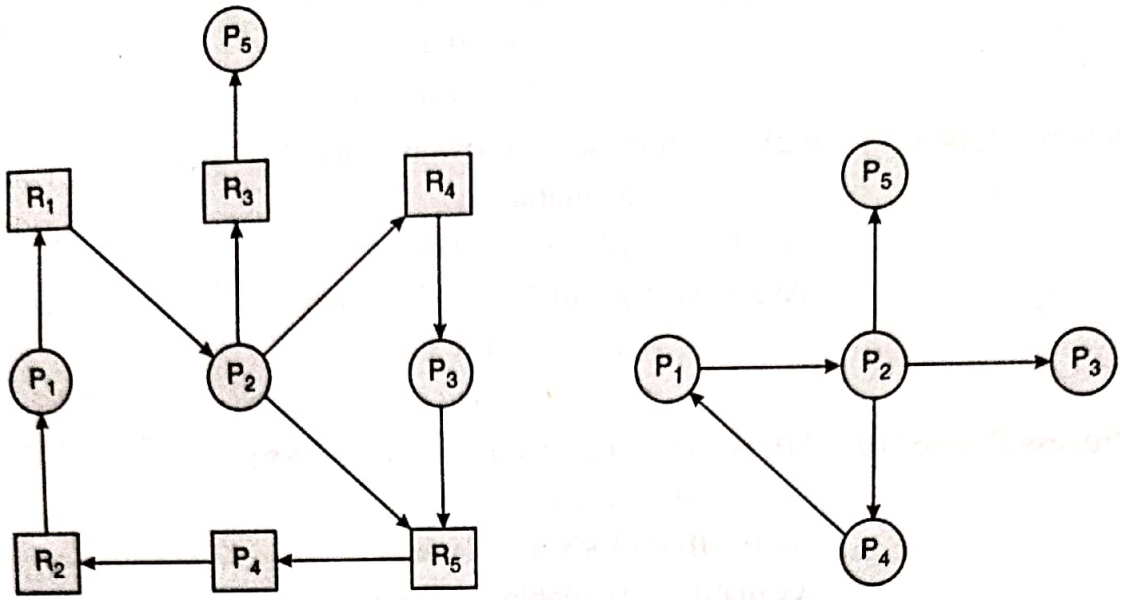
If the system is not using any deadlock avoidance and deadlock prevention, then a deadlock situation may occur. Deadlock detection approach do not limit resource access or restrict process actions. With deadlock detection request resources are granted to processes whenever possible. There are following deadlock detection algorithm.

1. Single instance of each resource type.
2. Several instance of a resource type.

1. Single instance of each resource type : If all resource have only a single instance, then we can define a deadlock detection algorithm that uses a variant of the resource allocation graph, called a wait for graph.

- This graph is obtained from the resource allocation graph by removing the resource nodes and collapsing the appropriate edges.
- More precisely, an edge from P_i to P_j in a wait for graph implies that process P_i is waiting for process P_j to release a resource that P_i needs.

(a) For example in following fig. a resource allocation graph and the corresponding wait for graph are presented.



(a) Resource allocation graph (b) Wait for graph

Fig. 3.5

(b) In the figure of resource allocation graph process P_1 is holding resource R_2 and requesting resource R_1. So this is shown in the following figure. In this process P_1 is request edge with proces P_2. It is shown in figure.

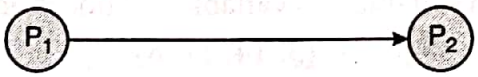


Fig. 3.6

2. Several instances of a resource type : In deadlock detection approaches, the resource allocater simply grants each request for an available resource. For checking for deadlock of the system, the algorithm is as follows.

(a) Unmark all active processes from allocation. Max and available in accordance with the system state.

(b) Find an unmarked process *i* such that

$$\text{Max}_i \leq \text{Available}$$

if found, mark process *i*, update available

$$\text{Available} = \text{Available} + \text{Allocation}$$

and repeat this step.

If no process is found, then go to next step.

(c) If all processes are marked, the system is not deadlocked. Otherwise system is in deadlock state and the set of unmarked processes is deadlocked.

Deadlock Recovery

Once deadlock has been detected, some strategy is needed for recovery. Following are the solutions to recover the system from deadlock.

1. Process termination
2. Resource preemption

1. Process termination :

(A) All deadlocked processes are aborted. Most of the operating system use this type of solution.

(i) Abort all deadlocked processes. This method clearly will break the deadlock cycle, but at great expense.

(ii) Abort one process at a time untill the deadlock cycle is eliminated.

(B) Many factors may affect which process is chosen, including :

(a) Least amount of processor time consumed so far.

(b) Least amount of output produced so far.

(c) Lowest priority.

(d) Most estimated time remaining.

(e) Least total resources allocated so far.

2. Resource preemption :

(a) If preemption is required to deal with deadlocks then three issues need to be addressed.

(b) Which resources and which processes are to be preempted?

(c) Rollback : Backup each deadlocked process to some previously defined check point and restart all processes. This requires roll back and restart mechanisms are built in to the system.

SUMMARY

- A deadlock is a situation where a group of processes are permanently blocked as a result of each process having acquired a subset of the resource needed for its completion and waiting for release of the remaining.
- In mutual exclusion at least one resource must be held in a nonsharable mode, that is, only one process at a time can use the resource.
- A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.
- Resource can not be preempted, that is a resource can be relesaed only voluntarily by the process holding it.
- Circular wait is a condition in which the first process is waiting for the resource held by the second process. The second process is waiting for the rsource held by the third process and so on. At last, last process is waiting for the resource held by the first process.
- Avoiding deadlocks we require additional information about how resources are to be requested, which resources a process will requrest and use during its lifetime.
- A state is safe if the system can allocate resource to each process in some order and still avoid a deadlock.

- The resource allocation graph is the pictorial representation of a system. The resource allocation graph is the complete information about all the process which are holding some resource or waiting for some resources.
- A deadlock occurence can be detected by the resource schedular. A resource schedular helps operating system to keep track of all the resources which are allocated to different processes.

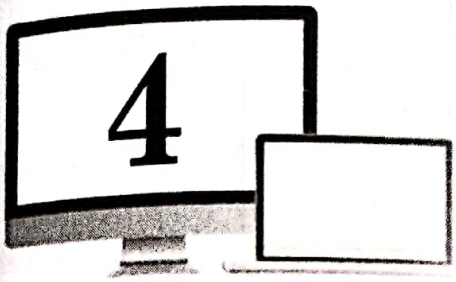
EXERCISE

1. Explain the different conditions of deadlock.
2. How the recovery from deadlock is done using combined approach?
3. Write down the methods for deadlock prevention.
4. Describe the necessary condition for deadlock to occur.
5. Write and explain Banker's algorithm.
6. Consider following snapshot.

Process	Allocation		Max		Available	
	R_1	R_2	R_1	R_2	R_1	R_2
P_1	1	2	4	2	1	1
P_2	0	1	1	2		
P_3	1	0	1	3		
P_4	2	0	3	2		

(a) What is the need matrix?

(b) Is the system in a safe state?



4

MEMORY MANAGEMENT FUNCTION

Definition – Logical and Physical address Space, Swapping, Memory allocation, Contiguous Memory allocation, Fixed and variable partition, internal and External fragmentation and Compaction, Paging – Principle of operation, Page allocation, Hardware support for paging, Protection and sharing, Disadvantages of paging, Segmentation, Virtual Memory.

4.1. Introduction

In operating systems, memory management is the function responsible for managing the computer's primary memory.

The memory management function keeps track of the status of each memory location, either allocated or free. It determines how memory is allocated among competing processes, deciding which gets memory, when they receive it, and how much they are allowed. When memory is allocated it determines which memory locations will be assigned. It tracks when memory is freed or unallocated and updates the status.

Logical and physical address space :

(A) Logical address : Logical address is generated by CPU while a program is running. The logical address is virtual address as it does not exist physically, therefore, it is also known as virtual address. This address is used as a reference to access the physical memory location by CPU.

Logical address space : The term logical address space is used for the set of all logical addresses generated by a program perspective.

The hardware device called memory management unit is used for mapping logical address to its corresponding physical address.

(B) Physical address : Physical address identifies data in a memory. The user never directly deals with the physical address but can access by its corresponding logical address. The user program generates the logical address and thinks that the program is running in this logical address but the program needs physical memory for its execution, therefore, the logical address must be mapped to the physical address by memory management unit before they are used.

Physical address space : The term physical address space is used for all physical address corresponding to the logical address in a logical address space.

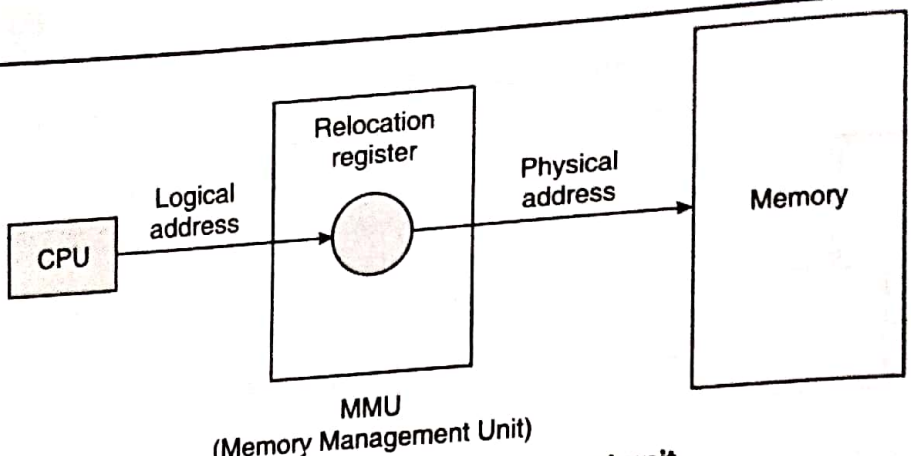


Fig. 4.1 : Memory management unit

Differences between logical and physical address :

1. The basic difference between logical and physical address is that logical address is generated by CPU in perspective of a program whereas the physical address is a location that exists in the memory unit.
2. Logical address space is the set of all logical address generated by CPU for a program whereas the set of all physical address mapped to corresponding logical address mapped to corresponding logical addresses is called physical address space.
3. The logical address does not exist physically in the memory whereas physical address is a location in the memory that can be accessed physically.
4. Indentical logical addresses are generated by compile-time and load time address binding methods whereas they differ from each other in run time address binding method.
5. The logical address is generated by the CPU while the program is running whereas the physical address is computed by the memory management unit (MMU).

4.2. Swapping

A Process must be in memory to be executed. A process can be swapped temporarily out of memory to a backing store and then brought back into memory for continued execution. For example, assume a multiprogramming environment with a round-robin CPU-scheduling algorithm.When a quantum time expires, the memory manager will start to swap out the process that just finished quantum time and to swap in another process into the memory space that has been freed. Once each process has received one time quantum, the memory manager is returned to first process for swap in (an additional time quantum).

A variant of this swapping policy is used for priority-based scheduling algorithms. If a higher-priority process arrives and wants service, the memory manager can swap out the lower-priority process and then load and execute

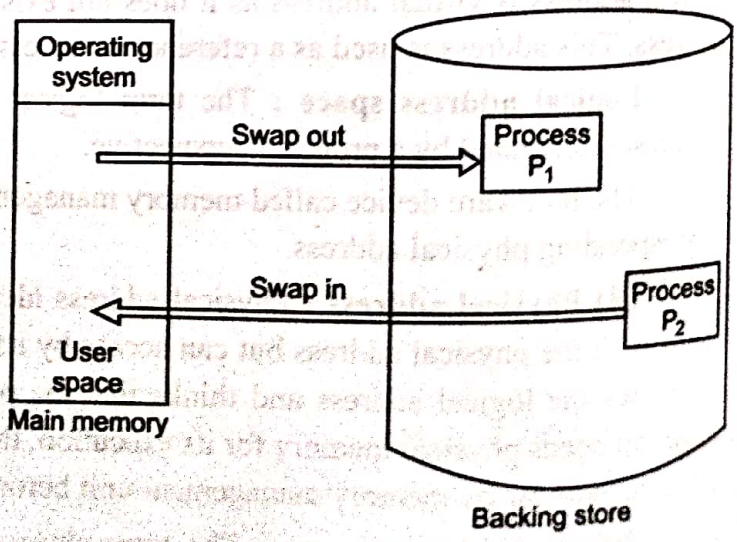


Fig. 4.2

the higher-priority process. When the higher-priority process finishes the lower-priority process can be swapped back in and continued. This variant of swapping is sometimes call rollout, roll in.

4.3. Memory Allocation

One of the simplest methods for allocating memory is to divide memory into several fixed sized partitions. Each partition may contain exactly on process. Thus the degree of multiprogramming is bound by the number of partitions. In this multiple partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes available for another process.

Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space.

Memory allocation is the process of reserving a partial or complete portion of computer memory for the execution of programs and processes. Memory allocation is achieved through a process known as memory management. Memory allocation has two types.

- **Static memory allocation :** The program is allocated memory at compile time.
- **Dynamic memory allocation :** The programs are allocated with memory at run time.

Contiguous memory allocation

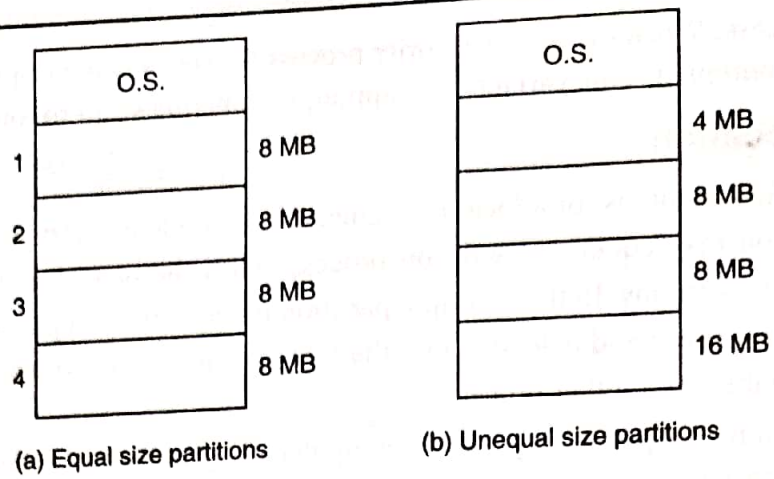
The memory is usually divided into two partitions. One for the resident operating system and for the user processes. We can place the operating system in either low memory or high memory. The major factor affecting this decision is the location of the interrupt vector. Since the interrupt vector is often in low memory programmers usually place the operating system in low memory as well.

We usually want several user processes to reside in memory at the same time. We therefore need to consider how to allocate available memory to the process that are in the queue waiting to be brought into memory. In this contiguous memory allocation, each process is contained in a single contiguous section in memory.

Fixed and variable partition

Fixed partitions : Fixed partitions which is known as static partitions. In contiguous memory allocation whenever processes are coming into the RAM how we are allocating them the space that is one of the method is fixed partitions or we can also say static partitions. The partitioning of memory in fixed partition is done before processing.

- No. of partitions are fixed. (No = number)
- Size of each partition may or may not same.
- If the partition is free, process is selected from the input queue and is loaded into the free partition of memory.
- When the process terminates, the memory partition becomes available for another process.
- Each partition contains exactly one process.
- Any process whose size is less than or equal to the partition size can be loaded into any available partition.



(a) Equal size partitions (b) Unequal size partitions

Fig. 4.3

Example :

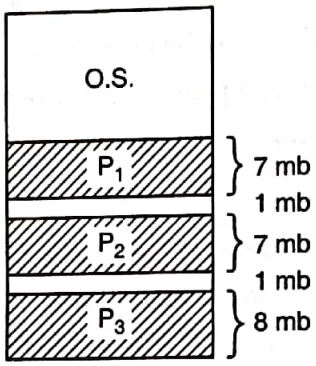
Process

$P_1 = 7$ mb

$P_2 = 7$ mb

$P_3 = 8$ mb

$P_4 = 2$ mb



Process P_1 and P_2 are 7 mb and process P_3 is 8 mb but each memory partition are 8 mb. In first and second partition 1 mb memory space left which we can not give to any other process. Memory which is left and can not use that memory for any other process then it is called internal fragmentation.

In memory 2 mb (1 mb partition 1 and 1 mb patition 2) memory space are free but we can not allocate memory for process P_4. It is called external fragmentation.

Variable partition : In variable partitions wherever the processes are come into the RAM only then we are allocating the space for processes. It is also known as dynamic partitions.

- No internal fragmentation
- No limitation on no. of processes.
- There is no limitation on process size

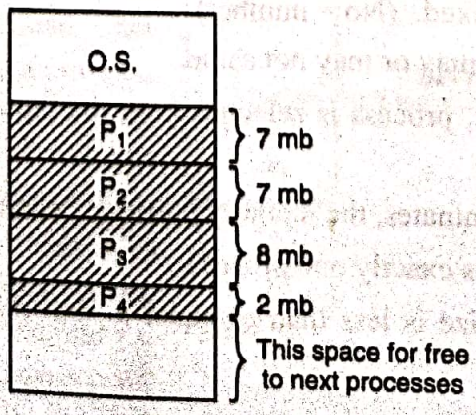
Example : Process

$P_1 = 7$ mb

$P_2 = 7$ mb

$P_3 = 8$ mb

$P_4 = 2$ mb



Size of process P_1 and P_2 are 7 mb and process P_1 and P_2 reserved only 7 mb of space in memory. Size of process P_3 is 8 mb so process P_3 reserved only 8 mb of space in memory and next process P_4 size is 2 mb so process P_4 reserved only 2 mb of space in memory. The remaining space of memory is free for the upcoming process.

4.3. Internal and External Fragmentation

Fragmentation : As processes are loaded and removed from memory the free memory space is broken into little pieces. It happens after sometimes that processes can not be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as fragmentation. Fragmentation is of two types.

1. Internal fragmentation : Memory block assigned to processes is bigger. Some portion of memory is left unused, as it can not be used by another process.

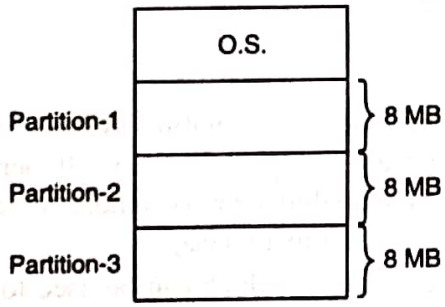
In internal fragmentation there is wasted space internal to a partition due to the fact that the block of data loaded is smaller than the partition and that wasted space is not being used.

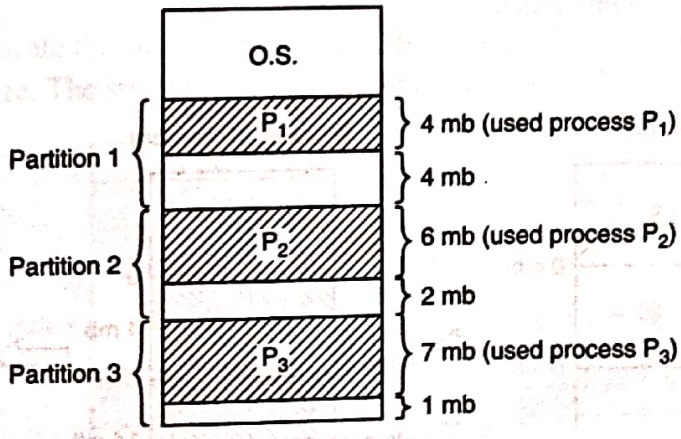
Example : **Process**

$P_1 = 4$ mb

$P_2 = 6$ mb

$P_3 = 7$ mb





Process P_1 is 4 mb and process P_2 is 6 mb and process P_3 is 7 mb. But each memory partition are 8 mb. In first partition 4 mb memory space left. In second partition 2 mb memory space left and in third partition 1 mb memory space left which we cannot allocate to any other process. Memory which is left and can not allocate that memory for any other process then it is called internal fragmentation.

2. External fragmentation : Total memory space is enough to satisfy a request or to reside a process in it, but is not contiguous, so it can not be used.

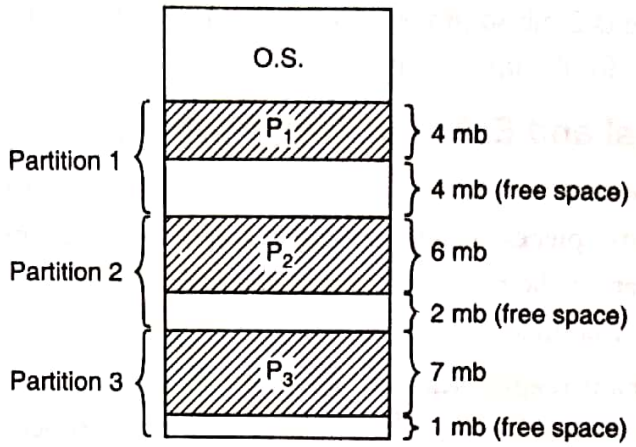
Example : **Process**

$P_1 = 4\,mb$

$P_2 = 6\,mb$

$P_3 = 7\,mb$

$P_4 = 6\,mb$



Total free memory space 7 mb is in main memory and process P_4 requires 6 mb of memory space but we can not allocate memory space in memory for process P_4 because the 7 mb of memory space that is free is not contiguous (4 mb partition 1, 2 mb partition 2 and 1 mb partition 3 have free memory space).

Compaction

In the fixed partition scheme, the operating system keeps a table indicating which parts of memory are available and which are occupied. Initially, all memory is available for user processes and is considered one large block of available memory, a hole. When a process arrives and needs memory, we search for a hole large enough for this process.

Compaction is one of the method which can be used to remove the external fragmentation. In compaction the operating system shifts the processes so that they are contiguous and all the free memory (holes) is together in one block.

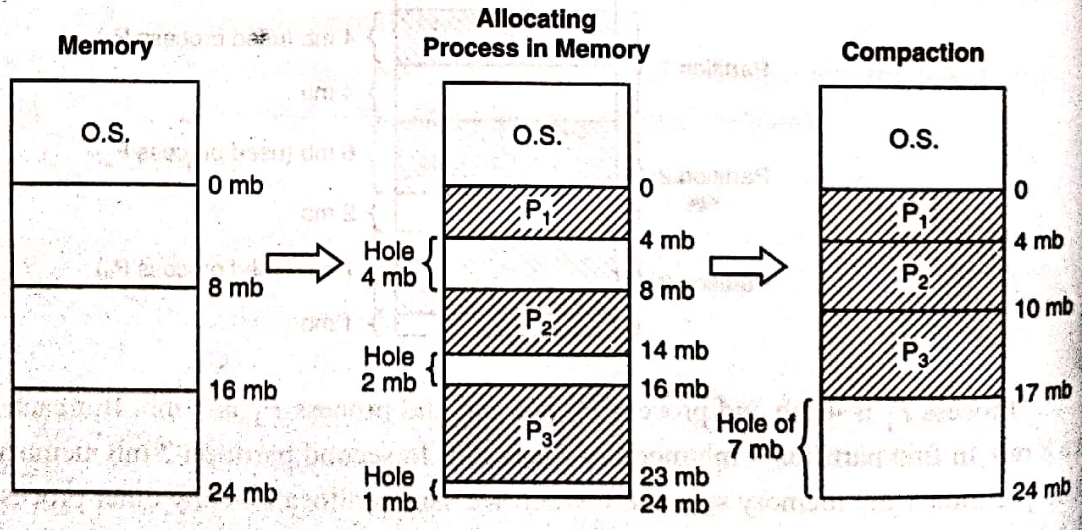
Example :

Process

$P_1 = 4\,mb$

$P_2 = 6\,mb$

$P_3 = 7\,mb$



Three holes of sizes 4 mb of first partition, 2 mb of second partition and 1 mb of third partition (total 7 mb) can be compacted in to hole of size 7 mb at last.

Allocation methods in contiguous memory allocation

Various allocation methods in contiguous memory allocation.

1. First fit : Allocate the first hole that is big enough. Searching can start either at the beginning of the set of holes or where the previous first fit search ended. We can stop searching as soon as we find a free hole that is large enough.

For example, suppose a process requests 12 KB of memory and the memory manager currently has a list of holes of 6 KB, 14 KB, 19 KB, 11 KB and 13 KB holes. First fit will allocate 12 KB of the 14 KB hole to the process.

Request process memory = 12 KB

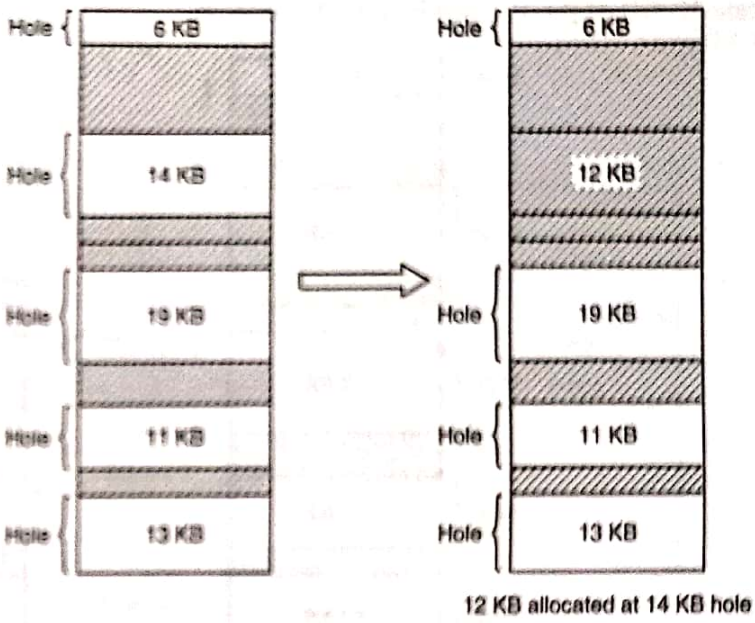


Fig. 4.4

2. Best fit : Allocate the smallest hole that is big enough. We must search the entire list, unless the list is ordered by size. The strategy produce the smallest leftover hole.

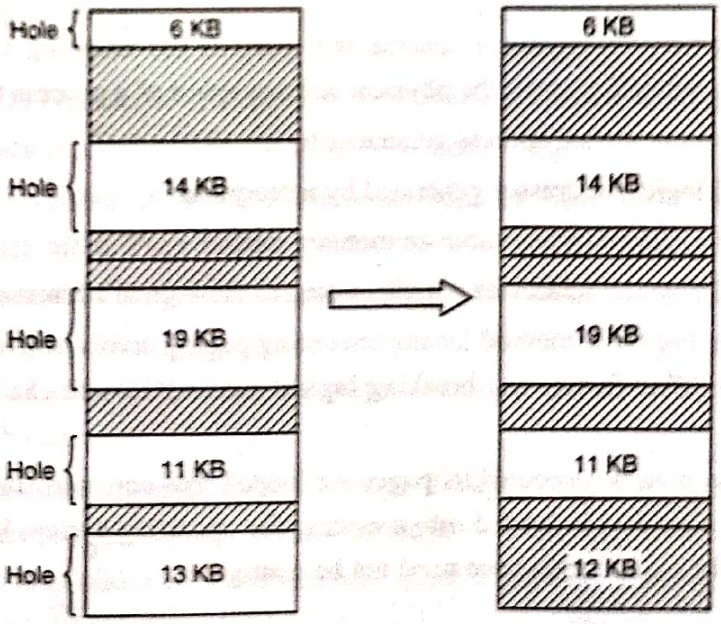


Fig. 4.5

For example : Suppose a process requests 12 KB of memory and the memory manager currently has a list of unallocated blocks of 6 KB, 14 KB, 19 KB, 11 KB and 13 KB blocks. The best fit strategy will allocate 12 KB of the 13 KB block to the process.

3. Worst fit : Allocate the largest hole. Again, we must search the entire list, unless it is sorted by size. This strategy produces the largest leftover hole, which may be more useful than the smaller leftover hole from best fit approach.

For Example : Suppose a process requests 12 KB of memory and the memory manager currently has a list of allocated blocks of 6 KB, 14 KB, 19 KB, 11 KB and 13 KB blocks. Worst fit will allocate 12 KB of the 19 KB block to the process.

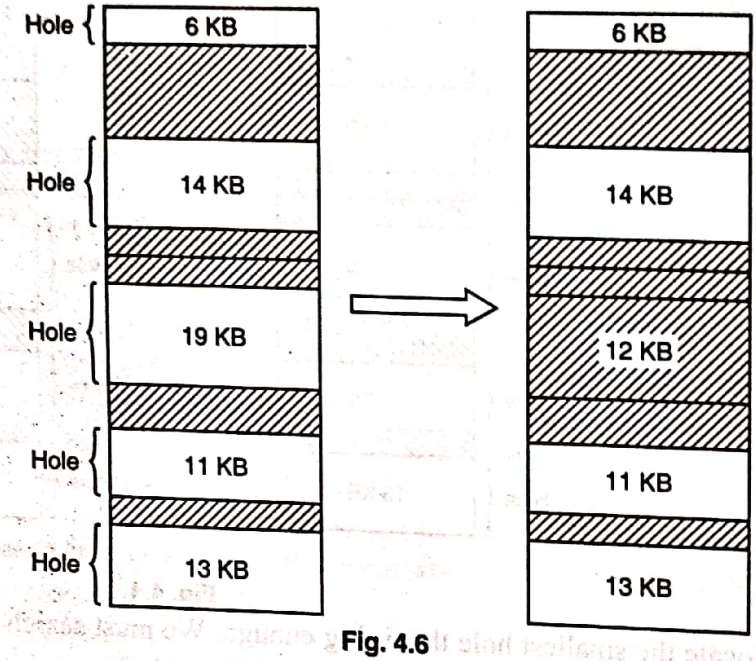


Fig. 4.6

4.4. Paging

Paging is a memory-management scheme that permits the mapping to logical memory and physical memory. This scheme permits the physical address space of a process to be non-contiguous.

- Logical address or virtual address generated by the CPU.
- The set of all logical addresses generated by a program.
- Physical address actually available on memory unit.
- The set of all physical addresses corresponding to the logical addresses.

Basic method : The basic method for implementing paging involves breaking physical memory into fixed size blocks called frames and breaking logical memory into blocks of the same size called pages.

When a process is to be executed,its pages are loaded into any available memory frames from backing store. When a process is loaded into memory, the operating system loads each page into an unused page frame. The page frames used need not be contiguous.

Operating system maintains the page table for each process. The page table shows the frame location for each page of the process.

Every address generated by the CPU is divided into two parts.

1. Page number (*P*) : Number of bits required to represent the pages in logical address space or page number.

2. Page offset (*d*) : Number of bits required to represent particular word in page or page size of logical address space or word number of a page or page offset. Physical address is divided into two parts.

(i) Frame number (*F*) : Number of bits required to represent the frame of physical address space or frame number.

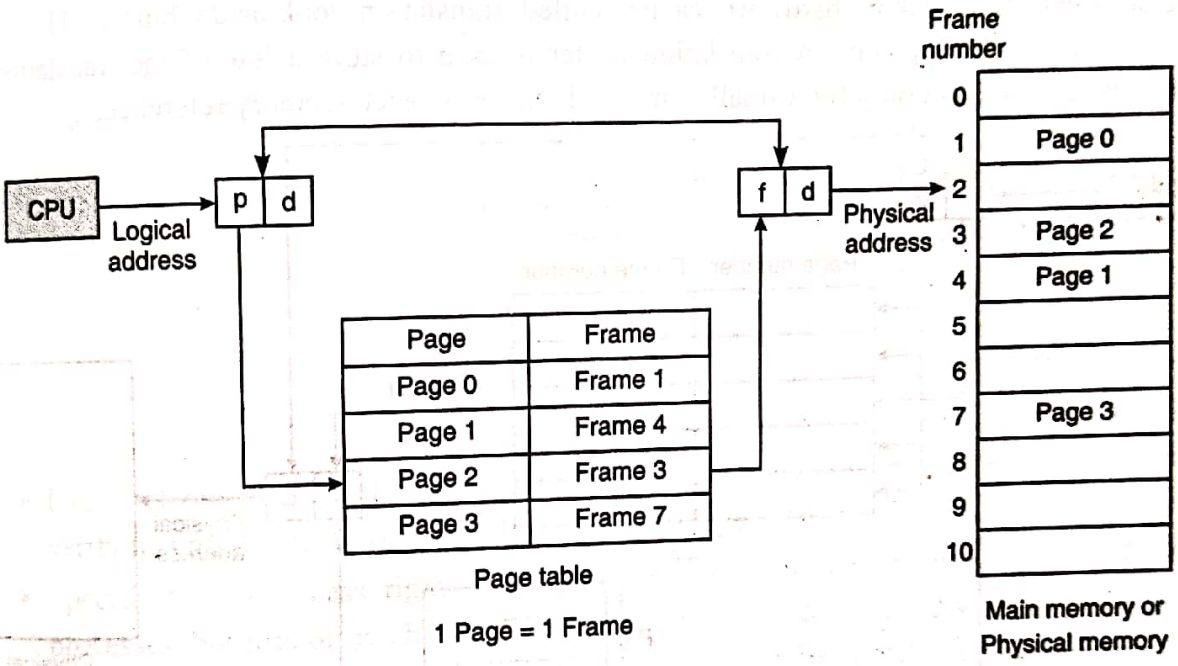


Fig. 4.7 : Paging model of logical and physical memory

(ii) Frame offset (*d*) : Number of bits required to represent particular word in a frame or frame size of physical address space or word number of a frame or frame offset.

Page table stores the number of the page frame allocated for each page. The page number is used as an index in to page table. Page table contains the base address of each page in physical memory. This base address is combined with the page offset to define the physical memory address that is sent ot the memory unit.

When a process execute thon CPU generates pages and that pages entered in page table. Page table allocate frame for each page.

In above figure CPU generates page 0, page 1, page 2, page 3 and that pages entered in page table. Page table allocate frame 1 for page 0, frame 4 for page 1, frame 3 for page 2 and frame 7 for page 3. These pages go into their allocated frame and take space in physical memory.

Hardware support for paging

A hardware mechanism implementation of the page table can be done in several ways. In the simplest case, the page table is implemented as a set of dedicated registers. These registers should be build with very high-speed logic to make the paging address translation efficient. Every access to memory must go through the paging map, so efficiency is a major consideration. The CPU dispatcher reloads these registers, just as it reloads the other register instruction to load or modify the page table registers are, of course, privileged, so that only the operating system can change the memory map.

Translation look-aside buffer : Problem with paging is that, extra memory references to access translation tables can slow programs down by a factor of two or three. Too many entries in translation tables to keep them all loaded in fast processor memory. The standard solution to this problem is to use a special, small fast lookup hardware cache, called translation look-aside buffer. The TLB is associative, highspeed memory. A translation buffer is used to store a few of the translation table entries. It is very fast, but only for a small number of entries on each memory reference.

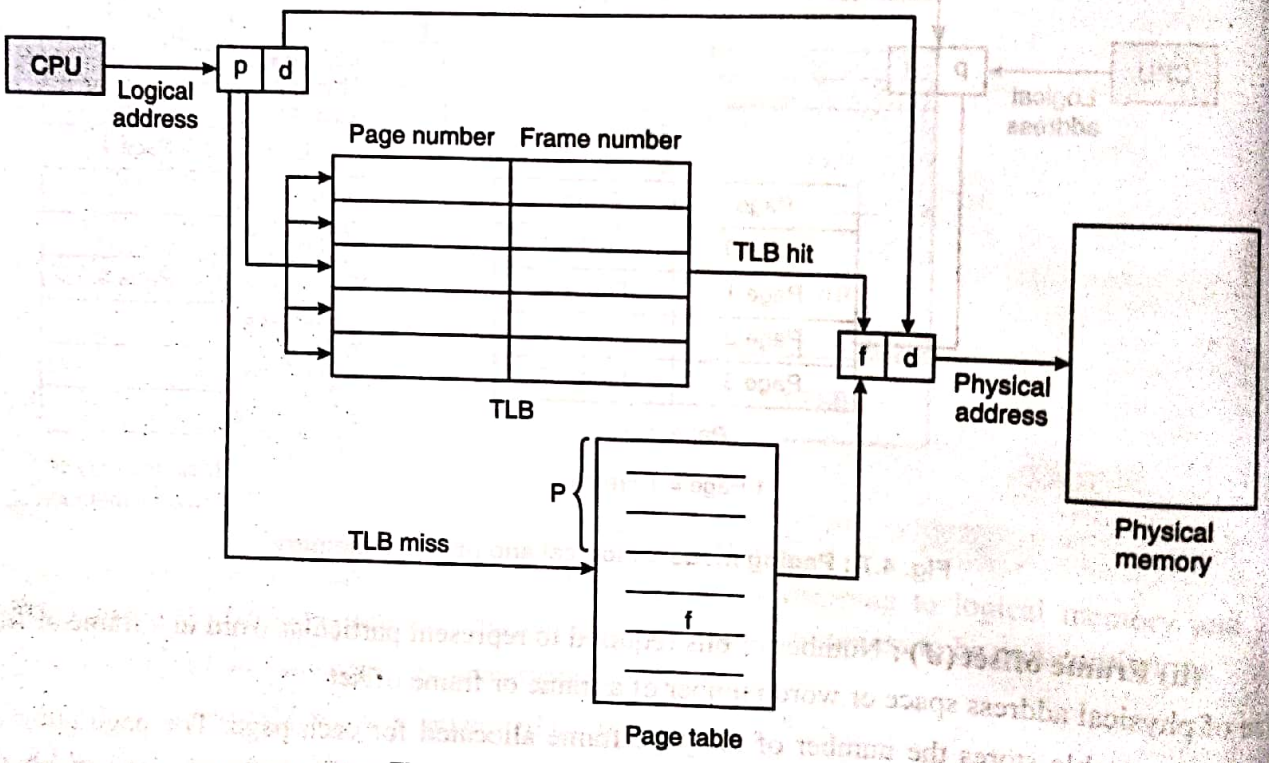


Fig. 4.8 : Paging hardware with TLB

- First ask TLB if it knows about the page. If so, the reference proceeds fast.
- If TLB has no information for page, must go through page and segment table to get information. Reference takes a long time, but gives the info for this page to TLB so it will know it for next reference.

The percentage of times that a particular page number in the TLB is called the hit ratio. The hit ratio is the ratio between accesses that find a match in the associative memory and those that do not. The greatest performance improvement is achieved when the associative memory is significantly faster than the normal page table lookup and the hit ratio is high.

Protection and sharing

- Protection bits are used with each page frame for protecting memory in paging. For example, protection bits (access bit) may allow read only, execute only or other restricted forms of access.

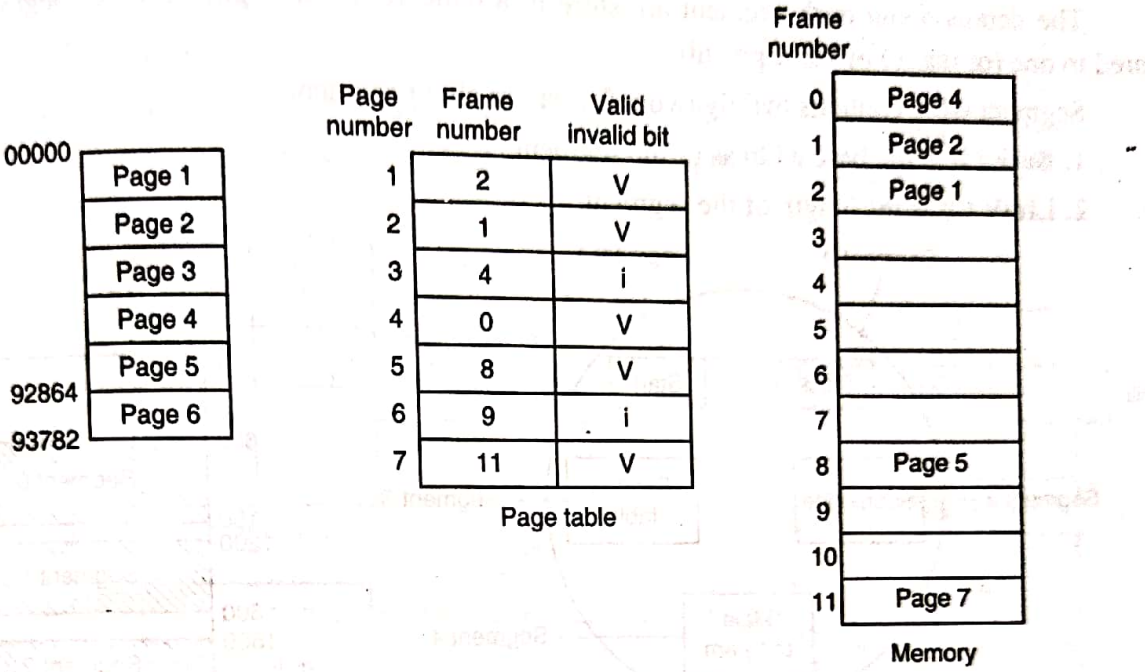


Fig. 4.9

- Every reference physical address is being computed, the protection bits can be checked to verify that no writes are being made to a read only page.
- Specification of access rights in paging systems is useful for pages shared by several processes, but it is of much less value inside the boundaries of a given address space. One more bit is generally attached to each entry in the page table, a valid invalid bit.
- When valid bit is set, then the page is in the process logical address space. Thus page is legal. If invalid bit is set, page is not in the process logical address space and illegal page. Illegal addresses are trapped by using the valid-invalid bit.
- A single physical copy of a shared page can be easily mapped into as many distinct address space as desired.

Disadvantages of paging

- May cause internal fragmentation.
- Complex memory management algorithm.
- Page table consume additional memory.
- Multi-level paging may lead to memory reference overhead.
- Page address mapping hardware usually increases the cost of the computer.

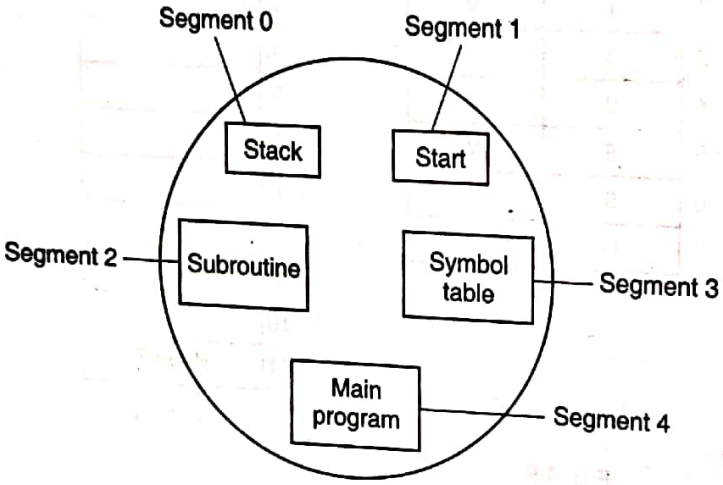
4.4. Segmentation

Segmentation is a memory management scheme. Segmentation divides a program into a numb of smaller blocks called segments. A logical address space is a collection of segments. Each segme has a name and length.

The details about each segment are store in a table called as segment table. Segment table stored in one (or many) of the segments.

Segment table contains mainly two information about segment.

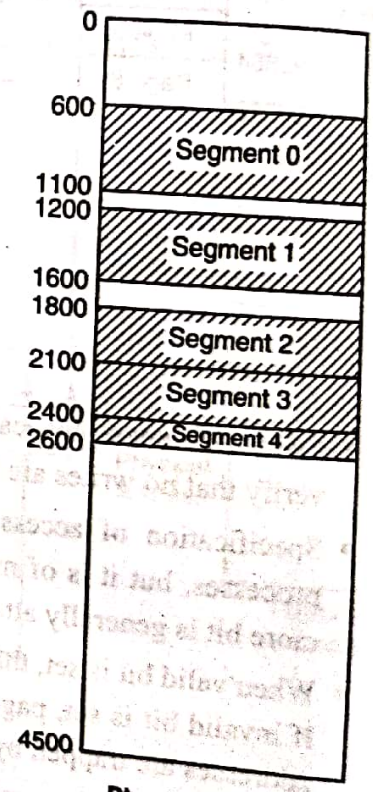
1. **Base :** It is the base address of the segment.
2. **Limit :** It is the length of the segment.



Logical address space

Segment Name	Limit	Base
Segment 0	500	600
Segment 1	400	1200
Segment 2	300	1800
Segment 3	300	2100
Segment 4	200	2400

Segment table



Physical memory

Fig. 4.10

The segment base contains the starting physical address where the segment resides in memory. Where as the segment limit specifies the length of the segment.

Segmentation is similar to dynamic partitioning because of the unequal size segments. Segmentation removes internal fragmentation.

4.5. Example of segmentation

As an example we have five segments numbered segment 0, segment 1, segment 2, segment 3, and segment 4. The segments are stored in physical memory as shown. The segment table has a separate entry for each segment, giving the beginning address of the segment in physical memory

(or base) and the length of that segment (or limit). For above example, segment 0 is 500 bytes long and begins at location 600. Thus, the memory space in physical memory for segment 0 will start from 600 and will be up to 1100 (600 + 500). Segment 1 is 400 bytes long and begins at location 1200. Thus, the memory space in physical memory for segment 1 will start from 1200 and will be up to 1600 (1200 + 400) and so on.

Hardware for segment

A segmented address space can be implemented by using address mapping hardware.

CPU generates a logical address which contains two parts.

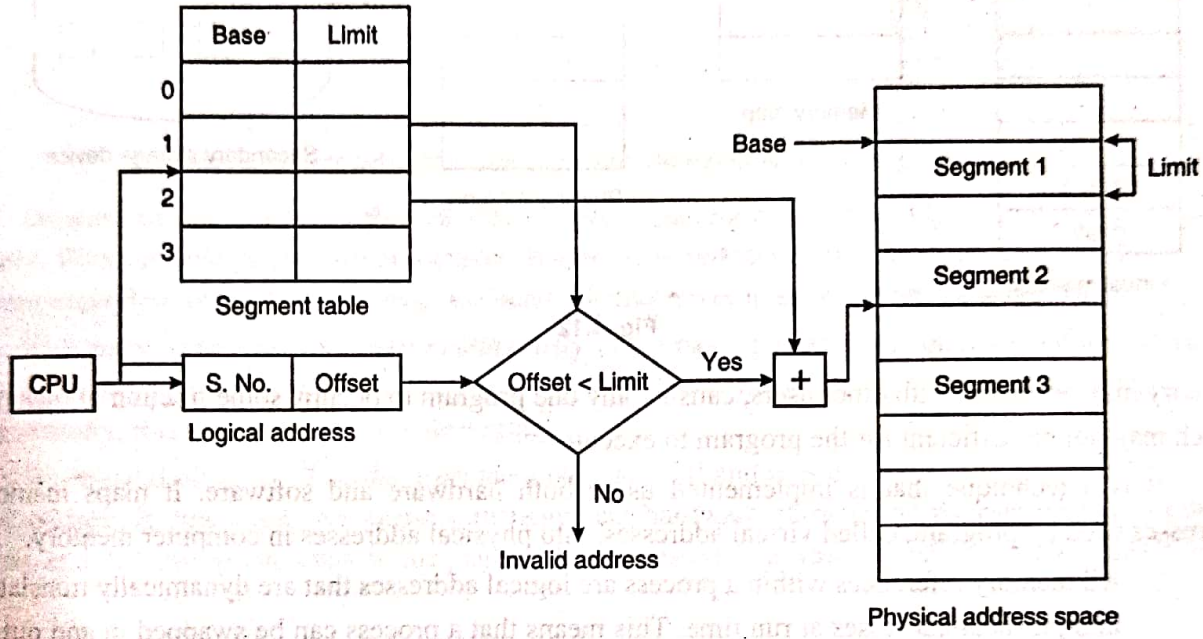


Fig. 4.11

1. Segment number (S)
2. Offset (d)

The segment number is mapped to the segment table. The limit of the respective segment is compared with the offset. If the offset is less than the limit then the address is valid otherwise it throws an error as the address is invalid.

In the case of valid address,the base address of the segment is added to the offset to get the physical address of actual word in the main memory.

4.6. Virtual Memory

Virtual memory involves the separation of logical memory as perceived by users from physical memory. This separation allows an extermley large virtual memory to be provided for programmers when only a smaller physical memory is available.

In many computer systems, programmers often realize that some of their large programs cannot fit in main memory for execution. Even if there is enough main memory for one program, the main

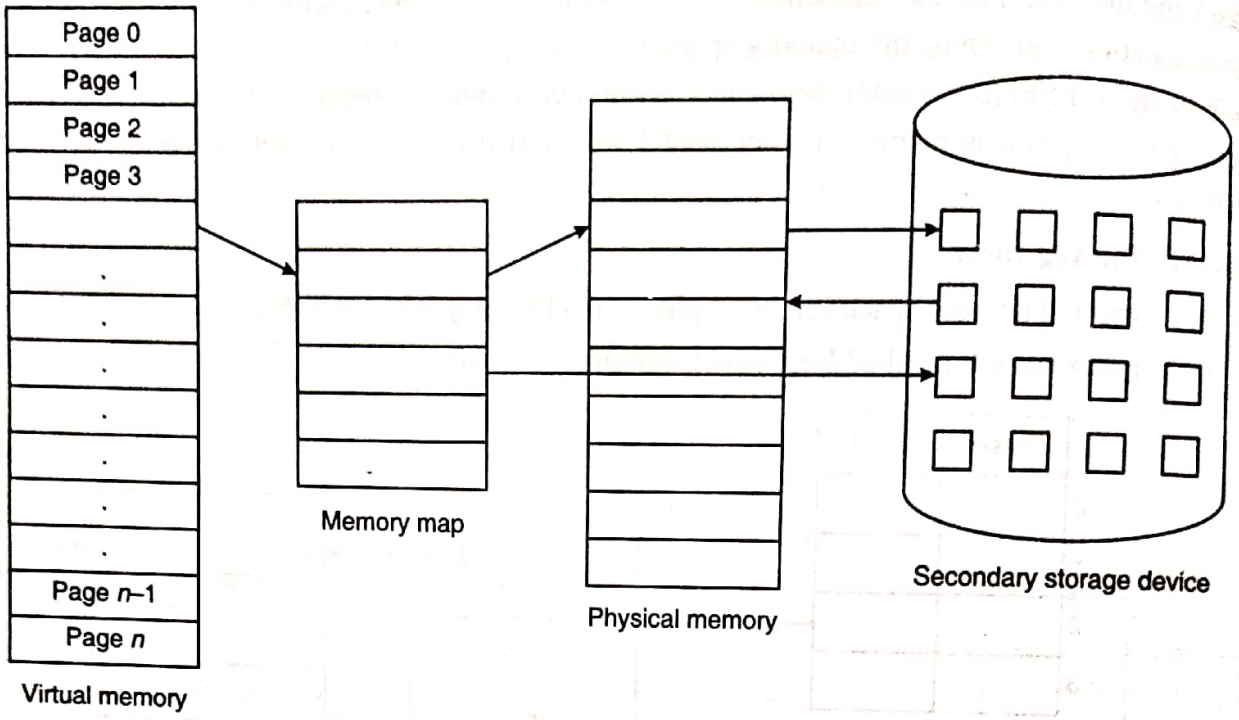


Fig. 4.12

memory may be shared with other users, causing any one program to occupy some fraction of memory which may not be sufficient for the program to execute.

It is a technique that is implemented using both hardware and software. It maps memory addresses used by program, called virtual addresses, into physical addresses in computer memory.

1. All memory references within a process are logical addresses that are dynamically translated into physical addresses at run time. This means that a process can be swapped in and out of main memory such that it occupies different places in main memory at different times during the course of execution.
2. A process may be broken into number of pieces and these pieces need not be continuously located in the main memory during execution. The combination of dynamic run time address translation and use of page or segment table permits this.

If these characteristics are present then, it is not necessary that all the pages or segments are present in the main memory during execution. This means that the required pages need to be loaded in to memory whenever required. Virtual memory is implemented using demand paging or demand segmentation.

4.6. Demand Paging

A demand paging is similar to a paging system with swapping. With demand paging, a page is brought into main memory only when a reference is made to a location on that page. Lazy swapper concept is used in demand paging. A lazy swapper never swaps a page into a memory unless that page will be needed.

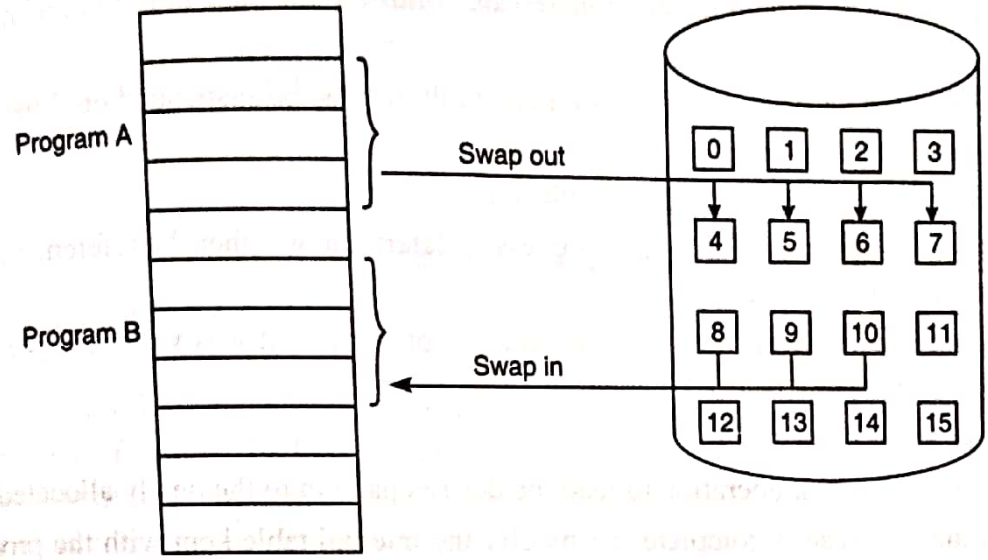


Fig. 4.13 : Transfer of a paged memory to contiguous disk space.

Demand paging combines the features of simple paging and overlaying to implement virtual memory. With demand paged virtual memory, pages are only loaded when they are demanded during program execution; pages that are never accessed are thus never loaded into physical memory.

Each page of program is stored contiguously in the paging swap space on a secondary storage. As locations in pages are referenced, the pages are copied into memory page frames. Once the page is in the memory, it is accessed as in simple paging.

When valid bit is set, then the associated page is legal and present in the memory. Whenever a virtual address is generated, the memory management hardware extracts the page number from the address, and the appropriate entry in the page table is accessed. The valid-invalid bit is checked. If the

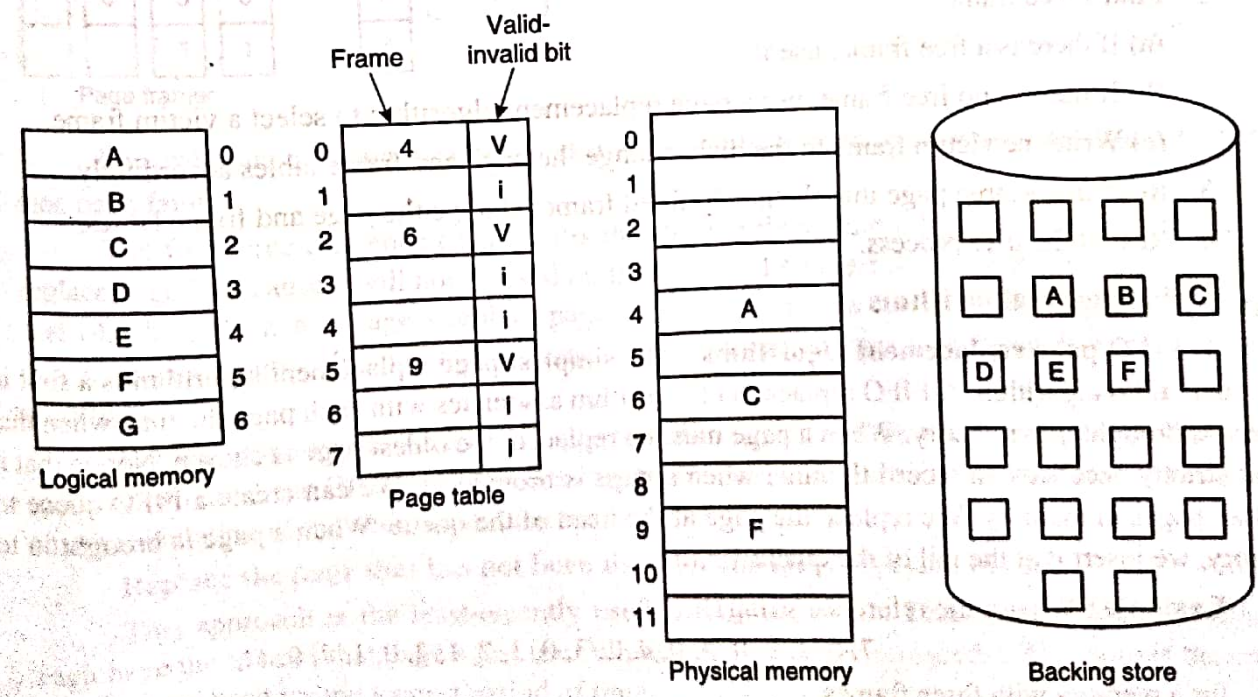


Fig. 4.14 : Page table with valid invalid bits.

page is not in memory, a page fault occurs transferring control to the page fault routine in the operating system.

When the running process experiences a page fault, it must be suspended until the missing page is brought into main memory.

The procedure for handling this page fault is straight forward.

1. We check an internal table for this process to determine whether the reference was a valid or an invalid memory access.
2. If the reference was invalid, we terminate the process. If it was valid, but we have not yet brought in that page, we now page it in.
3. We find a free frame.
4. We sechedule a disk operation to read the desired page in to the newly allocated frame.
5. When the disk read is complete, we modify the internal table kept with the process and the page table to indicate that the page is now in memory.
6. We restart the instruction that was interrupted by the trap. The process can now access the page as through it had always been in memory.

4.6. Page Replacement

Page replacement takes the following approach. If no frame is free, we find one that is not currently being used and free it. We can free a frame by writing its contents to swap space and changing the page table to indicate that the page is no longer in memory. We can now use the freed frame to hold the page for which the process faulted. We modify the page fualt service routine to include page replacement.

1. Find the location of the desired page on the disk.
2. Find a free frame.
   (a) If there is a free frame, use it.
   (b) If there is no free frame, use a page replacement algorithm to select a victim frame.
   (c) Write the victim frame to the disk; change the page and frame tables accordingly.
3. Read the desired page into the newly freed frame; change the page and frame tables.
4. Restart the user process.

Page replacement algorithms

1. FIFO page replacement algorithms : The simplest page replacement algorithm is a first in first out (FIFO) algorithm. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. Notice that it is not strictly necessary to record the time when a page is brought in. We can create a FIFO queue to hold all pages in memory. We replace the page at the head of the queue. When a page is brought in to memory, we insert it at the tail of the queue.

Example : We use the reference string.

7, 0, 1, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 1, 2, 0, 1, 7, 0, 1

for a memory with three frames.

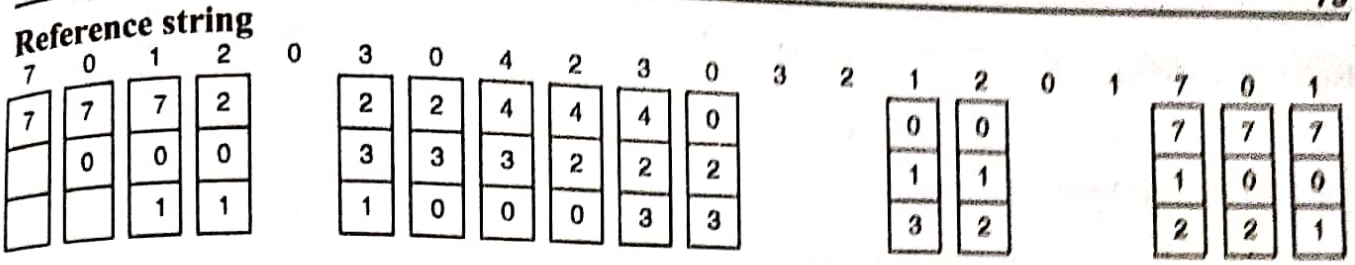


Fig. 4.15 : FIFO page replacement algorithm

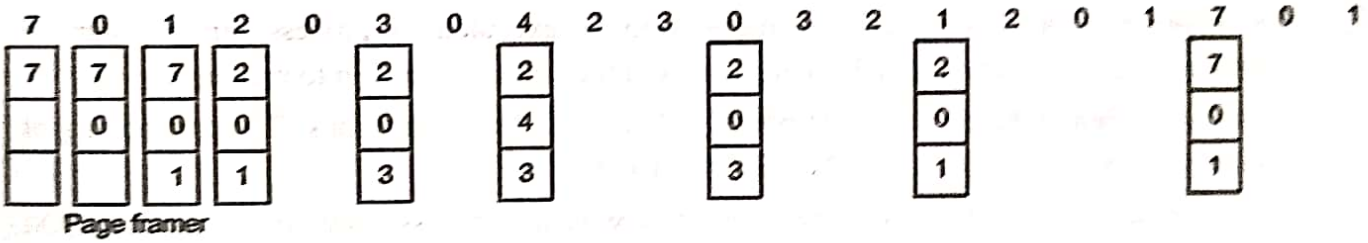
For our example reference string,our three frames are initially empty. The first three references (7, 0, 1) cause page faults and brought into these empty frames. The next reference 2 replace page 7, because page 7 was brought in first. Since 0 is the next reference and 0 is already in memory, we have no fault for this reference. The first reference to 3 results in replacement of page 0, since it is now first in line. Because of this replacement, the next reference to 0, will fault. page 1 is then replaced by page 0. This process continues as shown in fig. Every time a fault occurs, we show which pages are in our three frames. There are 15 faults altogether.

2. Optimal page replacement : An optimal page replacement algorithm has the lowest page fault rate of all algorithm. Such an algorithm does exist and has been called OPT or min. It is simply this.

Replace the page that will not be used for the longest period of time.

Use of this page replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames.

Reference string



For example, on above reference string, the optimal page replacement algorithm would yield nine page faults.

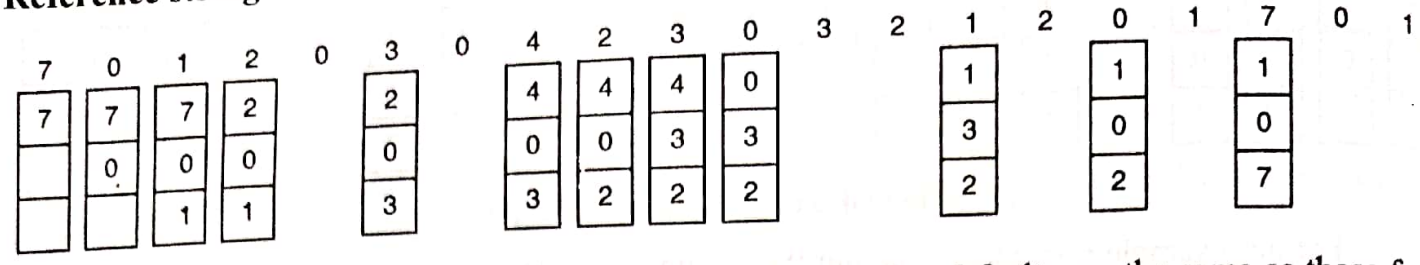
The first three reference cause faults that fill the three empty frames. The reference to page 2 replace page 7, because 7 will not be used until reference 18, whereas page 0 will be used at 5 and page 1 at 14. The reference to page 3 replace page 1, as page 1 will be the last of three pages in memory to be referenced again. With only nine page fualts, optimal replacement is much better than a FIFO algorithm.

3. LRU page replacement : FIFO algorithm uses the time when a page was brought in to memory, where as the OPT algorithm uses the time when a page is to be used. If we use recent past as an approximation of the near future then we can

Replace the page that has not been used for the longest period of time

This approach is the least-recently used (LRU) algorithm. LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time.

Reference string



The LRU algorithm produces 12 faults. Notice that the first 5 faults are the same as those for optimal replacement. When the reference to page 4 occurs, however LRU replacement sees that of the three frames in memory page 2 was used least recently. Thus, the LRU algorithm replaces page 2, not knowing that page 2 is about to be used. When it then faults for page 2, the LRU algorithm replace page 3, since it is now the least recently used of the three pages in memory.

SUMMARY

Logical address : Logical address is generated by CPU while a program is running. Logical address does not exist physically.

Physical address : Physical address identifies a physical location of required data in a memory.

Swapping : A process must be in memory to be executed. A process can be swapped temporarily out of memory to a backing store and then brought back in to memory.

Fixed partitions : Fixed partitions which is known as static partitions. The partitioning of memory in fixed partition is done before processing.

Variable partitions : In variable partitions whenever the processes are come in to the RAM only then we are allocating the space for processes. It is also known as dynamic partitions.

Internal fragmentation : Memory block assigned to processes is bigger, some portion of memory is left unused, as it can not be used by another processes.

External fragmentation : Total memory space is enough to satisfy a request or to reside a process in it, but is not contiguous, so it can not be used.

Paging : Paging is a memory management scheme that permits the mapping to logical memory and physical memory. Paging involves breaking physical memory into fixed sized block called frames and breaking logical memory in to blocks of same size called pages.

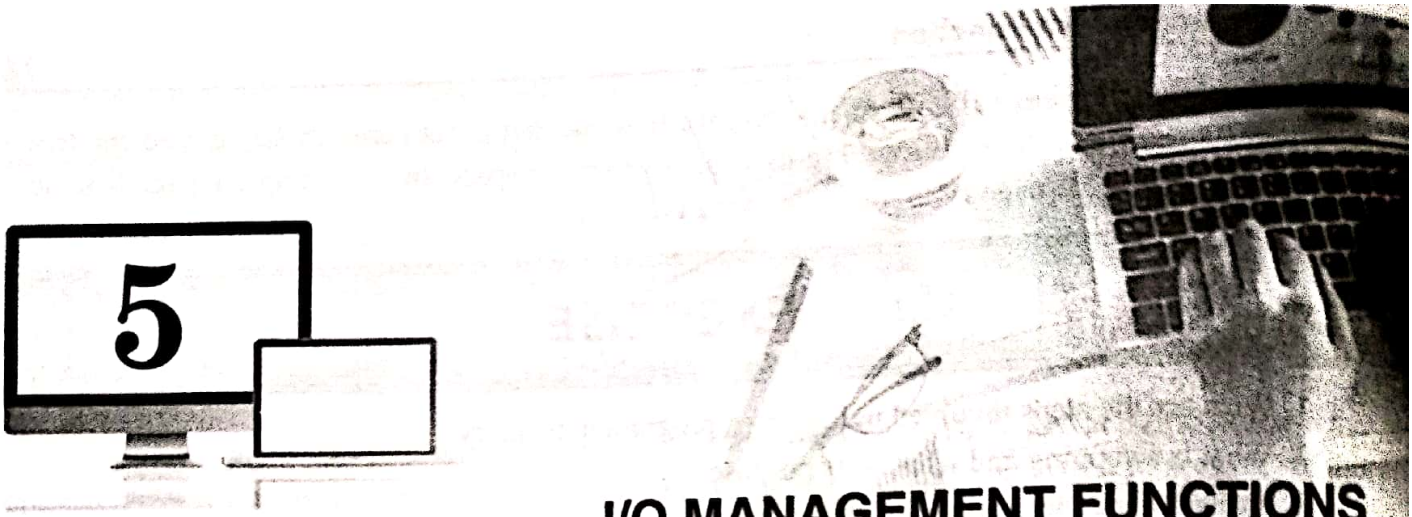
Segmentation : Segmentation divides a program into a number of smaller blocks called segments. A logical address space is a collection of segments. A logical address space is a collection of segments. Each segment has a name and a length.

Demand paging : We can use demand paging to reduce the number of frames allocated to a process. It also allows processes to be run even though their memory requirements exceed the total available physical memory. Such processes run in virtual memory.

Page replacement : If no frame is free, we find one that is not currently being used and free it.We can free a frame by writing its contents to swap space and changing the page table to indicate that the page is no longer in memory.

EXERCISE

1. What are the steps involved in loading a program in memory?
2. What is swapping and what is its purpose?
3. Explain dynamic partition of memory.
4. What is compaction?
5. What is fragmentation? Explain its types.
6. What is paging? Explain the basic method for implementing paging.
7. Explain concept of segmentation.
8. Explain difference between segmentation and paging.
9. What is virtual memory?
10. What is demand paging?
11. What is need of page replacement algorithm?
12. Explain FIFO page replacement algorithm.
13. Discuss LRU page replacement algorithm.



5

I/O MANAGEMENT FUNCTIONS

I/O Management Functions (Principles and Brief Concept) Dedicated Devices, Shared Devices, I/O Devices, Storage Devices, Buffering, Spooling.

5.1. I/O management

Input devices are used to get information into a computer system, and include peripheral devices like the keyboard and mouse now found attached to vertually all computer systems. Output devices receive information from a computer and include devices such as monitor and printers.

The transfer of data into or out of the computer can take place one character at a time (key board input) or in fixed size blocks (as for the transfer of data between secondary storage and working memory). In the personal computer systems of the 1980s and 90s,devices such as printers and disk drives were connected to the system's main circuit board (the mainboard or mother board) via parallel cables, allowing a number of bits to be sent along the cable at the same time using multiple signal wires.

One of the main functions of an operating system is to control access to the input and output devices attached to the system's mainboard. It must respond to user keystrokes and mouse clicks, interpret I/O request from user application and arbitrate when two or more processes require the services of a device at the same time.

Issues in I/O management

Let us first examine the context of input output in a computer system. We shall look at issues initially from the point of view of communication with a device. We notice that communication is required at the following three levels.

The need for a human to input information and receive output from a computer.

The need for a device to input information and receive output from a computer.

The need for computer to communicate (receive/send information) over network.

Organization of I/O function

Device management is the part of the operating system responsible for directly manipulating the hardware devices. Device management is implemented through the interaction of a device driver and interrupt routine.

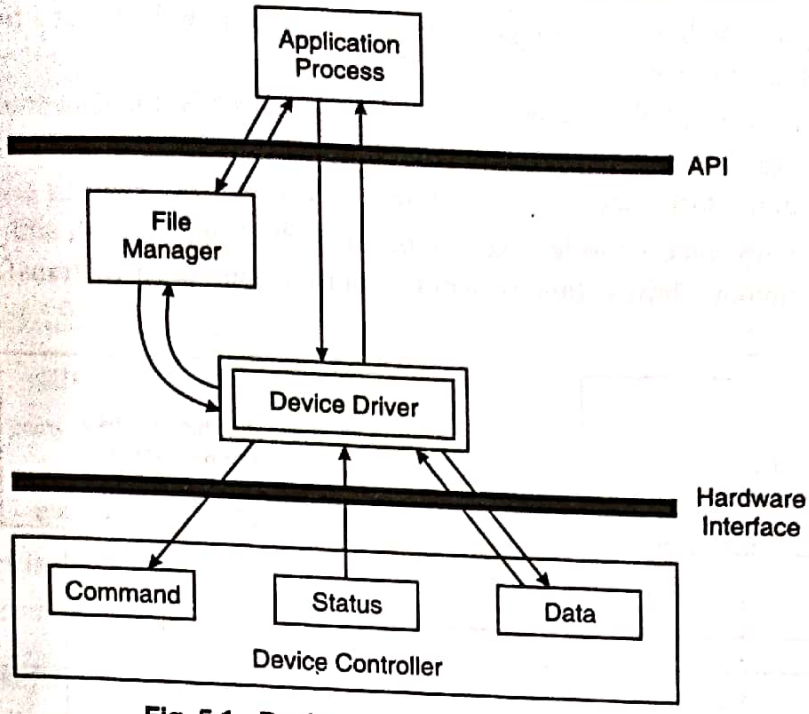


Fig. 5.1 : Device management organization

To start I/O operation, the CPU loads appropriate instruction and values into the registers of the device controller via the device driver. The device controller examines the registers.

1. The request may be read or write instruction.
2. The controller performs the appropriate actions.

Computers employ the following four basic mode of I/O operations.

1. Programmed mode
2. Polling mode
3. Interrupt mode
4. Direct memory access mode.

1. Programmed I/O mode : Programmed I/O instructions are the result of I/O instructions written in computer program. Each data item transfer is initiated by the instruction in the program.

Usually the program controls data transfer to and from CPU and peripheral. Transferring data under programmed I/O requires constant monitoring of the peripherals by the CPU.

2. Polling mode : In this mode, the system interrogates each device in turn to determine if it is ready to communicate. If it is ready, communication is initiated and subsequently the process continues again to interrogate in the same sequence. This is just like a round-robin strategy. Each I/O device gets an opportunity to establish communication in turn. No device has a particular advantage over other devices.

Polling is quite commonly used by systems to interrogate ports on a network. Polling may also be scheduled to interrogate at some pre-assigned time intervals.

3. Interrupt mode : Interrupt I/O is an alternative scheme dealing with I/O. Interrupt I/O is a way of controlling input/output activity whereby a peripheral or terminal that needs to make or receive

a data transfer sends a signal. This will cause a program interrupt to be set at a time appropriate to the priority level of the I/O interrupt.

When the interface determines that the peripheral is ready for data transfer, it generates an interrupt after receiving the interrupt signal. The CPU stops the task which it is processing and services the I/O transfer and then returns back to its previous processing task.

4. Direct memory access mode : Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer. This technique is known as DMA.

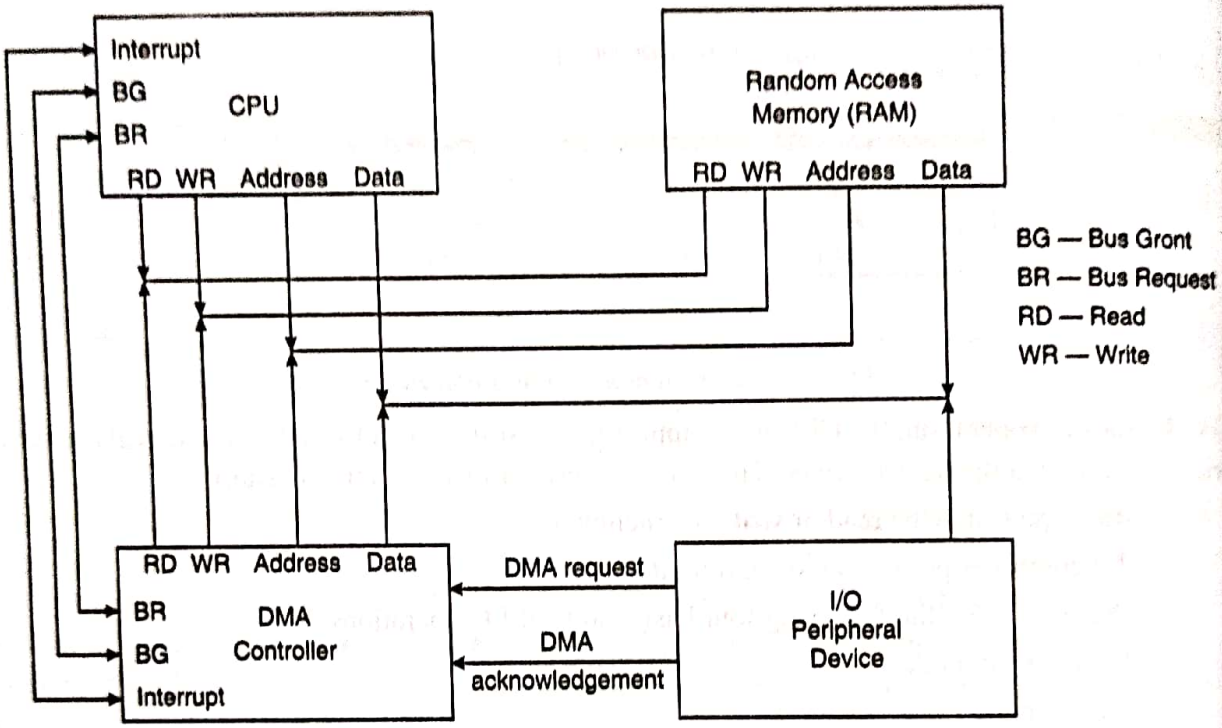


Fig. 5.2 : DMA transfer in a computer system

In this, the interface transfer data to and from the memory through memory bus. A DMA controller manages to transfer data between peripherals and memory unit.

5.2. Dedicated Devices

Dedicated devices are company-owned devices that fulfill a single use-case, such as digital signage, ticket printing, or inventory management. This allows admins to further lock down the usage of a device to a single app or small set of apps and prevents users from enabling other apps or performing other actions on the device.

Dedicated devices are fully managed device that serve a specific purpose. Android provides APIs that can help you create devices that cater to employee and customer specific needs.

Employee-facing : Inventory management, field service management, transport and logistics.

Customer facing : Kiosks, digital signage hospitally check in.

Some popular examples

Device	Single purpose
ATM	Cash dispensing/financial transaction
Airport kiosk	Flight check-in-boarding pass and baggage claims
POS	Payment acceptance
Delivery device	Scanning packages/collecting signatures

Dedicated device features

Android includes APIs to help people using dedicated devices focus on their tasks.

- Used for a single use case.
- Runs a restricted number of apps, and in some cases one app in kiosk mode.
- The enterprise owns the device.
- Devices are persistent and mission critical-always on.

5.3. Shared Devices

These are devcies that can be shared between several processes considering an example like a hard disk, it is shared, but interleaving between different processes requests. All conflicts for devices need to be resolved but pre-determined policies to decide which request is handled first.

The devices like disks, drums and other direct access devices are shared devices because these devices can be shared by several jobs at a time. Many jobs can be read from disk at a time. These devices are efficient but difficult to manage.

For example : If two jobs request a read from disk, some mechanism must be employed to determine which request should be handled first.

5.4. I/O Devices

Input/output management is the least organized of the components of a computer system. Stalling identifies three general categories of I/O devices.

1. Human readable : These devices are for the human interface. Human readable is suitable for communicating with the computer user. Examples are printers, video display terminals, keyboard, etc.

2. Machine readable : Machine readable is suitable for communicating with electronic equipment. Example are disk and tape drives sensors, controllers, etc.

3. Communication : Communication is suitable for communicating with remote devices. Example are network devices and modems. The following differences in characteristics contribute to the inconsistency of I/O devices.

(i) Data rate : There may be differences of several orders of magnitude between the data transfer rates.

(ii) Application : Different devices have different use in the system.

(iii) Complexity of control : A disk is much more complex whereas printer requires simple control interface.

(iv) Unit of transfer : Data may be transferred as a stream of bytes or characters or in larger blocks.

(v) Data representation : Different data encoding schemes are used for different devices.

(vi) Error conditions : The nature of errors differ widely from one device to another.

5.4. Storage Devices

A storage device for a computer enables its user to store and safely access the data and applications on a computer device. Knowing and learning about these computer storage devices is necessary as it works as one of the core components of the system.

Computer storage device definition

A hardware device which can be used to store digital data and applications which may be in the form of images, video, audio, etc. is called a storage device. It is a key componet of a computer and the hard drive is one of its examples.

Types of computer storage device

The computer storage devices can be classified in to various parts, but the computer storage unit is also divided in to three parts.

1. Primary storage : This is the direct memory which is accessible to the central processing unit (CPU).

- This is also known as the main memory and is volatile.
- This is temporary. As soon as the device turns off or is rebooted the memory is erased.
- It is smaller in size.
- Primary storage comprises only of internal memory.
- Example of primary storage include RAM, cache memory etc.

2. Secondary storage : This type of storage does not have direct accessibility to the central processing unit.

- The input and output channels are used to connect such storage devices to the computer, as they are mainly external.
- It is non volatile and larger storage capacity in comparison to primary storage.
- This type of storage is permanent until removed by an external factor.
- It comprises of both internal and external memory.
- Example of secondary storage are USB drives, flophy disk etc.

3. Tertiary memory : This type of storage is generally not considered to be important and is generally not part of personal computers.

- It involves mounting and unmounting of mass storage data which is removable from a computer device.
- This type of storage holds robotic functions.
- It does not always require human intervention and can function automatically.

5.5. List of computer Storage Devices

There are four types of devices in which computer data can be stored. Discussed below are the same in detail.

Magnetic Storage Devices

The most commonly used storage devices in today's time are magnetic storage devices. These are affordable and easily accessible. A large amount of data can be stored in these through magnetised mediums.

A magnetic field is created when the device is attached to the computer and with the help of the two magnetic polarities, the device is able to read the binary language and store the information given below are the examples of magnetic storage devices.

Floppy disk : It is a removable storage devcie which is in the shape of a square and comprises magnetic elements. When placed in the disk reader of the computer device, it spins around and can store information. Lately, these floppy disks have been replaced with CD's, DVD's and USB drives.

Hard drive : This primary storage device is directly attached to the motherboard's disk controller. It is an integral storage space as it is required to install any new program or application to the device. Software programs, images, videos, etc. can all be saved in a hard drive and hard drives with storage space in terabytes are also easily available now.

Zip disk : Introduced by lomega, is a removable storage device which was intially released with a storage space of 100 MB which was later increased to 250 and then finally 750 MB.

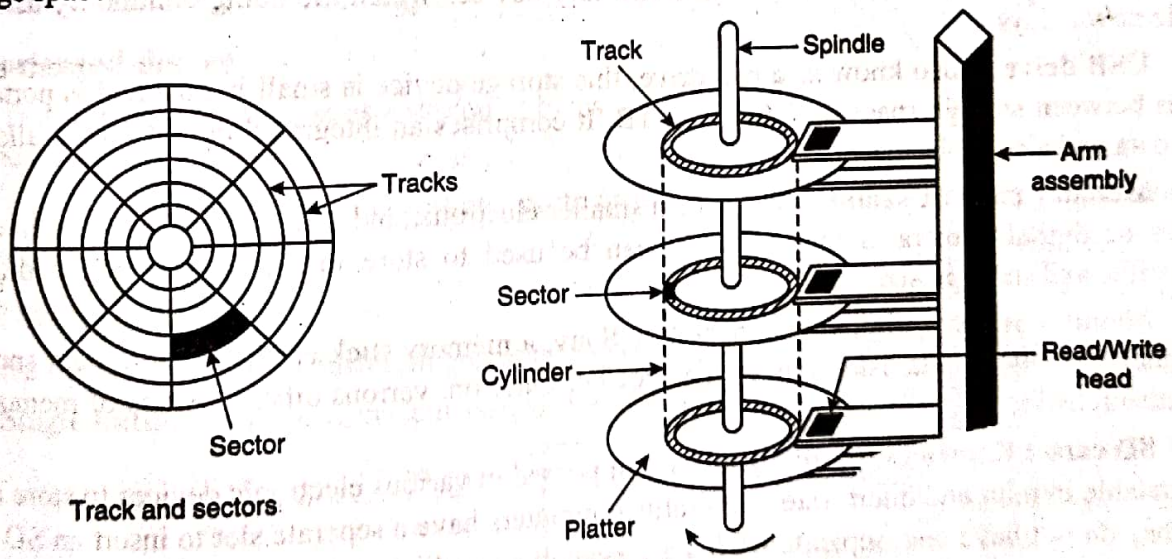


Fig. 5.3

Magnetic strip : A magnetic strip is attached in the dev
suitable example for this is a debit card which has a strip place
digital data.

- Magnetic disks provide bulk of secondary storage of mo
- Bits of data (0's and 1's) are stored on circular magnetic p

1. Disk contains concentric tracks.

2. Tracks are divided in to sectors.
3. A sector is the smallest addressable unit in a disk.
4. A cylinder is the set of tracks at a given radius of a disk pack.

5.6. Optical Storage Devices

Such devices use lasers and lights to detect and store data. They are cheaper in comparison to USB drives and can store more data. Discussed below are a few commonly used optical storage devices.

CD-ROM : This stands for compact disc-read only memory and is an external device which can store and read data in the form of audio or software data.

Blu-ray disc : Introduced in 2006, Blu-ray disk was backup by major IT and computer companies. It can store up 25 GB data in a single layer disc and 50 GB data in a dual layer disc.

DVD : Digital versatile disc is another type of optical storage device. It can be readable, recordable and re-writable. Recording can done in such devices and then can be attached to the system.

CD-R : It is a readable compact disc which uses photosensitive organic dye to record data and store it. They are a low-cost replacement for storing software and applications.

Flash memory deivces

These storage devices have now replaced bhoth magnetic and optical storage devices. They are easy to use, portable and easly available and accessible. They have become a cheaper and more convenient option to store data.

Discussed below are the major flash memory devices which are being commonly used by the people now a days.

USB drive : Also know as a pen drive, this storage device is small in size and is portable and ranges between storage space of 2 GB to 1 TB. It comprises an integrated circuit which allows it to store data and also replace it.

Memory card : Usually attached with smaller electronic and computerised devices like mobile phones or digital camera, a memory card can be used to store images, video and audio and is compatible and small in size.

Memory stick : Originally launched by Sony, a memory stick can store more data and is easy and quick to transfer data using this storage device. Later on, various other versions of memory stock were also released.

SD card : Known as secure digital card, it is used in various electronic devices to store data and is avaialable in mini and micro size. Generally, computers have a separate slot to insert an SD card. In case they do not have one, separate USB's are available in which these cards canbe inserted and then connected to the computer.

Characteristics of storage devices

Given below are a few characteristics of these storage devcies.

- Because of volatile memory, the data stored can be saved and also replaced whenever needed.
- These devices are readable, writable and rewritable which ensure that data saved if not necessary can be removed of replaced accordingly.
- The capacity and size of these drives and devices has become an added advantage.

comprising digital data. The most
one of its sides which stores
computers.
ers called disks.

- Even in terms of performance, using these storage devices the data can be saved easily but can also be transferred easily from one device to another.

Buffering

Operating system stores data in memory while transferring to or from devices.

- To cope with device speed mismatch.
- To cope with device transfer size mismatch.
- To maintain "Copy semantics".

I/O is the process of transferring data between a program and an external device. The process of optimizing I/O consists primarily of making the best possible use of the slowest part of the path between the program and the device.

The slowest part is usually the physical channel which is oftern slower than the CPU or a memory to memory data transfer.

A buffer is a temporary storage location for data while the data is being transferred.

Block Oriented Device

- Stores information in blocks that are usually of fixed size.
- Transfers are made one block at a time.
- Possible to reference data by its block number.
- Disks and USB keys are example.

Stream oriented deivce :

- Transfers data in and out as a stream of bytes.
- No block structure.
- Terminals, printers, communications ports and most other devcies that are not secondary storage are examples.

Types of buffering :

1. **No buffer :** Without a buffer, the operating system directly accesses the device.
2. **Single buffer :** Operating system assigns a buffer in main memory for an I/O request.

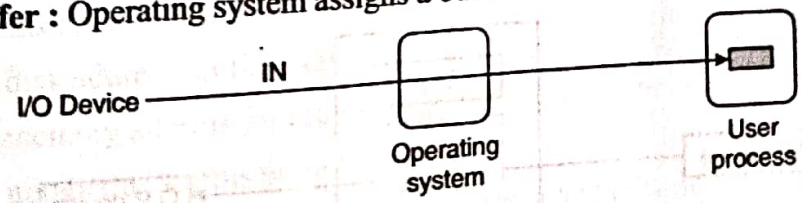


Fig. 5.4

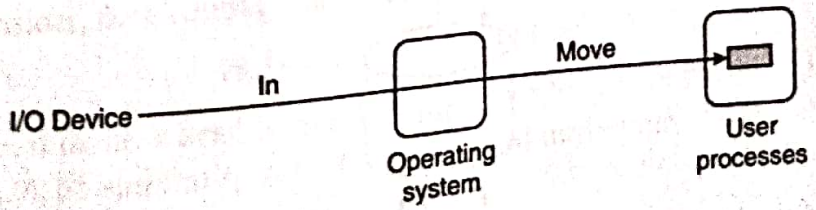


Fig. 5.5

3. Double buffer : A process can transfer data to or from one buffer while the operating system empties or fills the other buffer. Also known as buffer swapping.

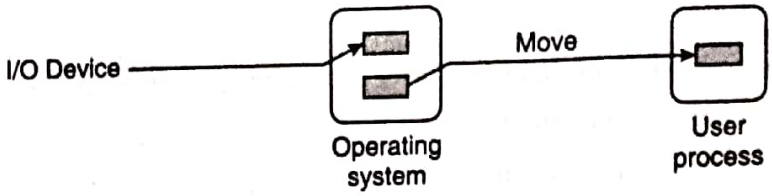


Fig. 5.6

4. Circular buffer :

- Two or more buffers are used.
- Each individual buffer is one unit in a circular buffer.
- Used when I/O operation must keep up with process.

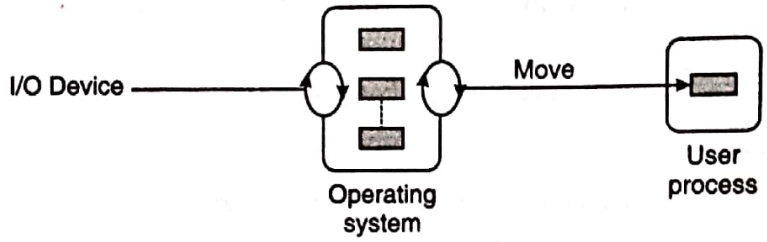


Fig. 5.7

Spooling

Spooling stands for "simultaneous peripheral operation online". So in a spooling, more than one I/O operation can be performed simultaneously at the time when the CPU is executing some process then more that one I/O operation can also done at the same time.

From the above image, we can see that the input data is stored in some kind of secondary device and this data is then fetched by the main memory. The benefit of this approach is that, in general the CPU works on the data stored in the main memory. Since we can have a number of input devices at a time, so all these input devices can put the data into the disk or secondary memory. Then the main memory will fetch the data one by one from the secondary memory and the CPU will excute some instruction on that data.

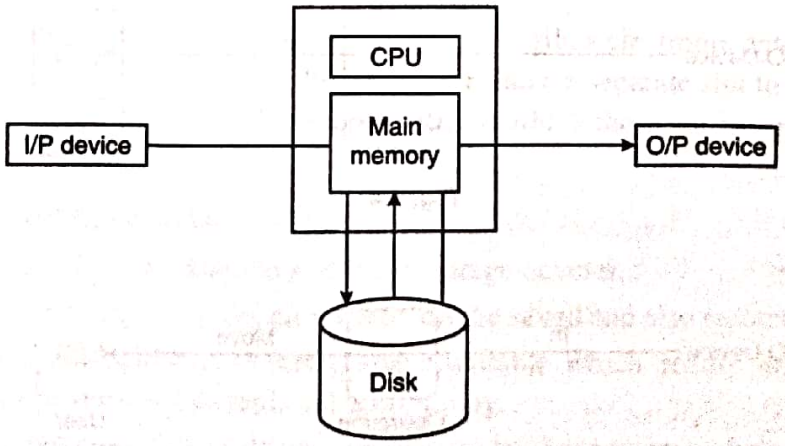


Fig. 5.8

When the CPU is excecuting some task then at that time, the input deices need not wait for its turn. They can directly put their data in the secondary memory without waiting for its turn.

When the CPU generates some output, then that output is first stored in the main memory and the main memory transfers that output to the secondary memory, the output will be provided to some output devices.

For example : In a printer spooling there can be more than one documents that need to be printed. So the documents can be stored into the spool and the printer can fetch that documents and print the documents one by one.

Advantages of spooling

Since there is no interaction of I/O devices with CPU, so the CPU need not wait for the I/O operation to take place. The I/O operations take a large amount of time.

The CPU is kept busy most of the time and hence it is not in the idle state which is good to have a situation.

More than one I/O devices can work simultaneously.

Disk Scheduling

The performance of a computer system is to a large extent dependent upon how fast a disk request is serviced. Since most jobs nearly depend upon the disk for input and output purpose.

In multi programmed environment, many processes may be generating request for reading and writing disk records because these processes often make requests faster than they can be serviced by disk systems, waiting lines or queues build up for each device.

For the disk drives, meeting this responsibility entails having fast access time and large disk bandwidth. The access time has two major components. The seek time is the time for the disk arm to mvoe the heads to the cylinder containing the desired sector. The disk band width is the total number of bytes transferred, divided by the total time between the first request for service and the completion of the last transfer.

Whenever a process needs I/O to or from the disk, it issues a system call to operating system. The request specifies several pieces of information.

- Whether this operation is input or output
- What the disk address for the transfer is
- What the memory address for the transfer is
- What the number of sectors to be transferred is

1. FCFS scheduling : The simplest form of disk scheduling is, of course, the first come first served (FCFS) algorithm. The algorithm is intrinsically fair, but it generally does not provide the fastest service. Consider, for example, a disk queue with requests for I/O to blocks on cylinders.

98,183, 37, 122, 14, 124, 65, 67

In that order, if the disk head is initially at cylinder 53, it will first move from 53 to 98, then to 183, 37, 122, 14, 124, 65 and finally to 67 for a total head movement of 640 cylinders.

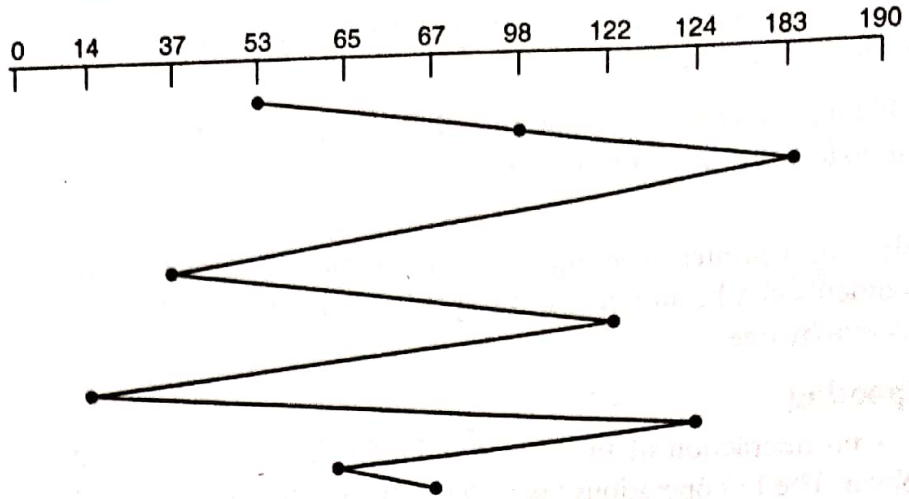


Fig. 5.9 : FCFS disk scheduling

2. SSTF scheduling : It seems reasonable to service all the rquests close to the current head position before moving the head for away to service other first (SSTF) algorithm. The SSTF algorithm selects the request with the minimum seek time from the current head position. Since seek time increases with the number of cylinders traversed by the head, SSTF chooses the pending request closet to the current head position.

For example, request queue, the closest request to the initial head position (53) is at cylinder 65. Once we are at cylinder 65, the next closest request is at cylinder 67. From there, the request at cylinder 37 is closer than the one at 98, so 37 is served next. Continuing, we service the request at cylinder 14, then 98, 122, 124 and finally 183. This scheduling method results in a total head movement of only 236 cylinders.

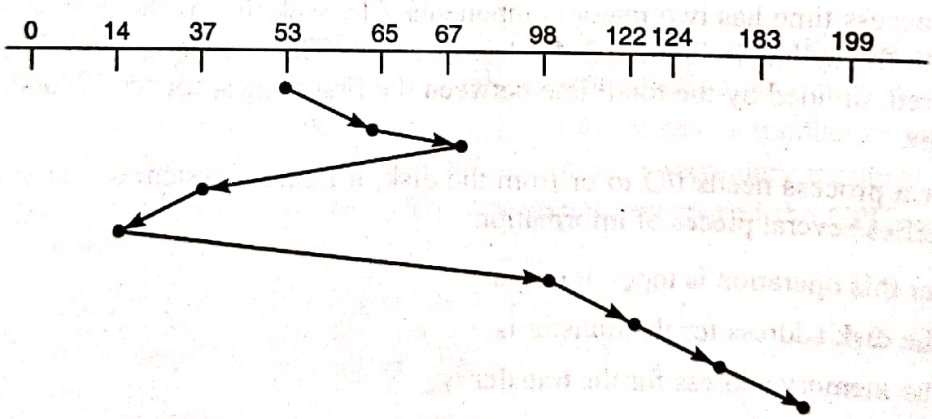


Fig. 5.10 : SSTF disk scheduling

Queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

3. Scan scheduling : In the scan algorithm, the disk arm starts at one end of the disk and moves towards the other end, servicing requests as it reaches each cylinder, until it gets to the other end of the disk. At the other end, the direction of head movement is reversed, and servicing continues. The head continuously scans back and forth across the disk.

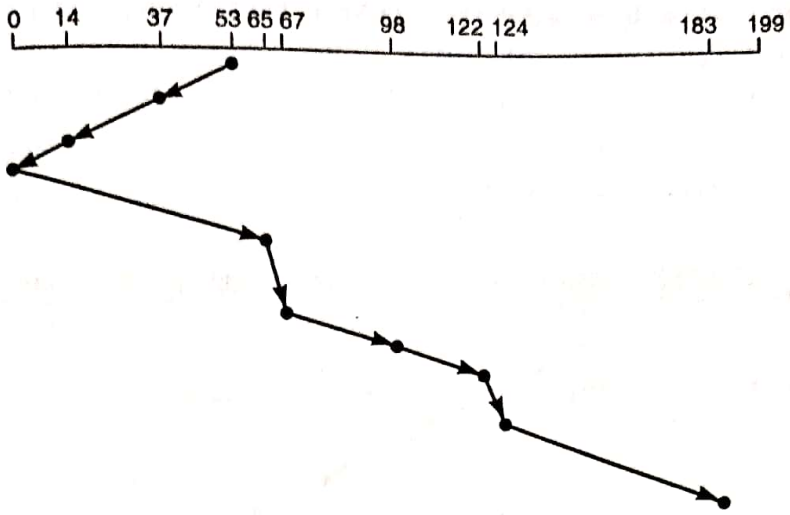


Fig. 5.11 : Scan disk scheduling

Let's return to our example to illustrate. Before applying scan to schedule the requests on cylinders 98, 183, 37, 122, 14, 124, 65 and 67, we need to know the direction of head movement in addtion to the head's current position (53). If the disk arm is moving toward 0, the head will service 37 and the 14. At cylinder 0, the arm will reverse and will move toward the other end of disk, servicing the requests at 65, 67, 98, 122, 124 and 183.

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

4. C-Scan scheduling : Circular scan (C-Scan) sheduling is a variant of scan designed to provide a more uniform wait time. Like scan, C-Scan moved from one end of the disk to the other, servicing request along the way. When the head reaches the other end, however, it immediately returns to the beginning of the disk without servicing and request on the return trip.

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

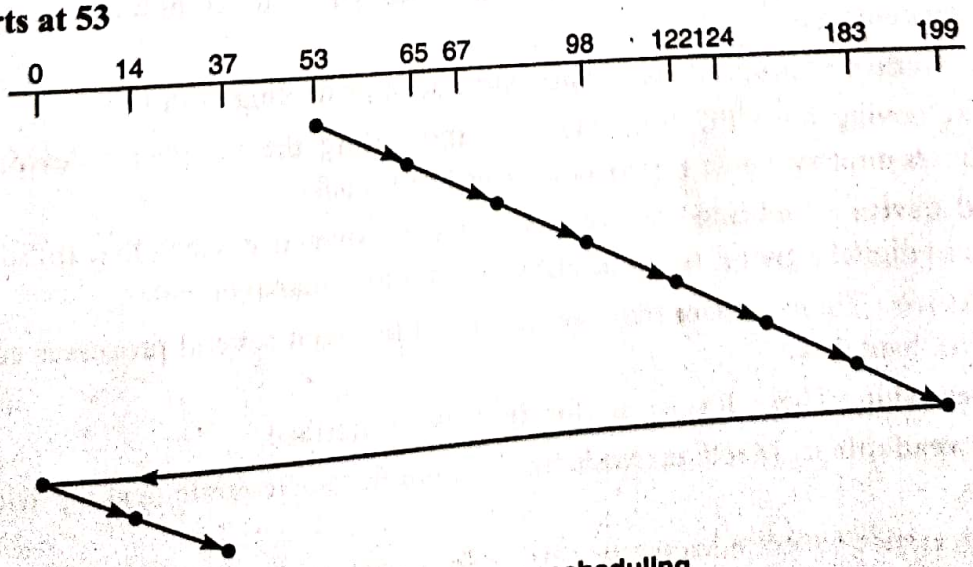


Fig. 5.12 : C-Scan scheduling

Look scheduling : As we described them, both Scan and C-Scan move the disk arm across the full width of the disk. In practice, neither algorithm is often implemented this way. More commonly the arm goes only as far as the final request in each direction. Then it reverses direction immediately, without going all the way to the end of the disk.

queue = 98, 183, 37, 122, 14, 124, 65, 67

head starts at 53

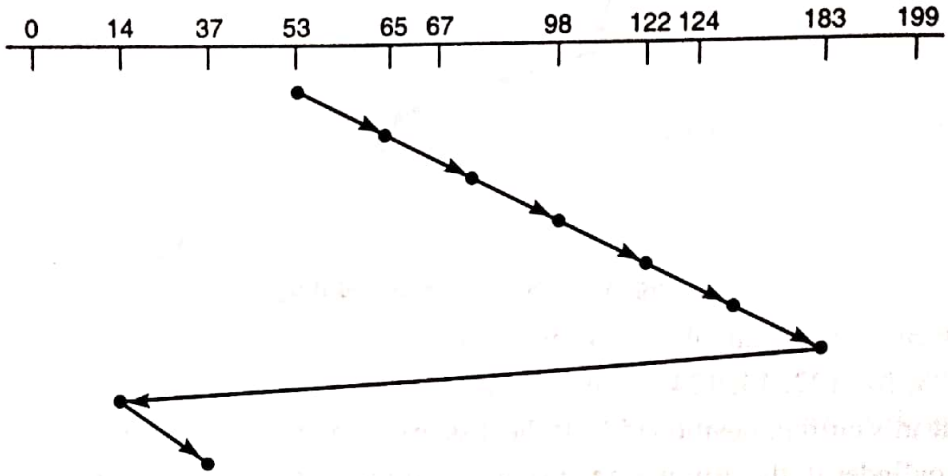


Fig. 5.13 : Look disk scheduling

SUMMARY

I/O management : One of the main functions of an operating system is to control access to the input and output devices attached to the system's mainboard.

Programmed I/O mode : Programmed I/O instructions are the result of I/O instructions written in computer program. Each data transfer is initiated by the instruction in the program.

Polling mode : In this mode, the system interrogates each device in turn to determine if it is ready to communicate.

Interrupt mode : Interrupt I/O is an alternative scheme dealing with I/O.

DMA : Removing the CPU from the path and letting the peripheral device manage the memory buses directly would improve the speed of transfer.

Dedicated device : Dedicated devices are company owned devices that fullill a single use case, such as digital signage, ticket printing or inventory management.

Shared device : These devices that can be shared between several processes considering an example like hard disk.

Human readable : These devcies are for the human interface.

Machine readable : Machine readable is suitable for communicating with electronic equipment.

Communication : Communication is suitable for communicating with remote devices.

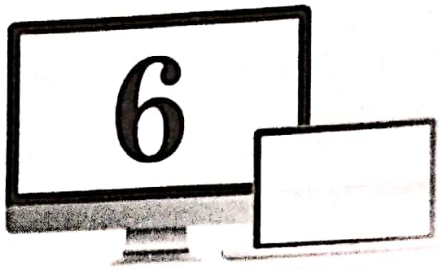
Storage device : A storage device for a computer enables its user to store and safely access the data and applications on a computer device.

Buffering : A buffer is a temporary storage device for data while the data is being transferred. Operating system stores data in memory while transferring to or from devcies.

Spooling : More than one I/O operation can be performed simultaneously.

EXERCISE

1. Explain the dedicated device.
2. Difference between dedicated device and shared device.
3. What is DMA?
4. Explain characteristics of I/O devices.
5. What is buffering and also explain its types.
6. What is storage device?
7. What is spooling?
8. Explain the I/O management functions. ❑



6

FILE MANAGEMENT

Types of File System; Simple file system, Basic file system, Logical file system, Physical file system, Various Methods of Allocating Disk Space

6.1. Introduction

A file is a named collection of related information that is recorded on secondary storage. A file is a collection of similar records. The file is treated as a single entity by users and application and may be referred by name. File have unique file names and maybe created and deleted.

File Attributes

(i) Name : The symbolic file name is the only information kept in human readable form.

(ii) Identifier : This unique tag, usually a number, identifies the file with in the file system. It is the non-human-readable name for the file.

(iii) Type : This information is needed for systems that support different types of files.

(iv) Location : This information is a pointer to a device and to the location of the file on the device.

(v) Size : The current size of the file (in bytes, words, or blocks) and possibly the maximum allowed sizes are included in this attribute.

(vi) Protection : Access-control information determines who can do reading, writing, executing, and so on.

(vii) Time, date and user identification : This information may be kept for creation, last modification and last use.

File operations

1. Creating a file : Two steps are necessary to create a file. First space in the file system must be found for the file. We discuss how to allocate space for the file. Second an entry for the new file must be made in the directory.

2. Writing a file : To write a file, we make a system call specifying both the name of the file and the information to be written to the file.

3. Reading a file : To read a file, system call is used. It requires the name of file and memory address.

4. Deleting a file : To delete a file, we search the directory for the named file. Having found the associated directory entry, we release all file space, so that it can be reused by other files and erase the directory entry.

5. Repositioning within a file : The directory is searched for the appropriate entry, and the current file position pointer is repositioned to a given value.

6. Truncating a file : The user may want to erase the contents of a file but keep its attributes. Rather than forcing the user to delete the file and then recreate it, this function allows all attributes to remain unchanged.

File Types

A name and an extension. Following table gives the file types with usual extension and function.

File Type	Usual Extension	Function
Executable	exe, com, bin, or none	Read to run machine language program
Object	obj, o	Compiled, machine language, not linked
Source code	c, cc, java, pas, asm, a	Source code in various language
Batch	bat, sh	Commands to the command interpreter
Text	txt, doc	Textual data, documents
Word processor	WP, tex, rrf, doc	Various word-processor formats
Library	lib, a, so, dll, mpeg, mov	Librariers of routines for programmers
Print or view	arc, zip, tar	ASCII or binary file in a format for printing
Archive	arc, xip, tar	Related files grouped into one file, sometimes compressed, for archiving or storage
Multimedia	mpeg, mov, rm	Binary file containing audio or A/V information

File structure

File structure uses following four terms :

1. Field : Field is the basic element of data. An individual field contains a single value. For example-first name, date, time, etc.

2. Record : A record is a collection of related fields that can be treated as a unit by some application program.

For example : Student record would contain such field as name, enrollement number, student name, father name, address and so on.

3. File : A file is a collection of similar records. Files have file names and may be created and deleted.

4. Database : A database is a collection of related data. A database may contain all of the information related to an organization or project.

File management

File management is one of the basic and important features of operating system. Operating system is used to manage files of computer system. All the files with different extensions are managed by operating system.

A file is collection of specific information stored in the memory of computer system. File management is defined as the process of manipulating files in computer system, its management includes the process of creating, modifying and deleting the files.

The following are some of the tasks performed by file management of operating system.

1. It helps to create new files in computer system and placing them at the specific locations.
2. It helps in easily and quickly locating these files in computer system.
3. It makes the process of sharing of the files among different users very easy and user friendly.
4. It helps to store the files in separate folders known as directories. These directories helps users to search file quickly or to manage the files according to their types or uses.
5. It helps the user to modify the data of files or to modify the name of the file in the directories.

6.2. Access Mehtods

File store information. When it is used, this information must be accessed and read into computer memory. The information in the file can be accessed in several ways. Some systems provide only one access method for files.

1. Sequential access : The simplest access method is sequential access. Information in the file is processed in order, one record after the other. This mode of access is by far the most common.

Reads and writes make up the bulk of the operation on a file. A read operation-read next-reads the next portion of file and automatically advances a file pointer, which track the I/O location. Similarly, the write operation-write next appends to the end of the file and advances to the end of the newly materials.

1. Direct access : Direct access allows random access to any file block. This method is based on a disk model of a file. A file is made up of fixed length logical records. A direct access file allows arbitrary block to be read or written. It allows programs to read and write records rapidly in no paraticular order.

- In a direct access file, no restrictions for reading or writing a file in any sequence.
- Not all operating systems support both sequential and direct access for files.

3. Indexed access : In general indexed access records are only accessed through their indexes. An index is itself organized as a sequential file for ease of searching. When a new record is added to the main file, all the index files must be updated.

Indexed files are used mostly in applications where timelines of information are critical and where data are rarely processed exhaustively.

6.3. Directory System

A directory contains information about the files, including attributes, location and ownership. We want to be able to insert entries, to delete entries to search for a named entry, and to list all the entries in the directory.

Following operation that may be performed on the directory.

Search for a file : Directory is searched for finding particular file in the directory. Files have symbolic names and similar names may indicate a relationship between files.

Create a file : New files need to be created and added to the directory.

Delete a file : When a file is deleted, an entry must be removed from the directory.

Rename a file : Because the name of a file represents its contents to its users, we must be able to change the name when the contents or use of the file changes.

List a directory : We need to be able to list the files in a directory and the contents of the directory entry for each file in the list.

Types of Directory

1. Single level directory : The simplest directory structure is the single level directory. All files are contained in the same directory, which is easy to support and understand.

When the number of files increases or when the system has more than one user. Since all files are in the same directory, they must have unique names.

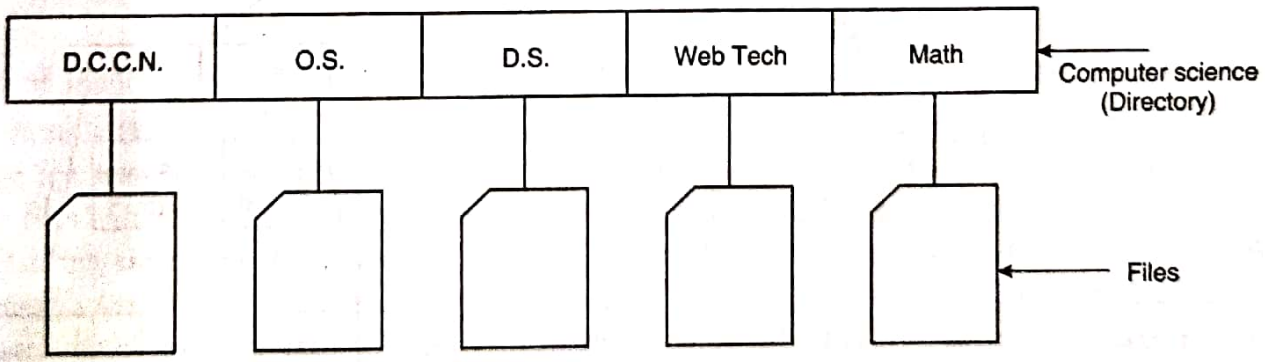


Fig. 6.1

2. Two level directory : A single-level directory often leads to confusion of file names among different users. The standard solution is to create a separate directory for each user.

In the two level directory structure, each user has his own user file directory (UFD). The UFD's have similar structures, but each lists only the files of a single user. When a user job starts or a user logs in, the system's master file directory (MFD) is searched. The MFD is indexed by user name or account number and each entry points to the UFD for that user.

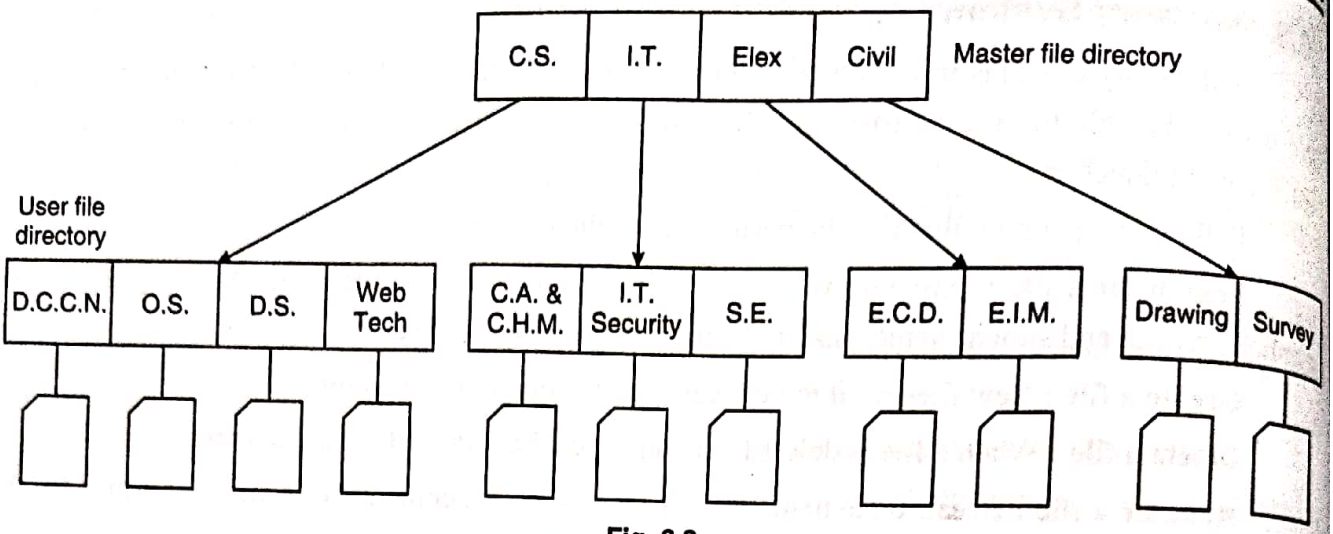


Fig. 6.2

3. Tree-structured directories : This generalization allows users to create their own subdirectories and to organize their files accordingly. A tree is the most common driectory structure. The tree has a root directory, and every file in the system has a unique path name.

A directory (or subdirectory) contains a set of files or subdirectories.

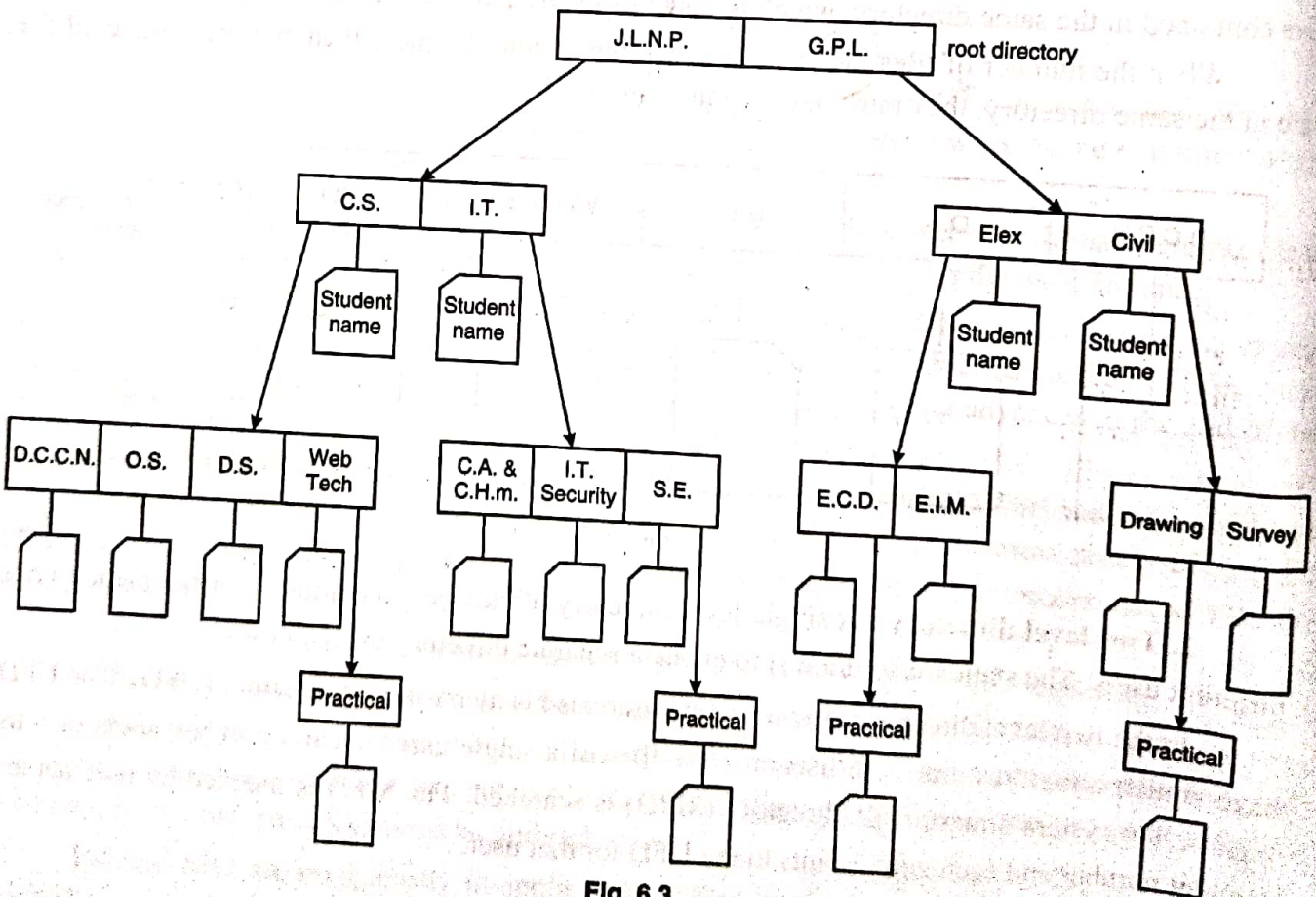


Fig. 6.3

4. Acyclic-graph directories : Acyclic graph directories allow directories to have shared subdirectories and files. Same file or directory may be in two different directories shared files and subdirectories can be implemented by using links.

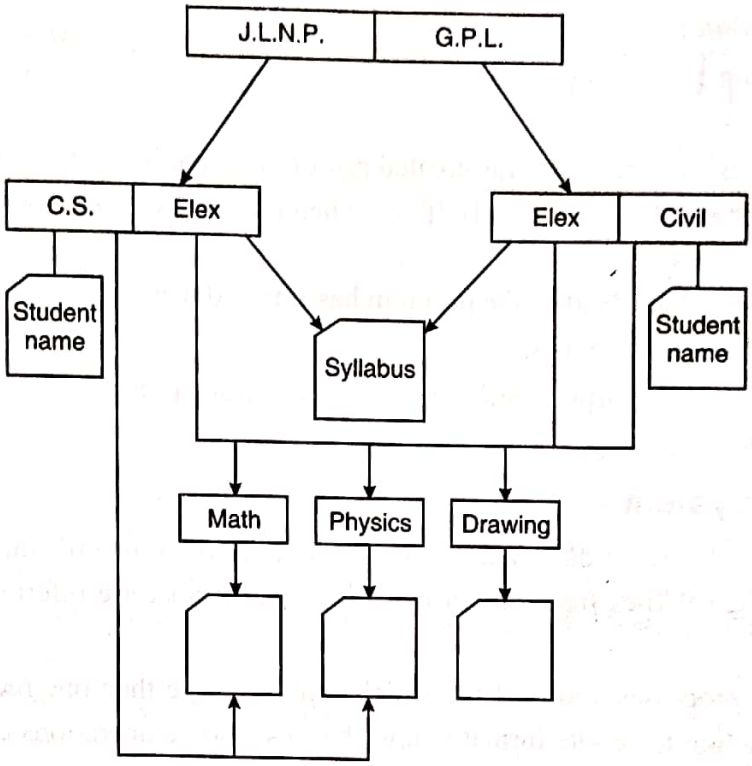


Fig. 6.4

Types of file system

Simple file system : Design and implement a simple file system. The simple file system handles a single application at any given time, it implements no user concept, does not support protection among files. Although these assumptions are quite dramatic it leaves the file system still usable in single tasking environments such as digital cameras and other embedded environment. Also you will implement a simplified interface to file system with notable restrictions such as : limited length file names, limited length file extensions, no subdirectories. (only a single root directory), few file attributes such as size and no file permissions.

Basic file system : The basic file system works directly with the device drivers in terms of retrieving and storing raw blocks of data, without any consideration for what is in each block may be referred to with a single block number with head sector cylinder combinations.

Open/close files : Retrieve and set up information for efficient access.

- Get file descriptor
- Manage open file table.

File descriptor :

- Owner id
- File type

- Protection information
- Mapping to physical disk blocks
- Time of creation, last use, last modification.

Basic file operation :

(A) File operation :

- Create new files
- Open, or start using, a file that was created prior to the running of this program.
- Increase the size of the file (extend), if and when the file has exhausted the current allocated space.
- Stop using (close) the file after the program has finished using it.
- Delete a file that already exists.

(B) Block data transfer operation : Transfer data from (READ) or to (writer) the device designed by the program.

6.4. Logical File System

Logical files do not contain data. They contain a description of records that are found in one or more physical files. Logical files that contain more than one format are referred to as multi-format logical files.

If your program processes a logical file which contains more than one record format, you can use the R format( ) function to set the format you wish to use. Some operations can not be performed on logical files.

The field-level description of the record includes a description of all fields and their arrangement in this record.

The logical file system manages meta data information. Metadata includes all of the file-system structure except the actual data (or contents of the files). The logical file system manages the directory structure to provide the file organization module with the information the latter needs, given a symbolic file name. It maintains file structure via file control blocks.

Physical File System

Physical files contain the actual data that is stored on the system, and a description of how data is to be presented to or received from a program. They contain only one record format, and one or more members. Records in database files can be described using either a field level description or record level description.

A field level description describes the fields in the record to the system. Database files that are created with field level descriptions are referred to as externally described files. A record-level description describes only the length of the record, and not the contents of the record.

A collection of bytes stored on a disk or tape. It occupies the portion of memory. It contains the original data. A physical file contains one record format.

6.5. Allocation Methods

Many files are stored on the same disk. The main problem is how to allocate space to these files so that disk space is utilized effectively and files can be accessed quickly.

On secondary storage, a file consists of a collection of blocks. File management system is responsible for allocating blocks to files. Space is allocated to a file as one or more contiguous units, which we shall refer to as portions.

Three major methods of allocating disk space are in wide use.

1. Contiguous allocation method.
2. Linked allocation method.
3. Indexed allocation method.

1. Contiguous allocation method : Contiguous allocation requires that each file occupy a set of contiguous blocks on the disk. The disk addresses define a linear ordering on the disk. With this ordering, assuming that only one job is accessing the disk, accessing block $b+1$ after block b normally requires no head movement.

Contiguous allocation of a file is defined by the disk address and length (in blcok units) of the first blcok. If the file is n blcoks long and starts at location b, then it occupies block $b, b+1, b+2, \ldots\ldots, b+n-1$. The directory entry for each file indicates the address of the starting block and the length of area allocated for this file.

Example : Following directory contains the files Tushar, Aman and Kajal; and each file defines the starting block and length of area allocated for this file. Such as Tushar's file starting block 2 and length is 3 blocks. So allocated the disk space of Tushar's file are block 2, block 3 and block 4.

Directory

File	Start	Length
Tushar	2	3
Aman	7	2
Kajal	22	3

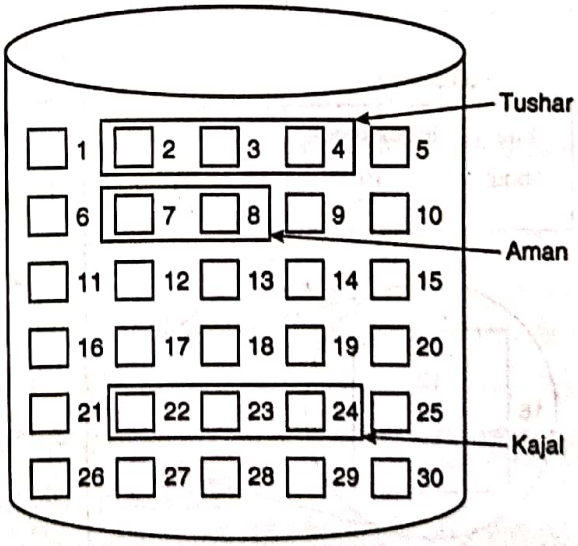


Fig. 6.5

2. Linked allocation method : Linked allocation solves all problems of contiguous allocation. With linked allocation, each file is a linked list of disk blocks and in linked list consists of blocks where each block divided in two parts. First part contains the information and second part contains address of next blocks.

The directory contains a pointer to the first and last block of the file. Each block contains a pointer to the next block. These pointers are not made available to the user.

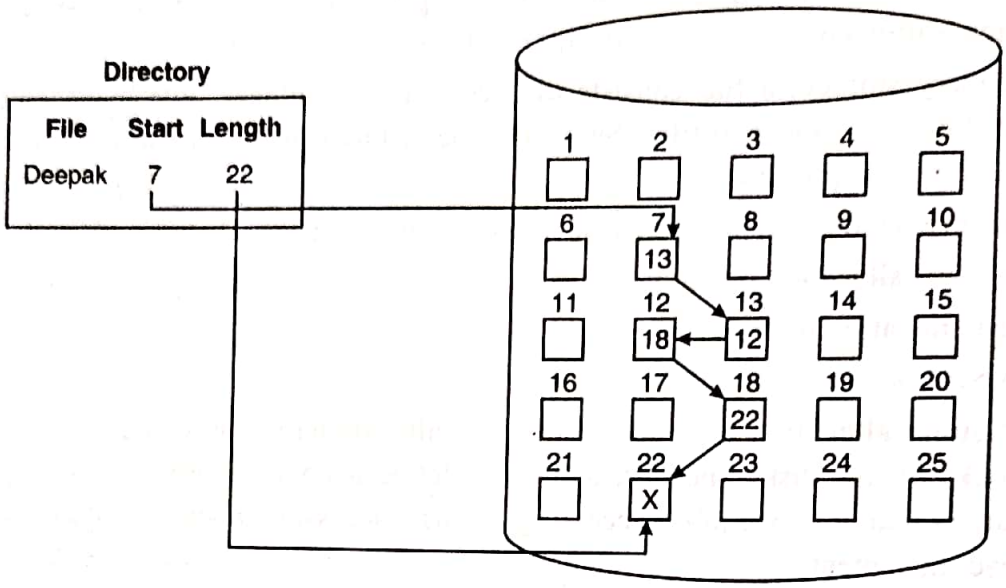


Fig. 6.6

Example : In above example, a file (Deepak) of five blocks might start at block 7 and continue at block 13, then block 12, then block 18 and finally block 22.

3. Indexed allocation method : Linked allocation solves the external-fragmentation and size declaration problems of contiguous allocation. However, in the absence of a FAT (file-allocation table), linked allocation can not support efficient direct access, since the pointers to the blocks are scattered with the blocks themselves all over the disk and must be retrieved in order.

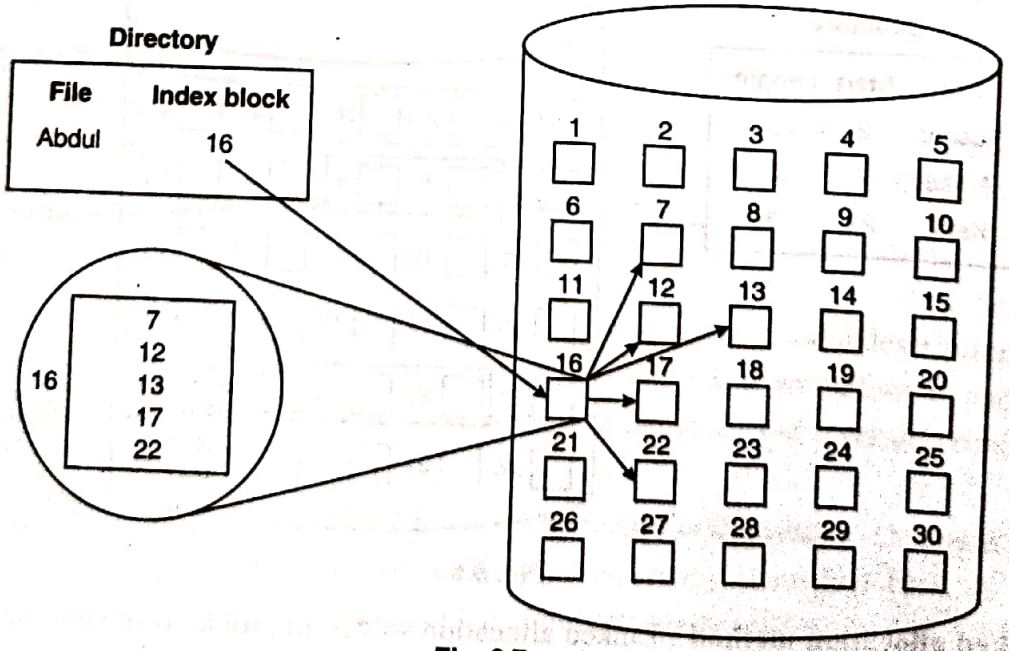


Fig. 6.7

Indexed allocation solves this problem by bringing all the pointers together into one location : the index block.

Each file has its own index block, which is an array of disk-block addresses. The directory contains the address of the index block. To find and read the i^{th} block, we use the pointer in the i^{th} index block entry.

SUMMARY

File : A file is a named collection of related information that is recorded on secondary storage. A file is a collection of similar records.

File management : File management is one of the basic and important features of operating system. Operating system is used to manage files of computer system.

Sequential access : Information in the file is processed in order, one record after the other.

Direct access : Direct access allows random access to any file block.

Indexed access : Records are only accessed through their indexes.

Directory system : A directory contains information about the files, including attributes, location and ownership.

Simple file system : The simple file system handles a single application at any given time, it implements no user concept, does not support protection among files.

Basic file system : The basic file system works directly with the device drivers in terms of retrieving and storing raw block of data without any consideration.

Logical file system : Logical files do not contain data. They contain a description of records that are found in one or more physical files.

Physical file system : Physical file contains the actual data that is stored on the system.

Contiguous allocation method : Contiguous allocation requires that each file occupy a set of contiguous blocks on the disk.

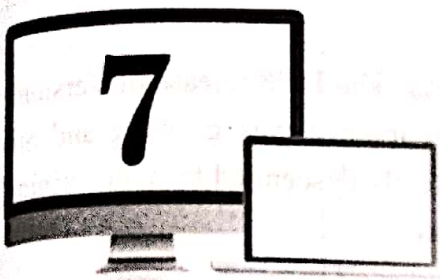
Linked allocation method : Linked allocation solves all problems of contiguous allocation. With linked allocation, each file is a linked list of disk blocks.

Indexed allocation method : In indexed allocation method each file has its own indexed block which is an array of disk block addresses.

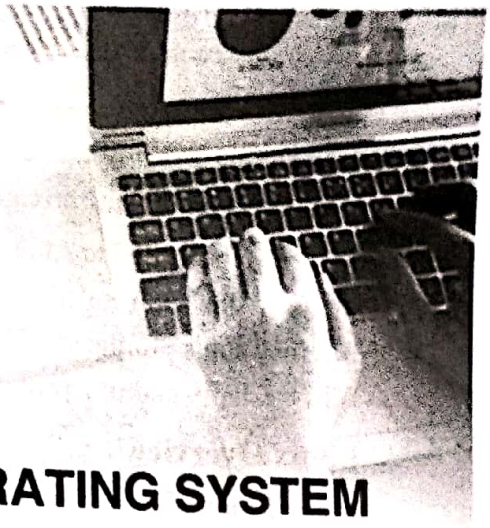
EXERCISE

1. What is file? List and explain attributes of files.
2. Explain concept of file structure.
3. Discuss the file management functions.
4. What are the various types of file operation which can be performed on a file with examples?
5. Explain the file system types.

6. Explain direct file access method.
7. What is sequential and indexed access method?
8. What is directory system?
9. Explain different types of directory structure.
10. What is tree structured directories? Explain with diagram.
11. Discuss various file allocation method in detail.
12. Compare the contiguous allocation and indexed allocation.
13. Draw diagram and explain linked file allocation.



7



LINUX OPERATING SYSTEM

History of Linux and Unix, Linux Overview, Structure of Linux, Linux releases, Open Linux, Linux System Requirements, Linux Commands and Filters: mkdir, cd,rmdir,pwd, ls, who, whoami, date, cat,chmod, cp, mv, rm,pg,more, pr, tail, head, cut, paste, nl, grep, wc, sort, kill, write, talk,mseg,wall, merge,mail, news
Shell: concepts of command options, input, output,redirection,pipes, redirecting and piping with standard errors, Shell scripts,vi editing commands

7.1. History of Linux

Linux operating system was created in the early 1990s by Finish software engineer Linus Tor Valds and the Free Software Foundation. While still a student at the university of Helsinki, Tor Valds started developing Linux to create a system similar to Minix, a Unix operating system. In 1991 he released version 0.02; version 1.0 of Linux Kernel, the core of the operating system, was released in 1994.

Linux grew throughout the 1990s because of the hobbyist developers. Although linux is not as user friendly as the popular microsoft windows and Mac Os operating system, it is an efficient and reliable system that rarely crashes. Combined with Apache, an open-source web server, linux accounts for more than a third of all servers used on the internet.

Because it is open source, and thus modifiable for different uses, Linus is popular for systems as diverse as cellular telephones and super computers.

It is useful to make the distinction between the linux kernel and a linux system. The linux kernel is an entirely original piece of software developed from scratch by the linux community. The linux system, as we know it today, includes a multitude of components, some written from scratch others borrowed from other development projects, and still others created in collaboration with other teams.

History of Unix

The first version of unix was created in 1969 by Kenneth Thompson and Dennis Ritchie, system engineers at AT & T's Bell Labs. It went through many revisions and gained in popularity untill 1977, when it was first made commercially available by interactive system corporation.

At the same time a team from the University of California at Berkeley was working to improve UNIX. In 1977 it released the first Berkeley software distribution, which became know as BSD over time this won favour through innovations such as the *C* shell.

Meanwhile the At & T version was developing in different ways. The 1978 release of Version 7 included the Bourne Shell for the first time. By 1983 commercial interest was growing and Sun Microsystems produced a UNIX work station. System *V* appeared, directly descended from the original AT & T Unix and the prototype of the more widely used variant today.

Linux Overview

Linux is similar to other operating system you may have used before, such as Windows, Mac Os (formely OSX) or ios like other operating system. Linux has a graphical interface, and the same types of software you are accustomed to, such as Word Processors, photo editors, video editors and so on.

Linux also is different from other operating systems in many important ways, first and perhaps most importantly, linux is open source software. The code used to create linux is free and avialable to the public to view, edit and for users with the appropriate skills-to contribute to.

Linux is also different in that, although the core pieces of the Linux operating system are generally common, there are many distributions of Linux, which include different software options. This means that Linux is incrediply customizable, because not just applications, such as word processors and web browsers, can be swapped out.

Depending on which user survey you look at, between one-and two thirds of the webpages on the internet are generated by servers running Linux.

Structure of linux

Linux is one of popular version of Unix operating system. It open source as its source code is freely available. It is free to use. Linux was designed considering Unix compatibility. Its functionality list is quite similar to that of Unix.

Components of Linux System : Linux operating system has primarily three components.

1. Kernel : Kernel is the core part of linuxs. It is responsible for all major activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware. Kernel provides the required abstraction to hide low level hardware details to system or application programs.

2. System Library : System libraries are special functions or programs using which application programs or system utilies accesses kernel's features. These libraries implement most of the functionalities of the operating system and do not requires kernel module code access rights.

3. System Utility : System utility programs are responsible to do specialized, individual level tasks.

Kernel Mode vs User Mode : Kernel component code executes in a special privileged mode called kernel mode with full access to all resources of the computer. This code represents a single process, executes in single address space and do not require any context switch and hence is very efficient and fast. Kernel runs each processes and provides system services to processes, provides protected access to hardware to processes.

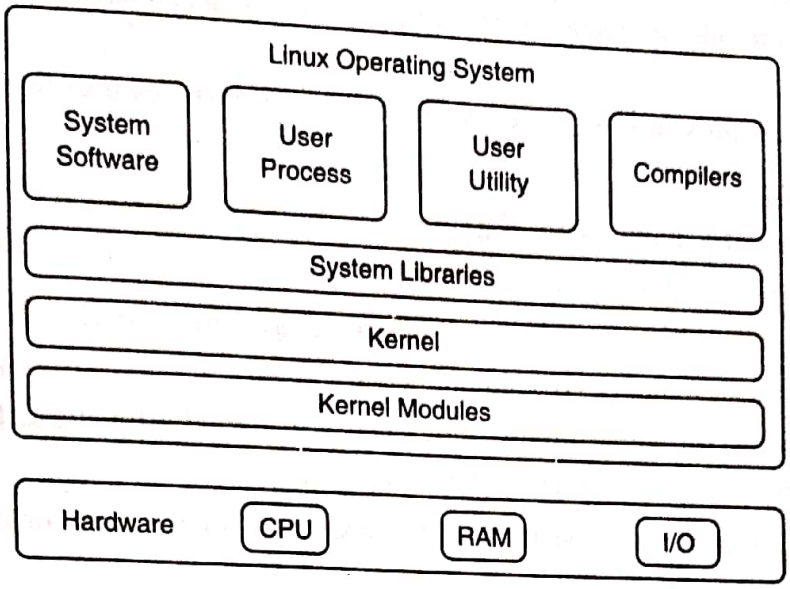


Fig. 7.1

Suport code which is not required to run in kernel mode is in system library. User programs and other system programs works in user mode which has no access to system hardware and kernel code.

7.2. Basic Features

Following are some of the important features of linux opearting system.

Portable : Portability means software can works on different types of hardware in same way. Linux kernel and application programs support their installation on any kind of hardware platform.

Open source : Linux source code is freely available and it is community based development project. Multiple teams work in collaboration to enhance the capability of linux operating system and it is continuously evolving.

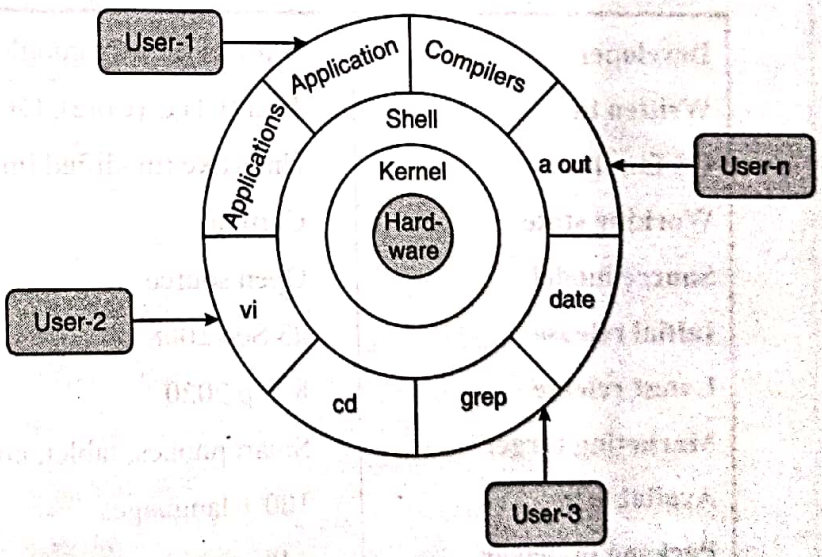


Fig. 7.2

Multi user : Linux is a multiuser system means multiple users can access system resources like memory |RAM| application programs at same time.

Multiprogramming : Linux is a multiprogramming system means multiple applications can run at same time.

Hierarchical file system : Linux provides a standard file structure in which system files/user files are arranged.

Shell : Linux provides a special interpreter program which can be used to execute commands of the operating system.

Security : Linux provides user security using authentication features like password.

Linux Architecture

The architecture of a linux system consists of the following layers :

Hardware Layer : Hardware consists of all peripheral devices (RAM | HDD | CPU etc.)

Kernel : It is the core component of operating system, interacts directly with hardware, provides low level services to upper layer components.

Shell : An interface to kernel, hiding complexity of kernel's functions from users. The Shell takes commands from the user and execute kernel's functions.

Utilities : Utility programs that provide the user most of the functionalities of an operating systems.

7.3. Linux Releases

List of all linux operating system releases, listed alphabetically. This operating system by linux list includes all linux based operating system releases. This linux operating system list contains all new and old releases of the linux operating system. Some of these might are top rated versions of linux, while others may be horribly outdated.

1. Android :

Developer	Various (mostly google and the open hand set alliance)
Written in	Java (UI) C (Core), C++ and others
OS family	Unix-like (modified linux kernel)
Working state	Current
Source model	Open source
Initial release	23 Sep 2008
Latest release	8 Sep 2020
Marketing target	Smart phones, tablet, computers, smart TVs
Available in	100 + languages
Package manager	APK-based
Platforms	64 and 32 bit ARM, X 86 and X 86 – 64
Kernel type	Linux kernel
Default user interface	Graphical
License	Apache license 2.0 for user space software GNU GPL V2 for the linux kernel.

2. Arch Linux :

Developer	Levente Polyak and others.
OS family	UNIX-like
Working state	Current
Source model	Open source
Initial release	11 March 2002
Latest release	01-01-2021
Marketing target	General purpose
Package manager	Pacman, libal pm
Platforms	X 86 – 64 / 686 (unofficial) ARM (unofficial)
Kernel type	Monolithic (Linux)
Default user interface	Command line interface
License	Free software

3. Debian GNU/Linux :

Developer	The debian project
OS family	Unix-like
Working state	Current
Source model	Open source
Initial release	Sep 1993
Latest release	5 Dec 2020
Available	75 languages
Package manager	APT, dpkg
Platforms	X86 –64, arm 64 i 386, mipsel
Kernel type	Linux kernel
Default user interface	GNOME on DVD, XFCE on CD
License	DFSG-Compatible licenses

4. Gentoo linux :

Developer	Gentoo foundation
OS family	Unix-Like (LINUX)
Working state	Current

Source model	Open source
Initial release	26 July 2000
Package manager	Portage
Platforms	IA-32, X86-64, 1A-64, PA-RISC (HPPA), Power PC 32/64, SPARC 64 bit
Kernel type	Monolithic (Linux)
License	Free software
Default user interface	Xfce, LXQt, KDE, GNOME, Fluxbox, i3 Sway.

5. Kubuntu :

Developer	Community driven previously blue system
OS family	Linux (Unix-like)
Working state	Current
Source model	Open source
Initial release	8 April 2005
Latest release	22 October 2020
Available in	Multilingual
Package manager	dpkg
Platforms	IA-32, X86-64, ARM
Kernel type	Monolithic (linux)
Default user interface	KDE plasma desktop, Plasma mobile
License	Free software

6. Mandriva linux :

Developer	Mandriva
OS family	Unix like
Working state	Discontinued
Source model	Open source
Initial release	23 July 1998
Latest release	28 Aug. 2011
Available in	Multilingual

Package manager	Urpmi, rpmdrake
Platforms	amd 64, i686, i586, i486m, i386, sparc 64, MIPS, Xbox
Kernel type	Monolithic (Linux)
Default user interface	KDE Plasma desktop
License	Various free software, plus proprietary binary blobs

7. PC Linux OS :

Developer	Bill Reynolds aka "Texstar"
OS Family	Linux (Unix-like)
Working state	Current
Source model	Open source
Initial release	October 2003
Latest release	15 October 2020
Package manager	Apt-get (RPM)
Platforms	amd 64
Kernel type	Monolithic (Linux)
Default user interface	KDE plasma Desktop. MATE. XFCE
License	Various

8. PS2 Linux :

Developer	Sony computer entertainment
OS family	UNIX like
Working state	Discontinued
Initial release	2002
Platforms	Playstation 2 SCPH – 50000 and earlier
Kernel type	Monolithic (Linux kernel)
Default user interface	Window Maker

9. Red hat Linux :

Developer	Red hat
OS family	Linux (unix like)
Working state	Discontinued

Source model	Open source
Initial release	13 May 1995
Final release	31 March 2003
Package manager	RPM package manager
Kernel type	Monolithic (Linux)
License	Various
Succeeded by	Red hat enterprise Linux, federa

10. Sabayon Linux :

Developer	Fabio Erculiani and Team
OS family	Linux (Unix like)
Working state	Current
Source model	Mixed
Inital release	28 November 2005
Latest release	31 March 2019
Package manager	Entropy/Portage
Platforms	X86-64, previously also IA-32
Kernel type	Monolithic Kernel (Linux)
Default user interface	GNOME, KDE, XFce, Mate, Flubox.
License	Various; Mainly GPL

11. Slackware :

Developer	Patrick Volkerding
OS family	Linux (Unix-like)
Working state	Current
Source model	Open source
Initial release	17 July 1993
Latest release	30 Jun 2016
Available in	Multilingual
Package manager	Pkgtool, slackpkg
Platforms	IA-32, X86-64, ARM

Kernel type	Monolithic (Linux)
Default user interface	CLI
License	GNU General Public License.

12. Ubuntu :

Developer	Conorical Ltd.
OS family	Linux
Working state	Current
Source model	Open source
Initial release	20 Oct. 2004
Latest release	22 Oct. 2020
Available in	More than 55 language by LOCOs
Package manager	GNOME software, APT, dpkg, snappy, flatpak
Plateforms	IA-32, X86-64 ARM 64, ARMhf PPC 64 le (Power 8) S 390x
Kernel type	Linux kernel
Default user interface	GNOME
License	Free software

7.4. Open File in Linux

There are various ways to open a file in a linux system. It is a fairly straight forward process to view the contents of a file, but if you are a new user, it may brother you. It is not as easy as opening a file in notepad. From the linux terminal, you must have some exposures to the linux basic commands. There are some commands such as Cat, ls that are used to read files from the terminal.

In linux, we can display various file formats such as text file, audio files, video, image, doc, pdf, or any other file contents. Following are some useful ways to open a file from the terminals.

1. Open file using cat command : It simply prints the file content of the terminal. It provides many option to make it more specific. To go in depth with cat command.

C at < file name >

Let's create a file to understand how to open a file. Execute the below command.

C at Test. txt

This is a test file.

2. Open file using less command : The less command allows us to view one page at a time to display the file content, execute the less command as follows.

Less Test txt

The above command will display the file content as a page at a time.

3. Open file using more command : The linux more command is also used to display the file content. As the less command automatically adjusts the height and width of the terminal window, it cuts the contents as the width of the terminal.

More Test. txt

4. Open file using nl command : The nl command displays the file content with the line number. It is almost the same as the cat command. Difference is that it prepends line numbers while displaying the output in the terminal.

***nl* Test. text**

5. Open file using head command : We can display the file content by using the head command, but it is slightly different than others. It displays the first part of files via standard input.

head < file name >

Linux system requirements

Before installing data connecton linux, your system must meet the minimum requirements provided in this section.

Operating system and software requirements :

- 64 bit U.S. English edition of one of the following
- Red hat Enterprise linux 7 (all releases)
- Red hat Enterprise linux 6 (all releases)
- Java Runtime environment 8 (embedded)

Hardware requirements :

- 64 bit processor, X 86-64, 2 GHz or faster
- At least 2 GB free hard drive space to install data connect.
- 8 GB high speed RAM

Standalone engine requirements : In addition to the minimum requirements for your operating system, each additional running standalone engine on the same machine require the following.

- 2 GB high speed RAM
- 1–2 available processor cores.

Note : These minimum requirements assumes that the data connect engine has full access to available resources and is executing in a dedicated physical or virtual server.

Data Connect development studio requirement : The development environment is a 64 bit desktop application used to generate and test integration design artifacts interactively. It requires :

- 1 GB RAM for Eclipse
- Recent dual (quad is recommended) core processor.

Note : Since the data connect developement studio embeds the standalone engine, you do not have to install the standalone engine.

7.5. Linux Commands and Filters

It's recommended to use a command-line interface (CLI) because its more powerful and effective tasks that require a multi-step process through GUI can be done in a matter of seconds by typing commands in to CLI.

1. Mkdir command : Use mkdir command to make a new directory. If you type mkdir music it will create a directory called music.

mkdir filename

To generate a new directory inside another directory, use this linux basic command.

mkdir music/New file

Use the *P* (parents) option to create a directory in between two existing directories, for example.

mkdir – P Music/2020/New file

will create the ''new 2020'' file.

2. cd command : The cd command changes your current directory. In other words, it moves you to a new place in the file system. Let's say you're in

/home/username/documents and you wants to go to photos, a subdirectory of document. To do so, simply type the following command.

Cd photos

3. rmdir command : To remove a directory, use the command.

rmdir directoryname

If you need to delete a directory, use the rmdir command. However, rmdir only allows you to delete empty directories.

4. pwd command : Use the pwd command to find out the path of the current working directory (folder) you're in. The command will return an absolute (full) path, which is basically a path of all the directories that starts with a forward slash (/).

/home/username

5. Is command : The Is command is used to view the contents of a directory. By default, this command will display the contents of your current working directory.

If you want to see the content of other directories, type Is and the directory's path. For example :

Is/home/username/documents

to view the content of documents

6. Who command : While working on the command line, you might want to know more about logged in users. The who command shows who all are logged in.

who [option] [file name]

7. Whoami command : As its name suggests,the whoami command print the user name of the effective user ID. In other words, it displays the name of the currenty logged in user.

whoami [option]

8. Date command : date commands displays and sets the system date and time.

date [option] ..... [+ format]

date [– u|-utc | .... universal] [MMDDHHmm] [CC] YY] [.SS]]

9. Cat command : Cat (concatenate) command is very frequently used in linux. It reads data from the file and gives their content as output. It helps us to create, view concatenate files.

- **To view a single file command**

Cat file name

- **To view multiple files command**

Cat file 1 file 2

10. Chmod command : The chmod command is used to change the access mode of a file.

Chmod [reference] [operator] [mode] file name

The references are used to distinguish the users to whom the permissions apply.

11. CP command : CP stands for copy. This command is used to copy files or group of files or directory. It create an exact image of a file on a disk with different file name. CP command require at least two file names in its arguments.

CP [Option] source destination

CP [Option] Source directory

CP [Option] Source 1 source 2 source 3

First and second syntax is used to copy source file to destination file or directory. Third syntax is used to copy multiple source files.

Example : CP scenery. jpg /home/ username/pictures.

12. mv command : The primary use of the mv command is to move files, although it can also be used to rename files.

The arguments in mv are similar to the cp command. You need to type mv, the file name and the destination's directory.

mv file. txt/home/username/documents

to rename files the linux command is

mv oldname. ext newname. ext.

13. rm command : The rm command is used to delete directories and the contents within them. If you only want to delete the directory as an alternative to rmdir-

rm [option]....file

rm-r

Be very careful with this command and double check which directory you are in. This will delete everything and there is no undo.

14. Pg command : Pg displays a text file, pausing after each "page" (the height of the terminal screen). After each page, a prompt is displayed. The user may the either press the newline key to view the next page or one of the key described below.

- If no file name is given on the command line, Pg reads from standard input. If standard output is not a terminal, Pg acts like cat but procedes each file with its name if there is more than one.

- If input comes form a pipe, pg stores the data in a buffer file while reading to make navigation possible.

Pg myfile text

Display the first screenful of the contents of text file myfile. Text and a prompt (":") pressing the return key displays the next page, or any of the command listed above may be entered to otherwise navigate the file.

15. More command : The more command is quite similar to the cat command, as it is used to display the file content in the same way that the cat command does. The only difference between both commands is that, in case of larger files, the more commands displays screenful output at a time.

In more command, the following keys are used to scroll the page.

Enter key : To scroll down page by line.

Space bar : To move to the next page.

b key : To move to the previous page

/ key : To search the string

Syntax : More < file name>

16. Pr command : Pr command is used to prepare a file for printing by adding suitable footers, headers, and the formatted text. pr command actually adds 5 lines of margin both at the top and bottom of the page.

Pr [option] [file name]

17. Tail command : The tail command is similar to the head command. The difference between both commands is that it displays the last ten lines of file content. It is useful for reading the error message.

tail < file name>

18. Head command : The head command is used to display the content of file. It displays the first 10 lines of a file.

head < file name>

19. Cut command : The cut command is used to select a specific column of a file. The '–d' option is used as a delimiter and it can be a space (''), a slash (/), a hyphen (–), or any thing else. And, the '–f' option is used to specify a column number.

Cut – d–f<file name>

20. Paste command : Paste command is one of the useful commands in unix or linux operating system. It is used to join files horizontally (parallel merging) by outputing lines consisting of lines from each file specified, separated by tab as delimiter, to the standard output. The paste command writes corresponding lines from files.

Paste <option> <file name>

21. nl command : nl command is used for numbering lines, accepting input either from a file or from STDIN. It copies each specified file to STD out, with line numbers appended before the lines.

nl < option> <file name>

22. Grep command : The grep is the most powerful and used filter in a linux system. The 'grep' stands for ''global regular expression print''. It is useful for searching the content from a file. Generally, it is used with the pipe.

grep <search word>

23. WC comaand : The WC command is used to count the lines words, and characters in a file.

WC < file name>

24. Sort command : The sort comand is used to sort files in alphabetical order.

sort < file name >

25. Kill command : The most common command to terminate a process is kill command. You need to know the PID (process id) of the process you want to terminate.

Linux system allows us to kill a process in various ways, such as kill a process by its name or process id (PID).

Kill command sends signal to the specified process for sending signal either signal name or signal number can be used.

Kill < Signal PID>

26. Write command : The write command allows you to communicate with other users by copying lines from your terminals to theirs.

When you run the write command the user you are writing to gets a message of the format.

Message from your name @ your host on yourtly at hh : mm: ..

Any further lines you enter will be copied to the specified user's terminal. If the other user wants to reply, they must run write as well.

Syntax : Write user [*tty*]

User : The user to write to.

***tty* :** The specific terminal to write to, if the user is logged in more than one session.

27. Talk command : The talk command provides a text chat interface which lets you communicate in real time with other logged in users.

Talk is a visual communication program which copies from your terminal to that of another user, much like an instant messenger service.

talk person [*tty* name]

Person : If you want to talk to someone on your machine, the person is just the person's login name. If you want to talk to a user on another host then person is of the form **'user@host'**

tty name : If you want to talk to a user who is logged in more than once. The tty name argument may be used to indicate the appropriate terminal name, where tty name is of the form

'tty xx'

28. Mesg command : The mesg command allows you control write access to your terminal by other users.

The write command allows other users to send a message to your terminal session; the mesg command is used to toggle these message on or off.

mesg [*n*/*y*]

n **:** Prevents the display of terminal messages from other users. This option is like using a "do not disturb" sign.

y **:** Allows messages to be displayed on your screen.

29. Wall command : The wall command write a message simultaneously to all other logged in users.

Wall displays the contents of file or, by default, its standard input, on the terminals of all currently logged in users. The command will cut any lines that are over 79 characters to new lines.

wall [–*n*] [–*t* Timeout] [File]

–*n*, – – nobanner : Suppress banner

–*t*, – – time out : Write time out to terminals in seconds.

TIMOUT must be positive integer. Default value is 300 seconds.

Example : Sudo wall message. txt

Using the sudo command to run wall as the superuser, sends the contents of message. text to all users.

30. Merge command : Merge command is used for three way file merge.

merge [option] file 1 file 2 file 3

Merge incorporates all changes that lead from file 2 to file 3 into file 1. The result ordinarily goes into file 1, merge is useful for combining separate changes to an original. Suppose file 2 is the original and both file 1 and file 3 are modifications of file 2. Then merge combines both changes.

31. Mail command : The mail command is used to send emails from the command line

mail - S "subject < recipient address>

7.6. News Shell

Linux shell are a lot more powerful than the windows command line because they function as a scripting language as well, with a complete set of tools. Multiple shell can be installed on a system and it is possible to quickly switch between them. Every shell comes with its own syntax and scripting features.

Concpet of command option :

Most linux commands have several options.

1. Linux command option are used to control the output of linex command.
2. For almost all linux commands the options are prefixed with a – (dash).
   For example – the following linux command runs the Is comamnd with *I* (*el*) option. The I stands for long and it gives you longer listing of files and directories in the linux system.

   ]#Is-I

3. Linux command option can be combined.
   The Is commands is used to list the directories and files in the linux file system. It has an I (for long) option and on a (for all) option. The a option shows "all" files, including hidden files.

4. Linux command option can be combined without a space between them and with a single (dash)

   The following command is a faster way to the I and a options and gives the same output as the linux command show above.

   ]#Is-Ia

5. The letter used for a linux command option may be different from one command to another. For example, the *r* option of one command may not provide the same output as the *r* option for another command. You can learn how to use linux commands easily by watching linux video tutorials.

7.7. Redirection

Redirection is a feature in linux such that when executing a command, you can change the standard input devices. The basic workflow of any linux command is that it takes an input and give an output.

- The standard input (stdin) device is the keyboard.
- The standard output (stdout) device is the screen.

With redirection, the above standard input/output can be changed.

Input redirection :

<input redirection

The '<' symbol is used for input (STDIN) redirection.

Example : The mail program in linux can help you send emails from the terminal.

You can type the content of the email using the standard device keyboard. But if you want to attach a file to email you can use the input redirection operation in the following format.

Mail - S"subject" to address <filename

Output redirection :

> output Redirection

The '>' symbol is used for output (STDOUT) redirection.

ls-al>listing

Here the output of command ls-al is redirected of file "listing" instead of your screen.

Use the correct file name while redirecting command output to a file. "If there is an existing file with the same name, the redirected command will delete the contents of that file and then it may be overwritten".

If you do not want a file to be overwritten but want to add more content to an existing file, then you should use '>>' operator.

Cat music.mp3>/Ram/audio.

The cat command reads the file music. mp3 and sends the output to/Ram/audio which is the audio device.

7.8. Pipes

In linux, the pipe command lets you sends the output of one command to another. Piping, as the term suggests, can redirect the standard output, input, or error of one process to another for further processing.

The syntax for the pipe or unnamed pipe command is the | character between any two commands.

Command-1/command-2/..../command-N

Here, the pipe cannot be accessed via another session, it is created temporarily to accommodate the execution of command-1 and redirect the standard output. It is deleted after successful execution.

A named pipe can last until as long as the system is up and running or until it is deleted. It is special file that follows the FIFO (first in, first out) mechanism. It can be used just like a normal file. You can write to it, read from it, and open or close it. To create a named pipe, the command is :

mk fifo <pipe-name>

This creates a named pipe file that can be used even over multiple shell sessions.

Redirecting and Piping the Standard Errors

When you execute commands, an error could possibly occur. You may give the wrong number of arguments, or some kind of system error could take place. When an error occurs, the system issues an error message. Usually such error messages are displayed on the screen, along with the standard output. Linux distinguishes between standard output and error messages however. Error messages are placed in yet another standard byte stream, called the standard error.

Example : The cat command is give as its argument the name of a file that does not exist, myintro. In this case, the cat command simply issues and error.

Cat my intro

Cat : my intro not found.

Because error message are in separate data stream from the standard output, error messages still appear on the screen for you to see even if you have redirected the standard output to a file.

Next example : The standard output of the cat command is redirected to the file mydata. However, the standard error, containing the error messages, is still directed to the screen.

Cat myintro>mydata

Cat: myintro not found

You can redirect the standard error, as you can the standard output. This means you can save your error messages in a file for future reference. This is helpful if you need a record of the error messages.

Redirection of the standard error relies on a special feature of shell redirection. You can reference all the standard byte streams in redirection operations with numbers. The number 0, 1 and 2 reference the standard input, standard, and standard error respectively. By default, an output redirection, >, operates on the standard output, 1. You can modify the output redirection to operate on the standard error, however, by preceding the output redirection operator with the number 2.

Example : The cat command again will generate an error. The error message is redirected to the standard byte stream represented by the number 2. The standard error.

Cat nodata 2> my errors

Cat my errors

Cat nodata not found

Shell scripts

Usually shells are interactive that mean, they accept command as input from users and execute them. However some time we want to execute a bunch of commands routinely, so we have type in all commands each time in terminals.

As shell can also take commands as input from file we can write these commands in a file and can execute them in shell to avoid this repetitive work. These files are called shell scripts or shell programs. Shell scripts are similar to the batch file in MSDOS. Each shell script is saved with. Sh file extension. **myscript.sh.**

A shell script have syntax just like any other programming language. If you have prior experience with any programming language like python, C/C++ etc. It would be very easy to get started with it. A shell script comprises following elements :

- **Shell keywords :** If, else, break etc.
- **Shell commands :** Cd, ls, echo, pwd, touch, etc.

Functions

- **Control flow :** if then else, case and shell loops etc.

Need of shell scripts :

- To avoid repetitive work and automation
- System admins use shell scripting for routine backups
- System monitoring
- Adding new functionality to shell.

7.9. VI editor

The VI editor is the most popular and classic text editor in linux family. Below, are some reasons which make it a widely used editor.

1. It is available in almost all linux distributions.
2. It works the same across different platforms and distributions.
3. It is user friendly. Hence, millions of linux user love it and use it for their editing needs.

VI editing commands

i — insert at cursor
a — write after cursor
A — write at the end of line
ESC — terminate insert mode
u — undo last change
U — undo all changes to the entire line

o — open a new line
dd — delete line
3 dd — delete 3 lines
D — delete contents of line after the curvor
C — delete contents of line after the curvor and insert new text.
dw — delete word
4 dw — delete 4 words
cw — change word
X — delete character at the curvor
r — replace character
R — overwrite characters from curvor onward
s — substitute one character under curvor continue to insert
S — Substitute entire line and begin to insert at the beginning of line
~ — change case of individual character.

SUMMARY

Linux is modern, free operating system based on unix standards. It has been designed to run efficiently and reliably on common PC hardware, it also runs on a variety of other platforms. It provides a programming interface and user interface compatible with standard unix systems and can run a large number of UNIX applications including an increasing number of commercially supported applications.

A complete linux system includes many components that were developed independently of linux.

Kernel : Kernel is the core part of linux. It is responsible for all majar activities of this operating system. It consists of various modules and it interacts directly with the underlying hardware.

System library : System libraries are special function or programs using which application programs or system utilies accesses kernel's features.

System utility : System utility programs are responsible to do specialized, individual level tasks.

Shell : An interface to kernel, hiding complexity of kernel's function from users. The shell takes commands from the user and execute kernel's functions.

Redirection : Redirection is a feature in linux such that when executing a command, you can change the standard input/output devices.

Linux is a multiuser system, providing protection between processes and running multiple processes according to a time sharing schedular. Newly created processes can share selective parts of their execution environment with their parent processes, allowing multithreaded programming.

EXERCISE

1. What is Linux? Explain the structure of Linux.
2. What is difference between UNIX and Linux?
3. What is Linux kernel?
4. What are the basic components of Linux?
5. What are the basic features of Linux?
6. What is redirection?
7. Define shell.
8. Write about 10 Linux commands with syntax.
9. Explain the Redirecting and Piping the standard errors.
10. What is shell scripts and also explain the need of shell scripts?
11. What is VI editor? Write about VI editing commands.