GOVERNMENT OF TAMILNADU
DIRECTORATE OF TECHNICAL EDUCATION
CHENNAI – 600 025

STATE PROJECT COORDINATION UNIT

Diploma in Computer Engineering
Course Code: 1052

M – Scheme

e-TEXTBOOK
on
OPERATING SYSTEMS
for
III Semester COMPUTER ENGINEERING

Convener for COMPUTER ENGG Discipline:
Mr.D.ARUL SELVAN
HOD/POST DIPLOMA IN COMPUTER APPLICATIONS
THIAGARAJAR POLYTECHNIC COLLEGE
SALEM

Team Members for Industrial Instrumentation:

1. Mrs. P.SARADHA
   HOD/COMPUTER ENGG,
   GOVERNMENT POLYTECHNIC COLLEGE
   KRISHNAGIRI -635 001

2. Mrs. B.SANTHI MEENA
   LECTURER /COMPUTER ENGG
   GOVERNMENT POLYTECHNIC COLLEGE
   KRISHNAGIRI -635 001

3.Mrs.V.BHUVANESWARI
   LECTURER /COMPUTER ENGG
   GOVERNMENT POLYTECHNIC COLLEGE
   DHARMAPURI-635 205

Validated by
Mrs.P.S.NEELAYATHAKSHI
LECTURER(SEL.GRADE)/COMPUTER ENGG
VALIVALAM DESIKAR POLYTECHNIC COLLEGE,
NAGAPATTINAM-611001

UNIT- I

INTRODUCTION TO OPERATING SYSTEMS

OBJECTIVES

- To describe the basic concept of operating system.
- To give an overview of generations of operating system.
- To explore several types of operating systems
- To provide information about major components of operating system
- To describe the various services of operating system .
- To know about the various ways of structuring an operating system.

INTRODUCTION

An operating system is a program that manages a computer's hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware. This unit provides a general overview about the major components of operating system and its functions . Also gives information about various services of an operating system provides, how they are provided, how they are debugged, and about the various methodologies of operating system structures.

BASICS OF AN OPERATING SYSTEM

An Operating System (OS) is an interface between a computer user and computer hardware. An operating system is a software which performs all the basic tasks like file management, memory management, process management, handling input and output, and controlling peripheral devices such as disk drives and printers. An OS is basically an intermediate agent between the user and the computer hardware.

Some popular Operating Systems include Linux, Windows, OS X, VMS, OS/400, AIX, z/OS, etc.

1.1.1 DEFINITION

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

Following are some of important functions of an operating System.

* Memory Management * Processor Management
* Device Management * File Management
* Security * Control over system performance
* Job accounting * Resources allocator & Manager
* Controlling programs to prevent errors and improper computer use.
* Interrupt driver.

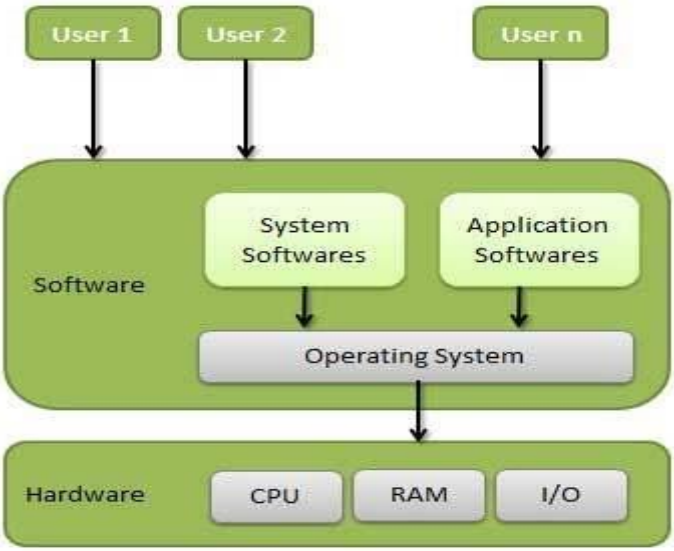


Fig – 1.1 operating system

Users and processes access the computers resources through the operating system.

1.1.2 GENERATIONS OF OPERATING SYSTEM

Operating systems have a series of revolutionary changes called generations. In computers hardware, generations have been marked by major advances in componentry from vacuum tubes (first generation) to transistors (Second generation), to integrated circuitry (Third generation), to large scale and very large. scale integrated circuitry (Forth generation). The successive hardware generations have each been accompanied by dramatic reductions in costs, size, heat, emission, and energy consumption, and try dramatic increases in speed and storage capacity.

1. The 1940's - First Generations
2. The 1950's - Second Generation
3. The 1960's - Third Generation
4. Fourth Generation (1971-Present) Microprocessors
5. Fifth Generation (Present and Beyond) Artificial Intelligence

1.1.2.1 The First generation

The earliest electronic digital computers had no operating systems. Machines of the time were so primitive that programs were often entered one bit at time on rows of mechanical switches (plug boards). Programming languages were unknown . The operating systems of the first generation were designed to smooth the transition between jobs. Before the systems were developed a great deal of time was lost between the completion of one job and the initiation of the next. This was the beginning of batch processing systems in which jobs were gathered in groups or batches. Once a job running, it had total control of the machine. An each job terminated, control was returned to the operating system.This mode of operation is called serial processing.Disadvantages of serial processing

1.Scheduling

2.Setup Time

1.1.2.2 The Second generation

By the early 1950's, the routine had improved somewhat with the introduction of punch cards. The General Motors Research Laboratories implemented the first operating systems in early 1950's for their IBM 701. The system of the 50's generally ran one job at a time. These were called single-stream batch processing systems because programs and data were submitted in groups or batches.

The second generation of OS was characterized by the development of shared systems with multiprogramming and beginnings of multiprocessing. In multiprogramming systems several user programs are in main storage at once and the processor is switched rapidly between the jobs. In multiprocessing systems several processors are used on a single computer system to increase the processing power of the machine.

1.1.2.3 The Third Generation

The systems of the 1960's were also batch processing systems, but they were able to take better advantage of the computer's resources by running several jobs at once. So operating systems designers developed the concept of multiprogramming in which several jobs are in main memory at once;

For example, on the system with no multiprogramming, when the current job paused to wait for other I/O operation to complete, the CPU simply sat idle until the I/O finished. The solution for this problem that evolved was to partition memory into several pieces, with a different job in each partition. While one job was waiting for I/O to complete, another job could be using the CPU.

Another major feature in third-generation operating system was the technique called spooling (simultaneous peripheral operations on line). In spooling, a high-speed device like a disk interposed between a running program and a low-speed device involved with the program in input/output.

Another feature present in this generation was time-sharing technique, a variant of multiprogramming technique, in which each user has an on-line (i.e., directly connected) terminal. Time-sharing systems were developed to multiprogramming large number of simultaneous interactive users.

1.1.2.3.1 Multiprogramming

Sharing the processor, when two or more programs reside in memory at the same time, is referred as **multiprogramming**. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

An OS does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the jobs in the memory.
- Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle, unless there are no jobs to process.

1.1.2.3.2 Time Sharing

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer **system** at the same **time**. **Time-sharing** or multitasking is a logical extension of multiprogramming. Processor's **time** which is **shared** among multiple users simultaneously is termed as **time-sharing**.

1.1.2.4 Fourth Generation

With the development of LSI (Large Scale Integration) circuits, chips, operating system entered in the system entered in the personal computer and the workstation age. Microprocessor technology evolved to the point that it become possible to build desktop computers as powerful as the mainframes of the 1970s. Two operating systems have dominated the personal computer scene: MS-DOS, written by Microsoft, Inc. for the IBM PC and other machines using the Intel 8088 CPU and its successors, and UNIX, which is dominant on the large personal computers using the Motorola 6899 CPU family.

1.1.3 TYPES OF OPERATING SYSTEM

1.1.3.1 Mainframe Operating system

Mainframe computers are large computers.This supports a large number of terminals.So this can be used by any number of users at a time.The main objective of the mainframe operating system is to process many jobs at a time.

A mainframe operating system provides

i) Transaction processing

Transaction system is for processing transactions .

Example: Bank Transaction, Airline Reservation

ii) Batch processing

Batch system processes routine tasks

Example: Super Market

iii) Timesharing services

Timesharing system can be used for multiple remote users to run tasks at the same time.

Example: IT Companies.

Example of a mainframe operating system is OS/390.

1.1.3.2 Desktop operating system

Desktop Operating SystemA desktop operating system is one that is intended for a desktop computer (Unless you are a network administrator or something like that, you probably use a desktop computer.) These OSes usually come with things that one would probably use at a desk. For example, Windows sometimes comes with Microsoft Office pre-installed.

The control program in a user's machine (desktop or laptop). Also called a "client operating system," Windows is the overwhelming majority while the Macintosh comes second. There are also several versions of Linux for the desktop.

1.1.3.3 Multiprocessor operating system

Each CPU has its own operating system. The Memory divide into an many partition as there are CPU's and give each CPU has its own private memory and its own private copy os operating system.

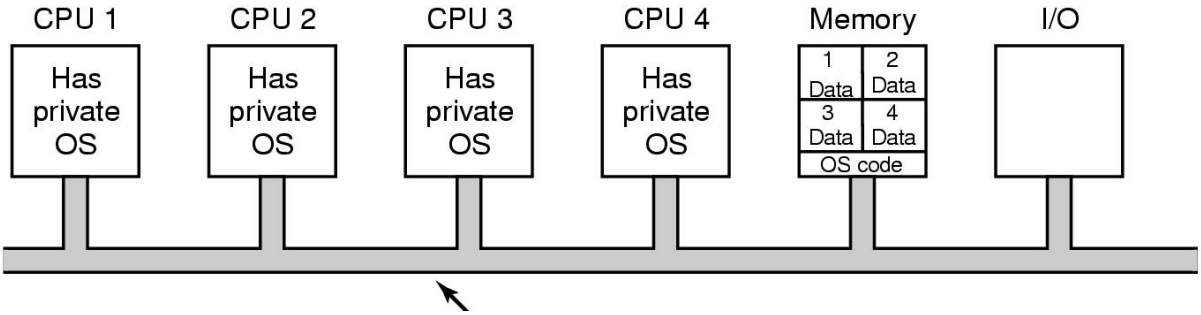


Fig.1.2 Multiprocessor operating system

Advantages

1. Increased Throughput.
2. Reduced cost
3. High Reliabilty.

1.1.3.4 Distributed operating System

Distributed systems use multiple central processors to serve multiple real-time applications and multiple users. Data processing jobs are distributed among the processors accordingly.

The processors communicate with one another through various communication lines (such as high-speed buses or telephone lines). These are referred as **loosely coupled systems** or distributed systems. Processors in a distributed system may vary in size and function. These processors are referred as sites, nodes, computers, and so on.

The advantages of distributed systems are as follows

- With resource sharing facility, a user at one site may be able to use the resources available at another.
- Speedup the exchange of data with one another via electronic mail.
- If one site fails in a distributed system, the remaining sites can potentially continue operating.
- Better service to the customers.
- Reduction of the load on the host computer.
- Reduction of delays in data processing.

1.1.3.5 Clustering Operating System

Clustering is the use of multiple computers, typically PCs or UNIX workstations, multiple storage devices, and redundant interconnections, to form what appears to users as a single highly available system. Cluster computing can be used for load balancing as well as for high availability.

Computer cluster technology puts clusters of systems together to provide better system reliability and performance. Cluster server systems connect a group of servers together in order to jointly provide processing service for the clients in the network.

Cluster operating systems divide the tasks amongst the available servers. Clusters of systems or workstations, on the other hand, connect a group of systems together to jointly share a critically demanding computational task. Theoretically, a cluster operating system should provide seamless optimization in every case.

At the present time, cluster server and workstation systems are mostly used in High Availability applications and in scientific applications such as numerical computations

Advantages

1. Reliability is high
2. Portable
3. Load balancing

1.1.3.6 Multiprogramming Operating system

Sharing the processor, when two or more programs reside in memory at the same time, is referred as **multiprogramming**. Multiprogramming assumes a single shared processor. Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

The following figure shows the memory layout for a multiprogramming system.

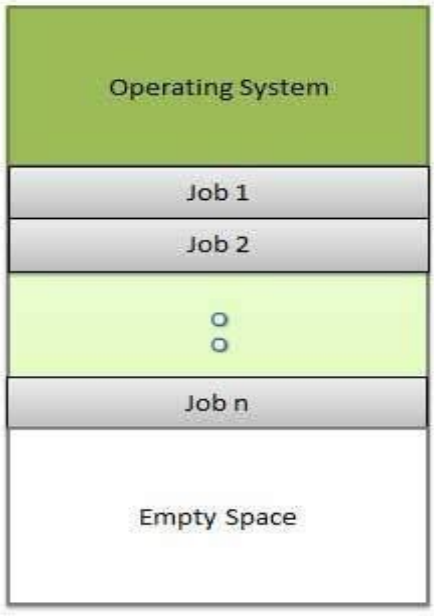


Fig 1.3 Multiprogramming

An OS does the following activities related to multiprogramming.

- The operating system keeps several jobs in memory at a time.
- This set of jobs is a subset of the jobs kept in the job pool.
- The operating system picks and begins to execute one of the jobs in the memory.
- Multiprogramming operating systems monitor the state of all active programs and system resources using memory management programs to ensures that the CPU is never idle, unless there are no jobs to process.

Advantages

- High and efficient CPU utilization.
- User feels that many programs are allotted CPU almost simultaneously.

Disadvantages

- CPU scheduling is required.

1.1.3.6. REAL TIME OPERATING SYSTEM

A real-time system is defined as a data processing system in which the time interval required to process and respond to inputs is so small that it controls the environment. The time taken by the system to respond to an input and display of required updated information is termed as the response time. So in this method, the response time is very less as compared to online processing.

There are two types of real-time operating systems.

Hard real-time systems

- Hard real-time systems guarantee that critical tasks complete on time. In hard real-time systems, secondary storage is limited or missing and the data is stored in ROM. In these systems, virtual memory is almost never found.

Soft real-time systems

- Soft real-time systems are less restrictive. A critical real-time task gets priority over other tasks and retains the priority until it completes. Soft real-time systems have limited utility than hard real-time systems. For example, multimedia, virtual reality, Advanced Scientific Projects like undersea exploration and planetary rovers, etc.

1.1.3.7 Embedded Operating System

An embedded operating system is a specialized OS for use in the computers built into larger systems. An embedded system is a computer that is part of a different kind of m achine. Examples include computers in cars, traffic lights, digital televisions, ATMs, airplane controls, point of sale (POS) terminals, digital cameras, GPS navigation systems, elevators, digital media receivers and smart meters, among many other possibilities.

1.1.3.7.1 Definition

Embedded system is application-oriented special computer system which is scalable on both software and hardware. It can satisfy the strict requirement of functionality, reliability, cost, volume, and power consumption of the particular application. With rapid development of IC design and manufacture, CPUs became cheap. Lots of consumer electronics have embedded CPU and thus became embedded systems. For example, PDAs, cellphones, point-of-sale devices, VCRs, industrial robot control, or even your toasters can be embedded system.

.eCos and TinyOS, are examples of Embedded operating system.

eCos -The Embedded Configurable Operating System (eCos) is an open source, royalty free real-time OS intended for embedded applications. The system is targeted at high-performance small embedded systems.

TinyOS has become a popular approach to implementing wireless sensor network software. Currently, over 500 organizations are developing and contributing to an open source standard for Tiny OS.

1.1.3.8 Time-sharing operating systems

Time-sharing is a technique which enables many people, located at various terminals, to use a particular computer system at the same time. Time-sharing or multitasking is a logical extension of multiprogramming. Processor's time which is shared among multiple users simultaneously is termed as time-sharing.

The main difference between Multiprogrammed Batch Systems and Time-Sharing Systems is that in case of Multiprogrammed batch systems, the objective is to maximize processor use, whereas in Time-Sharing Systems, the objective is to minimize response time.

The operating system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time. Computer systems that were designed primarily as batch systems have been modified to time-sharing systems.

Advantages of Timesharing operating systems are as follows

- Provides the advantage of quick response.
- Avoids duplication of software.
- Reduces CPU idle time.

Disadvantages of Time-sharing operating systems are as follows

- Problem of reliability.
- Question of security and integrity of user programs and data.
- Problem of data communication.

1.2. OPERATING SYSTEM COMPONENTS

Common operating system components are

- Process management
- Memory management
- I/O management
- File management
- Protection system
- Networking
- Command interpreter

1.2.1 PROCESS MANAGEMENT COMPONENT

Every program running on a computer then it's background services or applications is a process. Process management is an operating systems way of dealing with running multiple processes. A process needs certain resources including CPU time, memory, files and I/O devices to accomplish its task the OS is responsible for process creation and deletion, process suspension and resumption & provision of mechanisms.

Tasks of process management:-

i. Create, load, execute, suspend resume and terminate processes.
ii. Switch system among multiple processes in main memory.
iii. Provides communication mechanisms so that processes can send data to each other
iv. Allocate / de – allocate resources property to prevent or avoid deadlock situation
v. Deadlock handling.

Depending on the operating system as more processes run either each time slice will become smaller or there will be a longer delay before each process is given a change to run process management involves computing and distributing CPU time as well as other resources most operating systems allow a process to be assigned on priority which affects its allocation of CPU time.

1.2.2 MEMORY MANAGEMENT COMPONENT

Memory management is the most important part of an operating system .

There are two types of memory in a computer system. There are

a. Primary or Main memory
b. Secondary memory

1.2.2.1 Main Memory management

The main memory is a large array of bytes and each byte has its own address. Main memory provides the storage for a program that can be accessed directly by the CPU for its exertion. So for a program to be executed, the primary task of memory management is to load the program into main memory.

Memory management performs mainly two functions as follows :

1. Each process must have enough memory in which it has to execute.

2. The different locations of memory in the system must be used properly so that each

and every process can run most effectively.

Operating system loads the instructions into main memory then picks up these instructions and makes a queue to get CPU time for its execution. The memory manager tracks the available memory locations which one is available, which is to be allocated or de-allocated. It also takes decision regarding which pages are required to swap between the

main memory and secondary memory. This activity is referred as virtual memory management that increases the amount of memory available for each process.

The major activities of an operating system in regard to memory-management are

- Keep track of which part of memory are currently being used and by whom.
- Decide which processes should be loaded into memory when the memory space is free.
- Allocate and de-allocate memory spaces as and when required.

1.2.2.2 Secondary-memory Management

A computer system has several levels of storage such as primary storage, secondary storage and cache storage. But primary storage and cache storage cannot be used as a permanent storage because these are volatile memories and its data are lost when power is turned off Moreover, the main memory is too small to accommodate all data and programs. So the computer system must provide secondary storage to backup the main memory. Secondary storage consists of tapes drives, disk drives, and other media.

The secondary storage management provides an easy access to the file and folders placed on secondary storage using several disk scheduling algorithms.

The four major activities of an operating system in regard to secondary storage management are:

- Managing the free space available on the secondary-storage device .
- Allocation of storage space when new files have to be written .
- Scheduling the requests for memory access.
- Creation and deletion of files.

Another important part of memory management is managing virtual addnesses. If multiple processes are in memory at once they must be prevented from interfering with each other's memory. This is achieved by having separate address spaces. Each process sees the whole virtual address space, typically from address up to the maximum size of virtual memory. The operating system maintains a page table that matches virtual addresses to physical addresses. These memory allocations are tracked so that when a process terminals all memory used by that process can be made available for other processes.

1.2.3 I/O MANAGEMENT COMPONENT

In computing, I/O refers to the communication between an information processing system and the outside world processing system inputs are the signals or data received by the system and outputs are the signals or data sent from it. I/O devices are used by a person to communication with a computer. A keyboard or a mouse may be an input device for a computer, which monitors and printers are considered output devices for a computers.

Tasks of I/O Management:-

i. Disk management functions such as free space management
ii. Storage allocation
iii. Fragmentation removal
iv. Head scheduling

v. Consistent
vi. Convenient software to I/O device interface through buffering/ catching.
vii. Custom drivers for each device.

1.2.4 FILE MANAGEMENT COMPONENT

A file is a collection of related information defined by its creator. Computer can store files on the disk (secondary storage), which provide long term storage. Some examples of storage media are magnetic tape, magnetic disk and optical disk. Each of these media has its own properties like speed, capacity, and data transfer rate and access methods.

A file system is normally organized into directories to make ease of their use. These directories may contain files and other directories. Every file system is made up of similar directories and subdirectories. Microsoft separates its directories with a back slash and its file names aren't case sensitive whereas Unix-derived operating systems (including Linux) use the forward slash and their file names generally are case sensitive.

The main activities of an operating system in regard to file management are creation and deletion of files/ folders, support of manipulating files/ folders, mapping of files onto secondary storage and taking back up of files.

1.2.5 PROTECTION SYSTEM

Protection (or security) is the most demanding feature of an operating system. Protection is an ability to authenticate the users for an illegal access of data as well as system.

Operating system provides various services for data and system security by the means of passwords, file permissions and data encryption. Generally computers are connected through a network or Internet link, allowing the users for sharing their files accessing web sites and transferring their files over the network. For these situations a high level security is expected.

At the operating system level there are various software firewalls. A firewall is configured to allow or deny traffic to a service running on top of the operating system. Therefore by installing the firewall one can work with running the services, such as telnet or ftp, and not to worry about Internet threats because the firewall would deny all traffic trying to connect to the service on that port.

If a computer system has multiple users and allows the concurrent execution of multiple processes, then the various processes must be protected from one another's activities. Protection refers to mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system.

1.2.6 NETWORKING MANAGEMENT COMPONENT

An operating system works as a network resource manager when multiple computers are in a network or in a distributed architecture. A distributed system is a collection of processors that do not share memory, peripheral devices, or a clock. The processors communicate with one another through communication lines called network The communication-network design must consider routing and network strategies, and the problems with network and security.

Most of today's networks are based on client-server configuration. A client is a program running on the local machine requesting to a server for the service, whereas a server is a program running on the remote machine providing service to the clients by responding their request.

1.2.7 COMMAND INTERPRETER

A command interpreter is an interface of the operating system with the user. The user gives commands which are executed by operating system (usually by turning them into system calls). The main function of a command interpreter is to get and execute the user specified command.

Command-Interpreter is usually not a part of the kernel, since multiple command interpreters may be supported by an operating system, and they do not really need to run in kernel mode. There are two main advantages of separating the command interpreter from the kernel.

If you want to change the way the command interpreter looks, i.e., you want to change the interface of command interpreter, then you can do that if the command interpreter is separate from the kernel. But if it is not, then you cannot change the code of the kernel and will not be able to modify the interface.

If the command interpreter is a part of the kernel; it is possible for an unauthenticated process to gain access to certain part of the kernel. So it is advantageous to have the command interpreter separate from kernel.

1.3. OPERATING SYSTEM – SERVICES

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

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

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

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection
- Resource Allocation
- Protection

1.3.1 PROGRAM EXECUTION

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

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

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

1.3.2 I/O OPERATIONS

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

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

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

1.3.3 FILE SYSTEM MANIPULATIONS

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

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

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

1.3.4 COMMUNICATIONS

Operating system performs the communication among various types of processes in the form of shared memory. In multitasking environment, the processes need to communicate with each other and to exchange their information. These processes are created under a hierarchical structure where the main process is known as parent process and the sub processes are known as child processes.

The OS handles routing and connection strategies, and the problems of contention and security. Following are the major activities of an operating system with respect to communication −

- Two processes often require data to be transferred between them
- Both the processes can be on one computer or on different computers, but are connected through a computer network.

- Communication may be implemented by two methods, either by Shared Memory or by Message Passing.

1.3.5 ERROR DETECTION & RECOVERY

Operating system also deals with hardware problems. To avoid hardware problems the operating system constantly monitors the system for detecting the errors and fixing these errors (if found). The main function of operating system is to detect the errors like bad sectors on hard disk, memory overflow and errors related to I/O devices. After detecting the errors, operating system takes an appropriate action for consistent computing.

This service of error detection and error correction cannot be handled by user programs because it involves monitoring the entire computing process. These tasks are too critical to be handed over to the user programs. A user program, if given these privileges; can interfere with the corresponding operation of the operating systems

1.3.6 RESOURCE ALLOCATION

In the multitasking environment, when multiple jobs are running at a time, it is the responsibility of an operating system to allocate the required resources (like as CPU, main memory, tape drive or secondary storage etc.) to each process for its better utilization. For this purpose various types of algorithms are implemented such as process scheduling, CPU scheduling, disk scheduling etc

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

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

1.3.7 ACCOUNTING

Operating system keeps an account of all the resources accessed by each process or user. In multitasking, accounting enhances the system performance with the allocation of resources to each process ensuring the satisfaction to each process

1.3.8 SYSTEM PROTECTION

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

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

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

- The OS provides authentication features for each user by means of passwords.

1.3.9 SYSTEM CALLS

The invocation of an operating system routine. Operating systems contain sets of routines for performing various low-level operations. For example, all operating systems have a routine for creating a directory. If you want to execute an operating system routine from a program, you must make a system call.

This may include hardware-related services (for example, accessing a hard disk drive), creation and execution of new processes, and communication with integral kernel services such as process scheduling. **System calls** provide an essential interface between a process and the **operating system**.

There are 5 different categories of system calls

1. Process control
2. File manipulation
3. Device manipulation,
4. Information maintenance
5. Communication.

1.3.9.1 Process Control

A running program needs to be able to stop execution either normally or abnormally. When execution is stopped abnormally, often a dump of memory is taken and can be examined with a debugger. The list of process control are given below.

- end, abort
- load, execute
- create process, terminate process
- get process attributes, set process attributes
- wait for time
- wait event, signal event
- allocate and free memory

1.3.9.2 File Management

Some common system calls are ***create***, ***delete***, ***read***, ***write***, ***reposition***, or ***close***. Also, there is a need to determine the file attributes – ***get*** and ***set*** file attribute. Many times the OS provides an API to make these system calls.

- create file, delete file
- open, close
- read, write, reposition
- get file attributes, set file attributes

1.3.9.3 Device Management

Process usually require several resources to execute, if these resources are available, they will be granted and control returned to the user process. These resources are also thought of as devices. Some are physical, such as a video card, and others are abstract, such as a file.

User programs *request* the device, and when finished they *release* the device. Similar to files, we can *read*, *write*, and *reposition* the device.

- request device, release device
- read, write, reposition
- get device attributes, set device attributes
- logically attach or detach devices

1.3.9.4 Information Management

Some system calls exist purely for transferring information between the user program and the operating system. An example of this is *time*, or *date*.

The OS also keeps information about all its processes and provides system calls to report this information.

- get time or date, set time or date
- get system data, set system data
- get process, file, or device attributes
- set process, file, or device attributes

1.3.9.5 Communication

These calls are used to exchange information between different processes running in the same computer or between different processes running in different systems connected with each other.

The different communication calls are :

- create, delete communication connection
- send, receive messages
- transfer status information
- attach ordetachremote devices

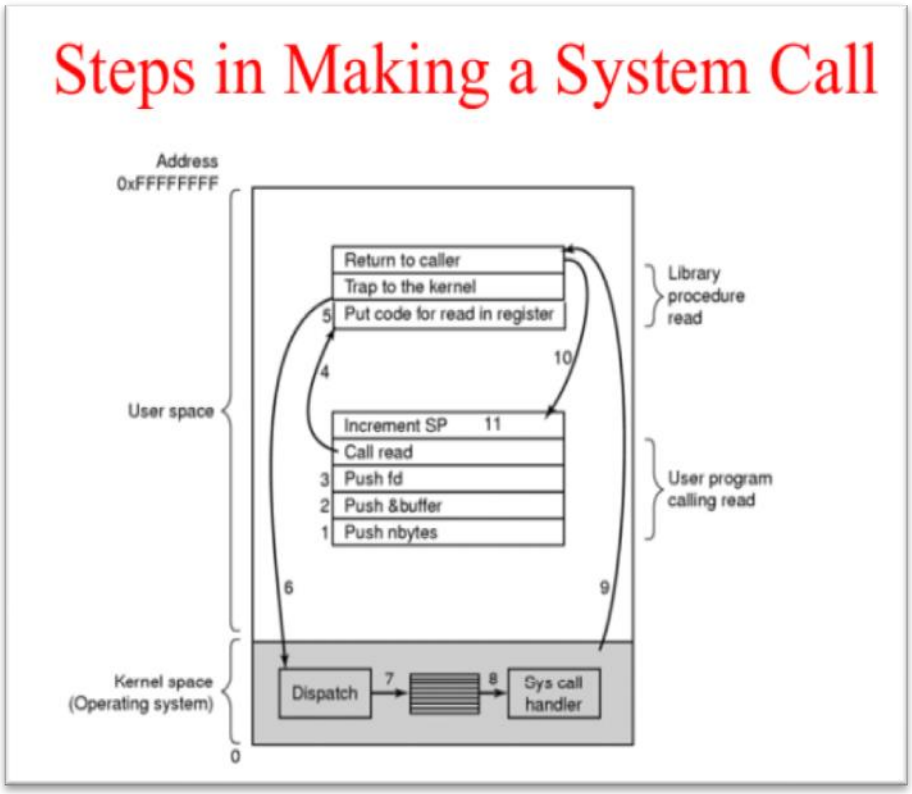
	Windows	Unix
Process Control	CreateProcess() ExitProcess() WaitForSingleObject()	fork() exit() wait()
File Manipulation	CreateFile() ReadFile() WriteFile() CloseHandle()	open() read() write() close()
Device Manipulation	SetConsoleMode() ReadConsole() WriteConsole()	ioctl() read() write()
Information Maintenance	GetCurrentProcessID() SetTimer() Sleep()	getpid() alarm() sleep()
Communication	CreatePipe() CreateFileMapping() MapViewOfFile()	pipe() shmget() mmap()
Protection	SetFileSecurity() InitlializeSecurityDescriptor() SetSecurityDescriptorGroup()	chmod() umask() chown()

Fig 1. 4 SYSTEM CALLS EXAMPLE

1.3.10 SYSTEM CALL EXECUTION

System call is programming interface to the services provided by the OS. It is typically written in a high-level language such as C or C++.

There are 11 steps to making a system call



1. 5 System call Execution

The steps for making a system call

Step 1,2,3: Push parameter on stack
Step 4: Invoke the systemcall
Step 5: Put code for systemcall on register
Step 6: Trap to the kernel
Step 7,8,9,10: since a number is associated with each system call, system call interface invokes/dispatch intended system call in OS kernel and return status of the system call and any return value

Step 11: increment stack pointer C program invoking printf() library call, which calls write() system call.

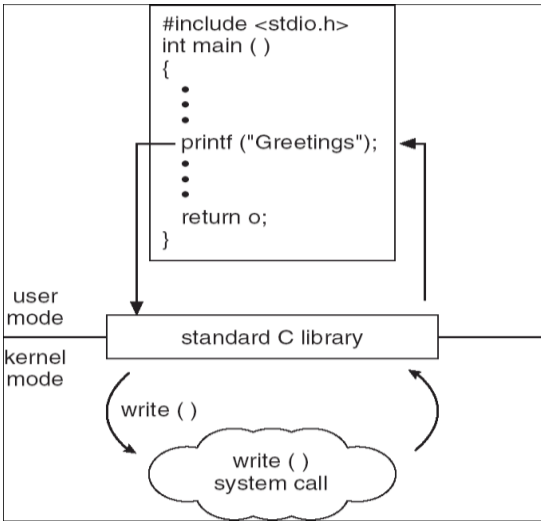


Fig 1.6 **Standard C Library Example**

1.4 OPERATING-SYSTEM STRUCTURES

For efficient performance and implementation an OS should be partitioned into separate subsystems, each with carefully defined tasks, inputs, outputs, and performance characteristics. These subsystems can then be arranged in various architectural configurations:

1.4.1 SIMPLE STRUCTURE

When DOS was originally written its developers had no idea how big and important it would eventually become. It was written by a few programmers in a relatively short amount of time, without the benefit of modern software engineering techniques, and then gradually grew over time to exceed its original expectations. It does not break the system into subsystems, and has no distinction between user and kernel modes, allowing all programs direct access to the underlying hardware.

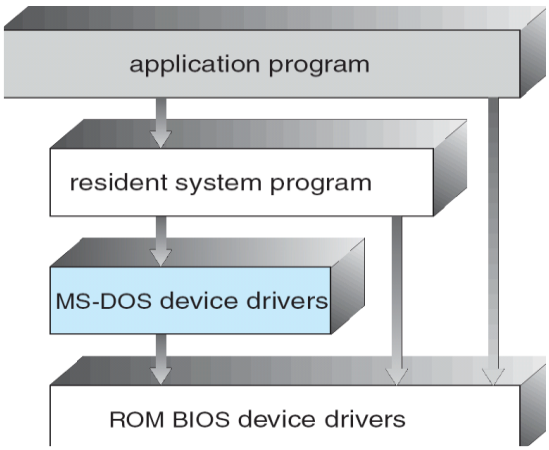


Fig 1.7 Simple structure

1.4.2 LAYERED OPERATING SYSTEM

The components of layered operating system are organized into modules and layers them one on top of the other. Each module provide a set of functions that other module can call. Interface functions at any particular level can invoke services provided by lower layers but not the other way around

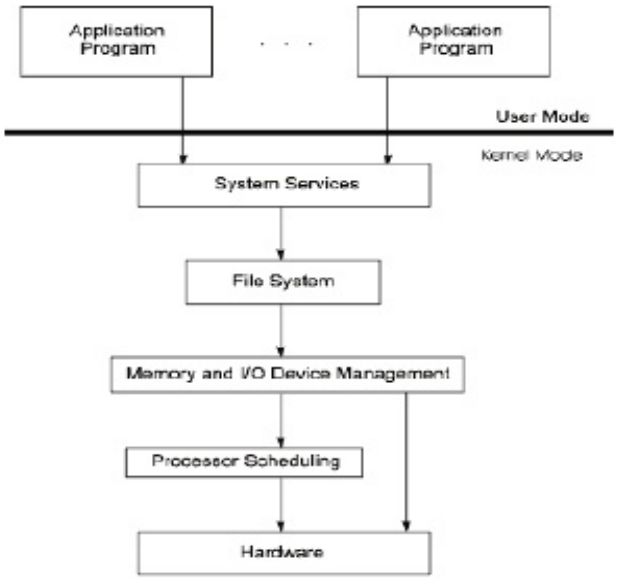


Fig 1.8 Layered Operating System

Advantages

1. More modules and extensible
2. Simple construction, debugging, and verification

Disadvantage

1. Interdependencies make it difficult to cleanly separate functionality among layers
2. less efficient than monolithic designs

1.4.3 MONOLITHIC OPERATING SYSTEM

- Most primitive form of the OS
- Practically no structure
- Characterized by a collection of procedures that can call any other procedure
- All procedures must have a well-defined interface
- Does not allow information hiding (private functions for procedures)
- Services provided by putting parameters in well-defined places and executing a supervisory call.
- Basic structure
    - Main program that invokes requested service procedures

    - Set of service procedures to carry out system calls
    - Set of utility procedures to help the service procedures
- User program executes until
    - program terminates
    - program makes a service request
    - a time-out signal occurs
    - an external interrupt occurs
- Problems with monolithic structure
    - Difficult to maintain
    - Difficult to take care of concurrency due to multiple users/jobs

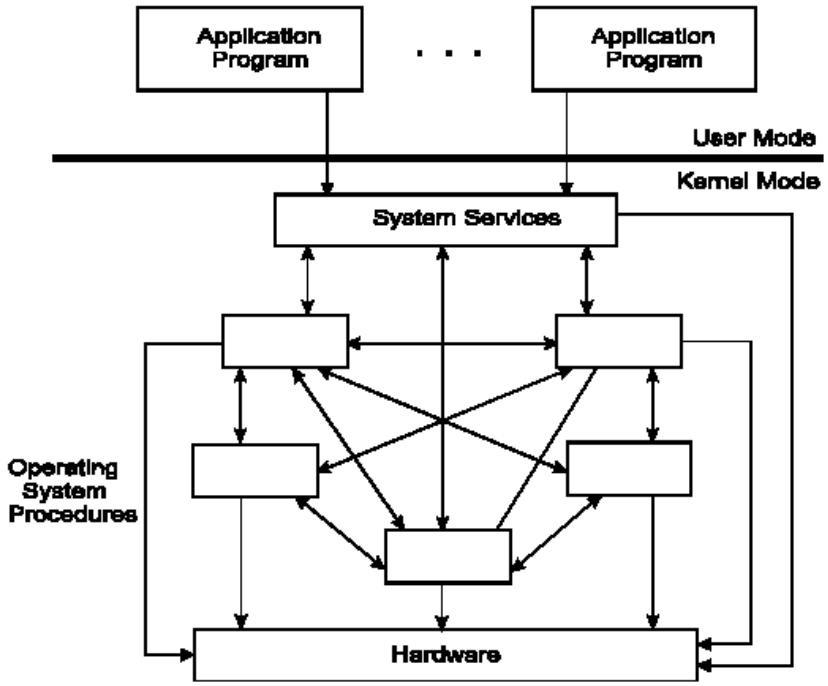


Fig 1.9 Monolithic Operating System

1.4.4 MICROKERNAL OPERATING SYSTEM

- The basic idea behind micro kernel is to remove all non-essential services from the kernel, and implement them as system applications instead, thereby making the kernel as small and efficient as possible.
- Most microkernels provide basic process and memory management, and message passing between other services, and not much more.
- Security and protection can be enhanced, as most services are performed in user mode, not kernel mode.
- System expansion can also be easier, because it only involves adding more system applications, not rebuilding a new kernel.
- Mach was the first and most widely known microkernel, and now forms a major component of Mac OSX.
- Windows NT was originally microkernel, but suffered from performance problems relative to Windows 95. NT 4.0 improved performance by moving more services into the kernel, and now XP is back to being more monolithic.

- Another microkernel example is QNX, a real-time OS for embedded systems

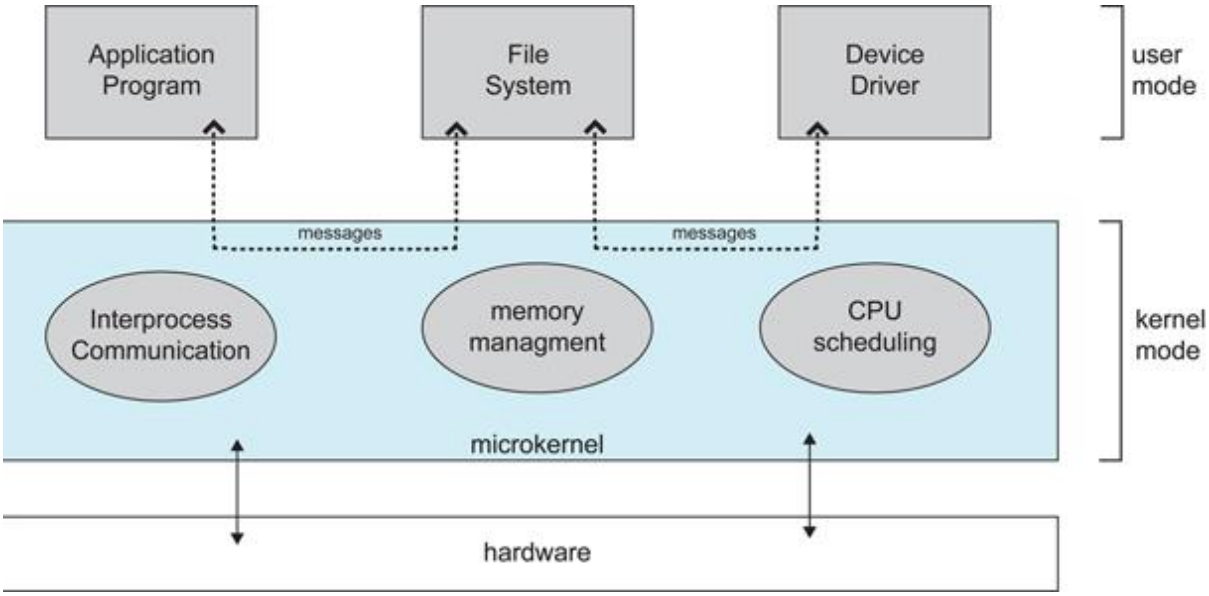


Fig1.10 Microkernel Operating System

Advantages

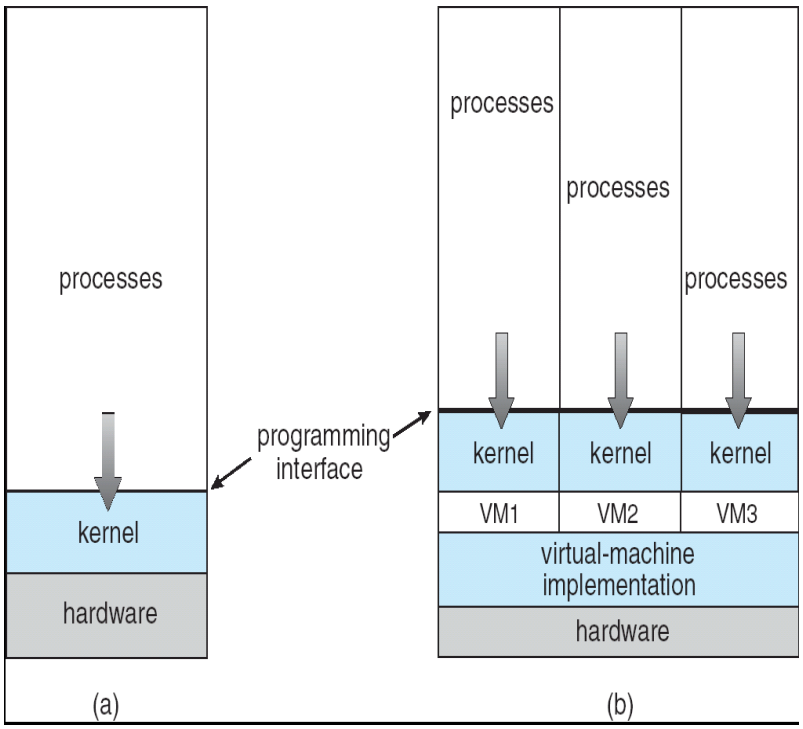
1.Easier to extend
2.Easier to port to new architectures
3.More reliable (less code is running in kernel mode)
4.More secure

Disadvantages

1.No consensus regarding services that should remain in the kernel
2.Performance overhead of user space to kernel space communication

1.4.5 CONCEPT OF VIRTUAL MACHINE

1) A *virtual machine* takes the layered approach to its logical conclusion.
It treats hardware and the operating system kernel as though they were all hardware
2) A virtual machine provides an interface *identical* to the underlying bare hardware
3) The operating system creates the illusion of multiple processes, each executing on its own processor with its own (virtual) memory
4) The resources of the physical computer are shared to create the virtual machines
5) CPU scheduling can create the appearance that users have their own processor
6) Spooling and a file system can provide virtual card readers and virtual line printers
normal user time-sharing terminal serves as the virtual machine operator's console.



(a) Non-virtual machine **(b) virtual machine**

Fig 1.11 Virtual Machine

7) The virtual-machine concept provides complete protection of system resources since each virtual machine is isolated from all other virtual machines. This isolation, however, permits no direct sharing of resources.

8) A virtual-machine system is a perfect vehicle for operating-systems research and development. System development is done on the virtual machine, instead of on a physical machine and so does not disrupt normal system operation.

9) The virtual machine concept is difficult to implement due to the effort required to provide an *exact* duplicate to the underlying machine

1.4.6 BOOTING

Booting is a process or set of operations that loads and hence starts the operating system, starting from the point when user switches on the power button.

Have you ever given it a thought that when you press the power button on your laptop or PC, what happens behind the logo of Windows XP/Vista/Seven or Linux? From the pressing of the power button in the computer. there are more than hundred components/peripherals that are initialized and thousand lines of code is executed during the process of booting.

Boot Sequence As soon as the computer is turned on, the basic input-output system (BIOS) on your system's read-only memory (ROM) chip is "woken up" and takes charge.

1. BIOS first does a power-on self test (POST) to make sure all the computer's components are operational.
2. First, it looks on drive A (unless you've set it up some other way or there is no diskette drive) at a specific place where operating system boot files are located. If there is a diskette in drive A but it's not a system disk, BIOS will send you a

message that drive A doesn't contain a system disk. If there is no diskette in drive A (which is the most common case), BIOS looks for the system files at a specific place on your hard drive.

3. Having identified the drive where boot files are located, BIOS next looks at the first sector (a 512-byte area) and copies information from it into specific locations in RAM. This information is known as the *boot record* or Master Boot Record.
4. It then loads the boot record into a specific place (hexadecimal address 7C00) in RAM.
5. The boot record loads the initial system file (for example, for DOS systems, IO.SYS) into RAM from the diskette or hard disk.
6. The initial file (for example, IO.SYS, which includes a program called SYSINIT) then loads the rest of the operating system into RAM.
7. The initial file (for example, SYSINIT) loads a system file (for example, MSDOS.SYS) that knows how to work with the BIOS.
8. One of the first operating system files that is loaded is a system configuration file (for DOS, it's called CONFIG.SYS). Information in the configuration file tells the loading program which specific operating system files need to be loaded (for example, specific device driver.
9. Another special file that is loaded is one that tells which specific applications or commands the user wants to have included or performed as part of the boot process. In DOS, this file is named AUTOEXEC.BAT. In Windows, it's called WIN.INI.
10. After all operating system files have been loaded, the operating system is given control of the computer and performs requested initial commands and then waits for the first interactive user input.

In order for a computer to successfully boot, its BIOS, operating system and hardware components must all be working properly; failure of any one of these three elements will likely result in a failed boot sequence.

Summary

- An operating system is a program that manages a computer’s hardware and acts as a intermediatry between the computer user and the computer hardware.

- There are five generations in operating system
    - First generation is a serial processing type having scheduling and setup time problem.
    - Second generation is a multiprocessing system i.e several processors on a single computer system
    - Third generation having spooling and multiprogramming technique
    - Fourth generation is a microprocessor technique used in personal computer and in workstation.
    - Fifth generation is the present operating system working in artificial intelligence.

- There are eight types of operating systems available ---- Main frame,Desktop,Multiprocessor,Distributed,Clustering,Multiprogramming,Real time, embedded and time sharing.

- The operating system's major components are process management, memory management, i/o management, File management, Protection system, networking and command interpreter.

- The operating system provide services to both the users and to the programs. example-Program execution, i/o operations, file system manipulation, communication, error detection, resource allocation, protection, system accounting and system calls.

- There are four types of structures are in operating system.-simple, layered, Monolithic, microkernel, concepts of virtual machine and booting operation.

REVIEW QUESTIONS

PART A (2 Marks)

1. Define operating system.
2. What are the functions of an operating system?
3. Define multiprogramming.
4. List the types of operating system.
5. What are the components available in operating system?
6. List the operating system services.
7. List the different structure of operating system.

PART B (3 Marks)

1. Explain clustered operating system.
2. Explain any one operating system.
3. Explain any one component of operating system.
4. Define system call. List the types of system call.
5. Draw the diagram of Microkernel Architecture.
6. What is Booting?

PART C (5 Marks)

1. Briefly explain the generation of operating system.
2. Explain any two operating systems.
3. Explain any two operating system component.
4. Briefly explain the operating system services
5. Explain the different system calls.
6. Explain system call execution.
7. Explain the concept of virtual machine.
8. Define booting. Explain Booting sequence.

REFERENCES

TEXT BOOK

1. Operating_Systems_Internals_and_Design_Principles_7th_Edition_ William Stallings(www.ebook-dl.com)

2. Silberschatz Operating System Concepts 9th edition [www.itkhiladi.com]

WEB SITES

1. http://www.ittc.ku.edu/~kulkarni/teaching/EECS678/slides/chap2.pdf
2. https://www.google.co.in/search?q=microkernel+os+images&biw=1366&bih=634&noj=1&
3. http://www.cs.iit.edu/~cs561/cs450/syscalls/steps.html
4. **www.tutorialspoint.com/operating_system/os_virtual_memory.htm**

UNIT – II
PROCESS MANAGEMENT

OBJECTIVES

At the end of this unit, the student will be able to

- Define the term *process* and explain the relationship between processes and process control blocks.
- Explain the concept of a process state and discuss the state transitions the processes undergo.
- Understand the distinction between process and thread.
- Describe the basic design issues for threads.

- Explain the difference between user-level threads and kernel-level threads.
- Define the term process scheduling
- Explain the various scheduling algorithms.
- Understand the term IPC and synchronization.
- Discuss basic concepts related race conditions and mutual exclusion.
- Define and explain semaphores.
- List and explain the conditions for deadlock.
- Explain the difference between deadlock prevention and deadlock avoidance.
- Understand the approaches to deadlock avoidance, deadlock detection and recovery.

INTRODUCTION

Early computer systems allowed only one program to be executed at a time. This program had complete control of the system and had access to all the system's resources. In contrast, current-day computer systems allow multiple programs to be loaded into memory and executed concurrently. This evolution required firmer control and more compartmentalization of the various programs. These needs resulted in the notion of a process/ which is a program in execution. A process is the unit of work in a modern time-sharing system.

2.1 PROCESS CONCEPT

A Process is a sequential program in execution. ***A process need certain resources such as CPU time, memory, files, and 1/0 devices to accomplish its task***. These resources are allocated to the process either when it is created or while it is executing.

There are two major categories of process are

1. Operating system processes - executing system code
2. User processes - executing user code.

A process includes the process stack, a data section, a heap and text section. The structure of the process in memory is shown in Figure 2.1.

- The text section includes the compiled program code.
- The data section contains global variables.
- The heap is used for dynamic memory allocation.
- The stack is used for local variables, function parameters and return addresses.

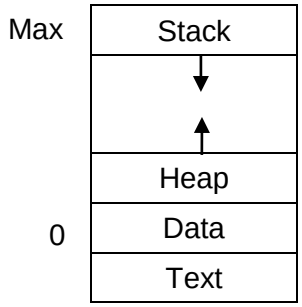


Fig 2.1 Processes in Memory

2.1.1 PROCESS RELATIONSHIP

- When bootstrapping, the kernel manually creates only one process called process 0 or swapper or idle process
- The swapper process initializes the kernel data structure, and creates process1.
- When no processes are ready to run, the kernel executes the swapper to keep the CPU occupied.
- Process1 also initializes some kernel data structures by invoking the kernel init function.
- There is a parent-child relationship among processes.
- The swapper process is the ancestor of all process; it does not have a parent.
- Processes created by the same parent are sibling of one another.
- When the parent dies, all the siblings become children of the init process. The init process monitors executions of its children.
- All processes in the system are organized into a single tree structure. The root of the tree is the init process. Each process in the tree is the original parent holds a pointer to the youngest child.

2.1.2 PROCESS STATES

As a process executes, it changes state. Process state is defined as the current activity of the process. There are ***five states*** of a ***process.*** They are

- ***New***: The process that has just been created.
- ***Ready:*** The process is waiting to be assigned to a process.
- ***Running:*** The process currently being executed.
- ***Waiting***: The process is waiting for some event to occur.
- ***Terminated***: The process has finished execution

2.1.3 PROCESS STATE TRANSITIONS

The possible transition of a process is

1. **Null → New**

   A new process is created to execute a program.
2. **New → Ready**

   The operating system will move a process from the new state to the ready state.
3. **Ready → Running**

   When it is time to select a process to run, the operating system chooses one of the processes in the ready state.
4. **Running → Terminated**

   When the process has completed its execution, then it is terminated by the operating system.
5. **Running → Ready**

   When a process's time quantum expires, then it is taken out from the CPU and placed back in the ready list.
6. **Running → Waiting**

   When a process issues an I/O request, then it is taken out from the CPU and placed back in the waiting list**.**

7. **Waiting → Ready**

   A process in the waiting state is moved to the ready state, when the event for waiting is completed.

Figure 2.2 shows the general structure of the process state transition diagram. Each process may be in one of the following states:

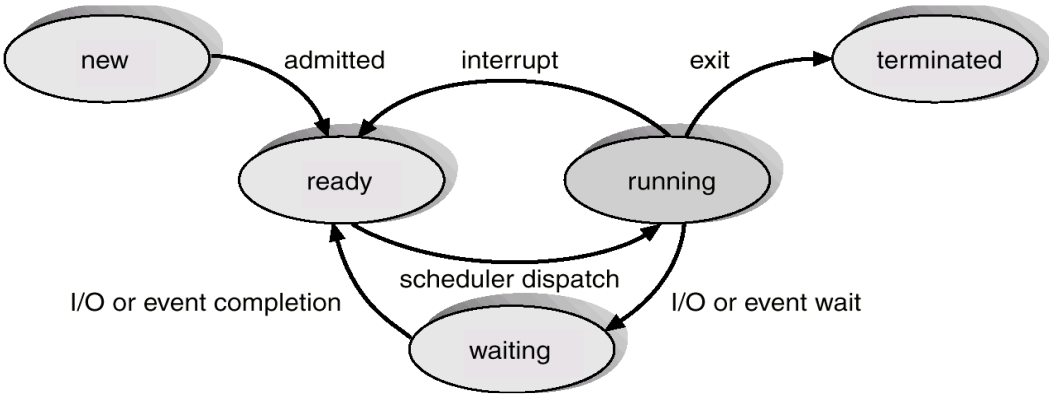


Figure 2.2 Process state transition diagram

Whenever process changes state, the operating system reacts by placing the process's PCB in the list that corresponds to its new state. Only one process can be running on any processor at any instant and many processes may be ready and waiting state.

2.1.4 PROCESS CONTROL BLOCK

Each process contains the Process Control Block (PCB), which stores the following process specific information, as illustrated in Figure 2.3.

Pointer	Process state
Process number	
Program counter	
CPU registers	
Memory limits	
List of open files	
. . .	

Figure 2.3 Process control block (PCB)

Pointer: Pointer points to another process control block.

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

Program Counter: It indicates the address of the next instruction to be executed for this process.

CPU registers: It includes general purpose register, stack pointers, Index registers and accumulators etc.

Memory management information: This information may include the value of base and limit register. This information is useful for de-allocating the memory when the process terminates.

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.

In brief, the ***PCB*** simply ***serves as the repository for any information that may vary from process to process.***

2.1.5 CONTEXT SWITCHING

Whenever an interrupt arrives, the CPU must **save the state** of the currently running process, then switch into kernel mode to handle the interrupt, and then **restore the state of** the interrupted process. When CPU switches to another process, the system must save the state of the old process and load the saved state for the new process is called context switching.

Saving and restoring states involves saving and restoring all of the registers and program counter(s), as well as the process control blocks. So Context switching creates overhead. Context switch time is highly dependent on hardware support.

2.1.6 THREADS

A thread is an execution unit consists of a thread ID, a program counter, a register set and a stack. It shares its code section, data section, and resources with other threads belonging to the same process. A traditional process has a single thread of control. If a process has multiple threads of control, it can perform more than one task at a time. A thread is sometimes called a light weight process. Fig 2.4 shows the single threaded process.

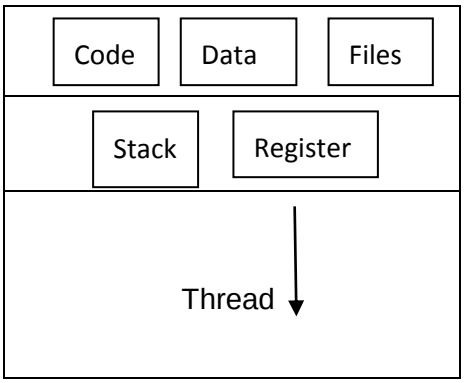


Figure 2.4 Single threaded process

2.1.7 CONCEPT OF MULTITHREADS

When a process has multiple threads it is able to perform multitasking at a time. Most of the modern software applications run with the multithreading concept.

For example, a web browser can have a process running based on the user request that can have a thread that displays images or text while another thread retrieving data from the network.

Threads within the same process can exchange information through their common address space and have access to the shared resources of the process. Threads in different processes can exchange information through shared memory.

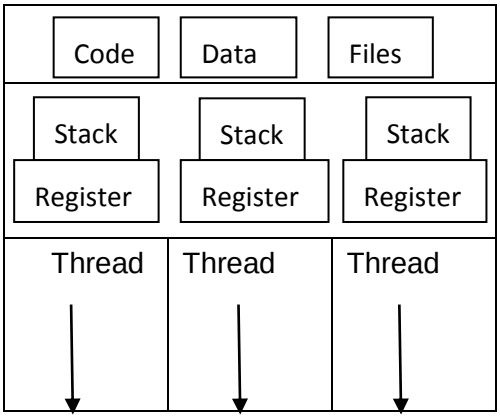


Figure 2.5 Multithreaded processes

Difference between Process and Thread :

S.No	Process	Thread
1.	Process is heavy weight	Thread is light weight
2.	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system
3.	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4.	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.

2.1.8 BENEFITS OF THREADS

1. Takes less time to create a new thread and terminate a thread than a process.
2. Less time required to switch between two threads within the same process.
3. Multithreading increases the responsiveness to the user. It allows a program to continue running even if part of it is blocked.
4. Threads share the memory and the resources of the process to which they belong.
5. Threads allow utilization of multiprocessor architectures to a greater scale and efficiency.

2.1.8 TYPES OF THREAD

Two types of threads are

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

2.1.8.1 User Level Threads

User threads are above the kernel space. User-Level threads are managed entirely by the user level library. In a user level thread, all of the thread management works are done by the user space and the kernel knows nothing about user-level threads. The thread library contains code for creating and destroying threads, data transmission between threads, scheduling thread execution and maintains thread contexts.

Advantages

- Thread switching can all be done without the involvement of the kernel
- User level threads are fast to create and manage.

Disadvantages

- Lack of co-ordination between user level threads and kernel.
- User-level thread requires non-blocking systems call i.e., a multithreaded kernel.

2.1.8.2 Kernel Level Threads

All thread operations are implemented in the kernel and the operating system schedules all threads in the system. Threads managed by operating system are called kernel-level threads. The Kernel performs thread creation, scheduling and management in Kernel space. Kernel threads are generally slower to create and manage than the user threads.

Advantages

- Kernel can simultaneously schedule multiple threads from the same process on multiple processes.
- If one thread in a process is blocked, the Kernel can schedule another thread of the same process.

Disadvantages

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

2.2 PROCESS SCHEDULING

2.2.1 BASIC CONCEPTS

The method of determining which process in the ready state should be moved to running state is known as process scheduling. Process scheduling is an essential part of Multiprogramming operating systems.

In a multiprogramming environment, more than one process resides in main memory at a time. To maximize CPU utilization, when one process has to wait, the operating system takes the CPU away from that process and gives the CPU to another process. Scheduling of this kind is a fundamental operating-system function. Almost all computer resources are scheduled before use. Since CPU is one of the primary computer resources, its scheduling is central to the operating system design.

2.2.2 SCHEDULING OBJECTIVES

The primary objective of the process scheduling is to keep the CPU busy all the time in order to maximize the system performance and to deliver minimum response time for all programs. Some goals that are desirable in all systems are

- Fairness to all processes
- Minimize overhead
- Balance available resources
- Maximize throughput

2.2.3 TYPES OF SCHEDULERS

When more than one process is runnable, the operating system must select one process for execution. The part of the operating system concerned with this decision is called the scheduler, and algorithm it uses is called the scheduling algorithm.

There are three types of schedulers available are

- Long term scheduler
- Short term scheduler
- Medium term scheduler

Long term scheduler

It is also known as job scheduler. It selects processes from the queue and loads them into memory for execution. Primary aim of the Job Scheduler is to maintain a good degree of Multiprogramming.

Short term scheduler

It is also called as CPU scheduler. It selects a process among the processes that are ready to execute and allocates CPU to one of them. The primary aim of this scheduler is to maximize CPU performance. Short term schedulers are faster than long term schedulers.

Medium term scheduler

Medium term scheduler temporarily removes processes from main memory and places them on secondary memory or vice versa. This is commonly referred to as swapping out or swapping in. It reduces the degree of multiprogramming.

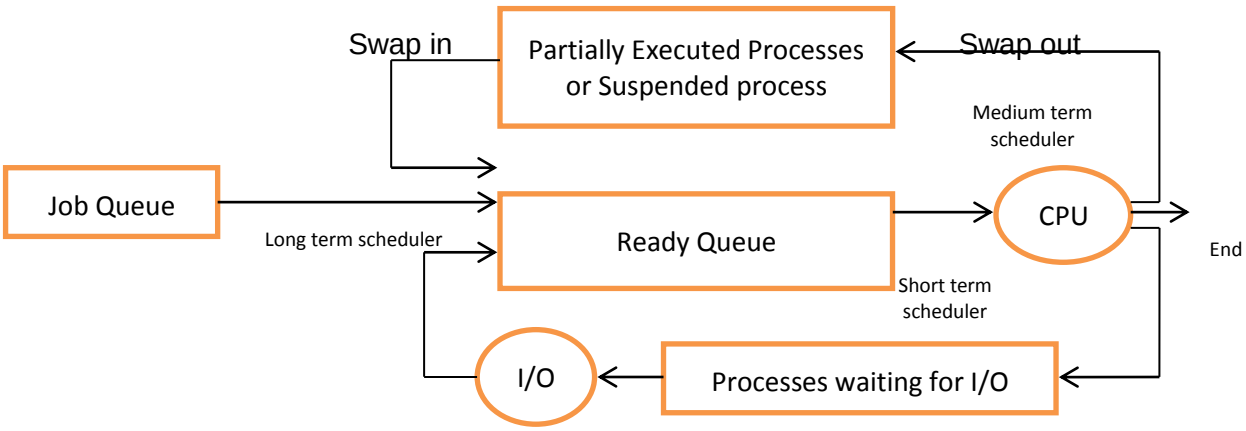


Figure 2.6 Types of schedulers

2.2.4 SCHEDULING CRITERIA

There are various CPU scheduling algorithms are available and they have different properties. Many different criteria have been recommended to compare the scheduling algorithms. They are :

1. ***CPU utilization :***
   The scheduling algorithm should keep the CPU as busy as possible.

2. ***Throughput :***
   It is the total number of processes that are completed per unit time.

3. ***Turnaround time :***
   The interval between the time of submission of the process and the time of completion of the process.

4. ***Waiting time :***
   The average period of time a process spent waiting in the ready queue.
5. ***Response time :***
   Amount of time it takes from when a request was submitted until the first response is produced. It is necessary to increase CPU utilization and throughput and to reduce turnaround time, waiting time, and response time.

2.2.5 SCHEDULING ALGORITHMS

CPU scheduling algorithm may use different criteria for selecting which of the processes in the ready queue is to be allocated the CPU.

Scheduling algorithm may be preemptive or non-preemptive. In preemptive scheduling, a running process may be replaced by a higher priority process at any time. In non-preemptive scheduling, once the CPU has been allocated to a process, the process continues its execution until it terminates or waits for I/O. There are many different CPU-scheduling algorithms.

2.2.5.1 First-Come First-Served scheduling

It is the simplest scheduling algorithm. With this scheme, the process that requests the CPU first is allocated the CPU first. FCFS is a non-preemptive scheduling algorithm.

Let us consider the set of processes that arrive at time 0. The CPU-burst time is given in milliseconds.

Process	Burst time
P1	21
P2	6
P3	3

Gantt chart is a bar chart that shows a particular schedule, including the start and finish times of each of the participating processes.

Figure 2.7 shows the Gantt chart of FCFS scheduling, if the processes arrivein the order p1, p2 & p3.

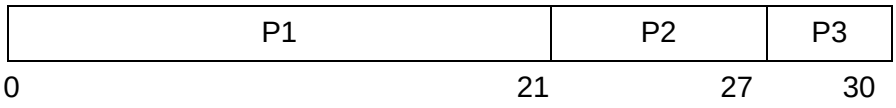


Figure 2.7 Gantt chart

Waiting time:

Process	Waiting time
P1	0
P2	21
P3	27

Average waiting time:

$$\text{Average waiting time} = \frac{Waiting\ times\ of\ all\ processes}{Number\ of\ processes}$$

$$= \frac{0+21+27}{3}$$

= 16 milliseconds

In FCFS policy, the average waiting time is not generally minimum. This may vary substantially if the processes CPU burst times vary greatly. FCFS has relatively low throughput for heavy workload.

2.2.5.2 Shortest-Job-First scheduling

Shortest Job First (SJF) is a scheduling policy that selects the waiting process with the smallest execution time to execute next. If two processes have the same length, FCFS

scheduling is used to break the tie. SJF algorithm may be either preemptive or non-preemptive.

A preemptive SJF algorithm will preempt the currently running process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU-burst.

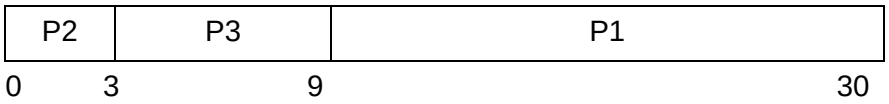
2.2.5.2.1 Non-preemptive SJF

Let us consider the set of process with burst time in milliseconds. Arrival time of the processes is 0 and the processes arrive in the order of p1, p2 and p3

Process	Burst time
P1	21
P2	3
P3	6

.

Gantt chart:



Waiting time:

Process	Waiting time
P1	9
P2	0
P3	3

Average waiting time:

$$\text{Average waiting time} = \frac{9+0+3}{3}$$

$$= \frac{12}{3}$$

$$= 4 \text{ milliseconds}$$

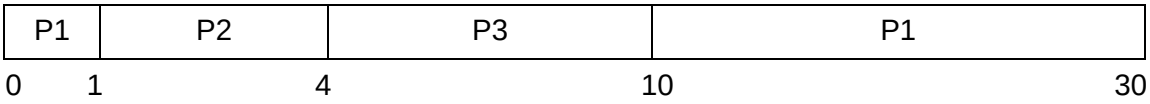
Shortest-Job-First scheduling is an optimal algorithm. It gives the minimum average waiting time for a given set of processes.

2.2.5.2.1 Preemptive SJF

Let us consider the following three processes, with the length of the CPU burst given in milliseconds

Process	Arrival Time	Burst time
P1	0	21
P2	1	3
P3	2	6

If the processes arrive at the ready queue at the times shown in the above table, then the resulting preemptive SJF schedule is as depicted in the following Gantt chart



- Process p1 is started at time 0, since it is the only process in the queue.
- Process p2 arrives at time 1. The remaining time for process p1 (20 milliseconds) is larger than the time required by process p2 (3 milliseconds), so process p1 is preempted and process p2 is scheduled.

- Process p3 arrives at time 2 with burst time 6ms. The burst time of process p3 is larger than process p2. So process p2 continues its execution. After the completion of process p2 process p3 is scheduled since the burst time of process p3 (6 milliseconds) is lesser than process p1.
- Finally process p1 is scheduled.

Waiting time:

Waiting time for process p1: The process p1 is scheduled at time 0 and preempted at time 1. Then it is again scheduled for execution at time10ms. So the waiting time of process p1 is calculated as follows

Waiting time = (Entry time of next scheduling – Exit time of first scheduling) – Arrival time.

Waiting time for p1 = (10 – 1) – 0
= 9 ms

Process p2 and p3 are not preempted during execution. So the waiting time for p2 and p3 are

Waiting time = Entry time of scheduling – Arrival time
Waiting time for p2 = 1 – 1 = 0ms
Waiting time for p3 = 4 – 2 = 2ms

Process	Waiting time
P1	9
P2	0
P3	2

Average waiting time:

$$\text{Average waiting time} = \frac{9+0+2}{3}$$

$$= \frac{11}{3}$$

$$= 3.67 \text{ milliseconds}$$

2.2.5.3 Round-Robin Scheduling

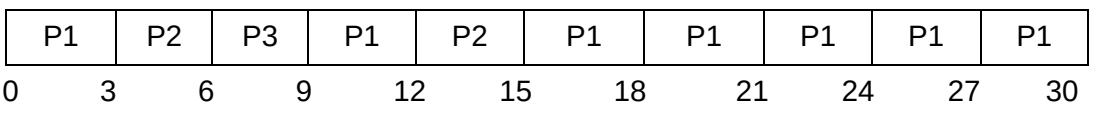
The **round-robin (RR) scheduling algorithm** is designed especially for timesharing systems. It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called a **time quantum** or time slice, is defined. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue and allocating the CPU to each process. Let us calculate the average waiting time for 3 processes with the following table.

Process	Burst time
P1	21
P2	6
P3	3

Let us consider the time quantum of 3 milliseconds. Process P1 gets the first 3 milliseconds. Since it requires another 18 milliseconds, it is preempted after the first time quantum, and the CPU is given to the next process in the queue, process P2. Process P2 is preempted after 3 milliseconds.

The CPU is then given to the next process, process P3. Process P3 completes its execution since its burst time is 3ms. Once each process has received 1 time quantum, the CPU is returned to process P1 for an additional time quantum. The resulting RR schedule is as follows:

Gantt chart:



Waiting Time:

Process P1 = 0 + (9 – 3) + (15 – 12) = 9ms

Process P2 = 3 + (12 – 6) = 9ms

Process P3 = 6ms

Process	Waiting time
P1	9
P2	9
P3	6

Average waiting time:

$$\text{Average waiting time} = \frac{9+9+6}{3}$$

$$= \frac{24}{3}$$

= 8 milliseconds

2.2.6 MULTIPROCESSOR SCHEDULING

Multiprocessor *Operating System* refers to the use of ***two or more central processing units (CPU) within a single computer system.*** These multiple CPUs sharing the computer bus, memory and other peripheral devices. Multiprocessor scheduling refers to a set of procedures and mechanisms built into the operating system to execute the available processes by multiple processors.

On a uniprocessor, scheduling is one dimensional. On a multiprocessor, scheduling is two dimensional. The scheduler has to decide which process to run and which CPU to run it on.

Multiprocessor scheduling is generally complicated for both unrelated processes and related processes. If the processes are unrelated then each process can be scheduled without regard to the other. If all the processes have related to one another then the scheduling is complicated.

2.2.7 TYPES OF MULTIPROCESSOR SCHEDULING

2.2.7.1 Timesharing

This is the simplest scheduling algorithm for dealing with unrelated processes. The unrelated processes which are ready to execute are placed in different queues depending on their priority. Based on the priority, the processes are allocated the available processors.

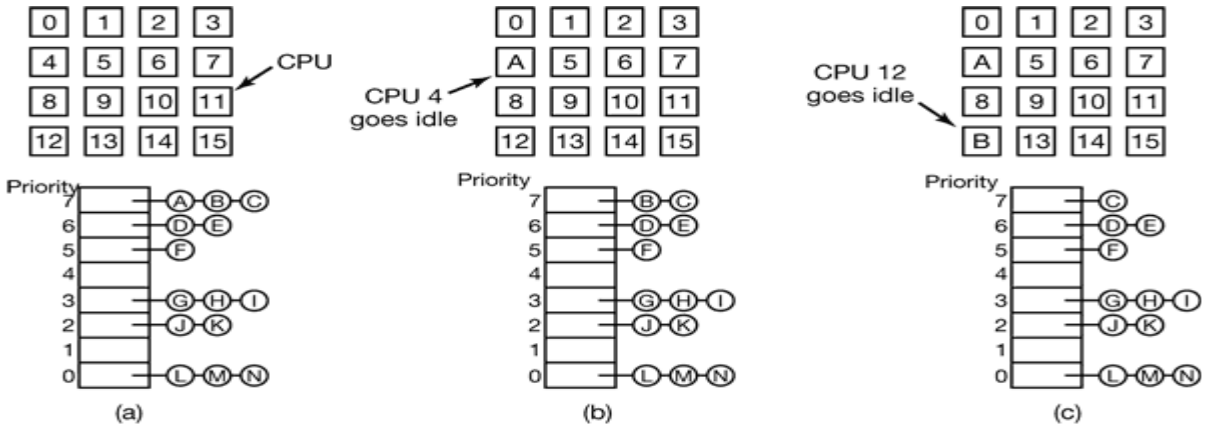


Figure 2.8 Time sharing scheduling

In figure 2.8, there the 16 CPUs are all currently busy. There are 14 set of prioritized processes are waiting to run. The first CPU to finish its current work is CPU 4. The highest priority process A is allocated to CPU4. Next, CPU 12 goes idle and chooses process B. This scheduling is reasonable as long as the processes are completely unrelated**.**

2.2.7.2 Space Sharing

This multiprocessor scheduling can be used when processes are related to one another. A single process creates multiple threads that work together. Scheduling multiple threads at the same time across multiple CPUs is called **space sharing.**

The simplest space sharing algorithm works like this.

Assume that an entire group of related threads is created at once.

i. All the threads start its execution when there is enough number of CPUs available. If there are not enough CPUs, none of the threads are started.
ii. Each thread holds its CPU until it terminates. If the threads wait for I/O, it continues to hold the CPU. The CPU is idle until the completion of I/O.
iii. The same algorithm is applied for the next batch of threads.

2.2.7.3 Gang Scheduling

Gang scheduling is used to schedule in both time and space together.

It has three parts

1. Groups of related threads are scheduled as a unit, a gang.
2. All members of a gang run simultaneously, on different timeshared CPUs.
3. All gang members start and end their time slices together.

2.2.7 PERFORMANCE EVALUATION OF THE SCHEDULING

Selecting a proper scheduling algorithm is difficult task. Criteria are often defined in terms of CPU utilization, throughput, waiting time or response time. For evaluating algorithms following methods are used.

1. Deterministic modeling
2. Queuing model
3. Simulation

2.2.7.1 DETERMINISTIC MODELING

One type of analytic evaluation method is **Deterministic modeling.** This method takes a particular predetermined workload and defines the performance of each algorithm for that workload. Deterministic modeling is simple and fast. It gives us exact numbers, allowing us to compare the algorithms. The main uses of deterministic modeling are in describing scheduling algorithms and providing examples. In general deterministic modeling is too specific and requires too much exact knowledge.

2.2.7.2 QUEUING MODEL

The computer system is described as a network of servers. Each server has a queue of waiting processes. The CPU is a server with its ready queue, as is the I/O system with its device queues. Knowing arrival rates and service rates, we can compute utilization, average queue length, average wait time, and so on. This area of study is called queueing-network analysis.

As an example, let η be the average queue length, let W be the average waiting time in the queue, and let λ be the average arrival rate for new processes in the queue. We expect that during the time W that a process waits, λ x W new processes will arrive in the queue.

If the system is in a steady state, then the number of processes leaving the queue must be equal to the number of processes that arrive. Thus,

$$\eta = \lambda \times W.$$

This equation, known as Little's formula.

Queueing analysis can be useful in comparing scheduling algorithms, but it has some limitations.

2.2.7.3 SIMULATION

Simulation is used for more accurate evaluation of scheduling algorithms. It involves programming a model of the computer system. Software data structures represent the major components of the system. The simulator has a variable representing a clock. As this variable's value is increased, the simulator modifies the system state to reflect the activities of the devices, the processes, and the scheduler. The data to drive the simulation can be generated in several ways. The most common method uses a random-number generator. Simulations can be expensive, often requiring hours of computer time

2.3 INTERPROCESS COMMUNICATION AND SYNCHRONIZATION

Concurrently executing processes in the operating system may be either independent processes or cooperating processes. A process is independent if it cannot affect or be affected by the other processes 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. Any process that shares data with other processes is a cooperating process.

Interprocess communication (IPC) is a mechanism that allows the exchange of data between processes. It allows the processes to synchronize their action without sharing the same address space.

There are two Models of Interprocess communication. They are

1. Shared Memory
2. Message passing

2.3.1 SHARED MEMORY:

An area of memory that is shared among the cooperating processes is known as shared memory. Normally the shared-memory region exists in address space of the process creating the shared memory segment. Other processes that wish to communicate using this shared memory segment must attach it to their address space. The processes can then exchange information by reading and writing data in the shared areas.

Example: producer-consumer problem

Consider two processes named as producer and consumer. A producer process produces information that is consumed by a consumer process. The inter-process communication occurs through a shared buffer that can be filled by the producer and emptied by the consumer. A producer can produce one item while the consumer is consuming another item. The producer and consumer must be synchronized, so that the consumer does not try to consume an item that has not yet been produced.

Two types of buffers used are

1. Unbounded buffer – Variable size buffer
2. Bounded buffer – Fixed size buffer

Let us assume that the bounded buffer is used as shared memory. The shared buffer is implemented as a circular array with two pointers: in and out.

```
#define BUFFER_SIZE 10
int in = 0;
int out = 0;
int count = 0;
int buffer[BUFFER_SIZE];


Producer process
Void Producer()
{
int item;
while (true)
{
item = produceitem();
if(count == BUFFER_SIZE); /* do nothing */
buffer[in] = item;
in = (in + 1) % BUFFER_SIZE;
count++;
}
```

```
Consumer process
Void consumer()
(
Int item;
while (true) {
if(count == 0); /* do nothing */
item = buffer[out];
out = (out + 1) % BUFFER_SIZE;
count--;
}
```

From the above example, it is clear the both the producer process and the consumer process attempt to access the shared buffer concurrently.

2.3.2 MESSAGE PASSING

Message passing is the facility that allows processes to communicate and to synchronize their actions without sharing the same address space.

Example: Chat program allows the participants to communicate each other by exchanging the messages.

Two operations of message passing system are

1. Send(message)
2. Receive(message)

There are two types of communication of message passing system. They are

- Direct communication
- Indirect communication

2.3.2.1 DIRECT COMMUNICATION

In this type of communication each process wanting to communicate must explicitly name the recipient or sender of the communication. The general syntax of send and receive messages are

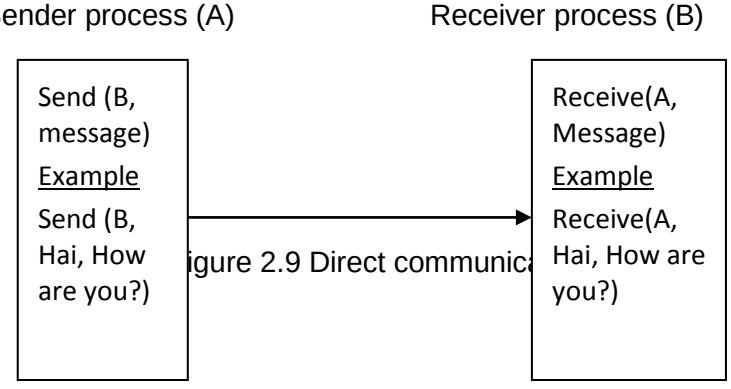
The syntax of the send and receive are

Send (P, Message)

Send a message to process P.

Receive (Q, Message)

Receive a message from process Q.



igure 2.9 Direct communica

Example – Producer –Consumer Problem

Process A

```
while (TRUE) {
 produce an item
        send (B, item)
              }
```

Process B

```
while (TRUE) {
receive(A,item)
consume-item
}
```

Properties of direct communication link are :

- Links are established automatically between each pair of processes that wants to communicate.
- A link is associated with exactly one pair of communicating processes.
- Between each pair there exists exactly one link.
- The link may be unidirectional or bi-directional.

2.3.2.2 INDIRECT COMMUNICATION

In this type messages sent to and received from *mailboxes* (also referred as ports).

The syntax of the send and receive are

Send (B, Message)

Send a message to mail box B.

Receive (B, Message)

Receive a message from mail box B.

Process can communicate only if they share a mailbox. Each mailbox has a unique id Mailboxes can be seen as objects into which process placed their messages and the placed messages can be removed by other processes.

Properties of Indirect communication link are

- Link is established only if share a common mailbox.
- A link may be associated with many processes.
- Each pair of processes may share several communication links
- The link may be unidirectional or bi-directional.

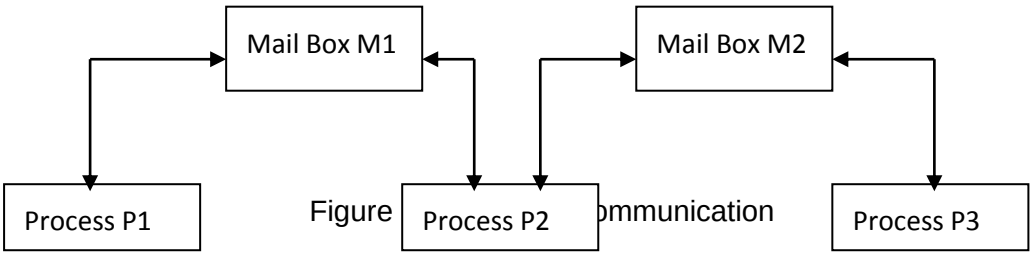


Figure ommunication

There are two types of mail boxes. They are

1. Process owned - The process that creates the mail box is the owner. Owner can only receive messages through this mailbox. Other processes can only send messages to this mail box. When the owner terminates, the mail box disappears.
2. System owned – The operating system is the owner of this mailbox. It is independent and is not attached to any particular process.

2.3.3 RACE CONDITION

The situation where more than one processes access and manipulate shared data at a time then the result of the execution depends on the order in which the access takes place, is called a race condition.

2.3.4 CRITICAL SECTION

Consider more than one of processes competing to use some shared data. Each process has a code segment, called critical section in which the process can access common variables and files. Critical section is used to avoid race condition. The main feature of the operating system is that, when one process in critical section, no other is allowed to be in its critical section.

Each process has three sections. They are

Entry section: This section contains code which is used by the processes to get permission to enter into the critical section.

Remainder section: contains the remaining codes.

Exit section: contains code that is used by the processes to come out of the critical section

While(true)

{

Entry section

Critical section

Exit section

Remainder section

}

The critical sections should satisfy the following requirements.

1. Mutual exclusion: Only one process can be in the critical section at a time. i.e., If one process is executing in its critical section, then other processes are not allowed to execute their critical section.

2. Progress: If no process is executing in its critical section then only those processes that are not executing in their remainder sections will enter its critical section next.

3. Bounded waiting: After a process made a request to enter its critical section and beforeit is granted the permission to enter, there exists a bound on the number of times that other processes are allowed to enter.

2.3.5 MUTUAL EXCLUSION

If process *Pi* is executing in its critical section, then no other processes can be executing in their critical sections is called critical section.

We consider only two processes that interchange execution between their critical sections and remainder sections. Both processes share the common integer variable.

Let us consider the name of the processes is p0 and p1, the name of the variable is turn and flag. If flag [0] is true then process P0is ready to enter its critical section. The variable turn indicates whose turn it is to enter its critical section. That is, if turn == 0, then process *P0* is allowed to execute in its critical section.

```
do
{
flag[0]=true;
turn = 1;
while(flag(1) && turn==1);

critical section

flag[0]=false;
} while(true)
```

To enter the critical section, process *P0* first sets flag [0] to be true and then sets turn to the value. Then it checks the condition. Since the condition is false, process p0 performs no-operation at entry section. It enters its critical section. Once it completes its execution, it sets the flag value as false. The same procedure used by process p1 to executes its critical section. From the example the mutual exclusion is preserved.

2.3.6 SEMAPHORES

Semaphore is a hardware based solution used to solve critical section problem. A Semaphore is indicated by an integer variable S. Semaphore variable is accessed through two operations wait() and signal().

1. Wait: It is used to test the semaphore variable. It decrement the semaphore value. If the value become negative, then the execution of wait() operation is blocked.
2. Signal: It increment the semaphore value.

Entry to the critical section is controlled by wait operation and exit from a critical section is taken care by signal operation. The wait and signal operation are also called P and V operations. If number of processes attempts the wait operation, only one process will be allowed to proceed. Thus mutual exclusion is enforced

Pseudo code for wait

```
Wait(S)
{
While(S<=0) do no-op;
S=S-1;
}
```

Pseudo code for signal

```
Signal(S)
{
S=S+1
}
```

Semaphores are of two types

1. Binary Semaphore – It can take the value 0 and 1 only.
2. Counting semaphore – It can take any positive integer value.

2.4 DEADLOCK

2.4.1 DEFINITION

In a multiprogramming environment, several processes may compete for a finite number of resources. A process requests resources; 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 processes. This situation is called a deadlock.

Deadlocks are a set of blocked processes each holding a resource and waiting to acquire a resource held by another process.

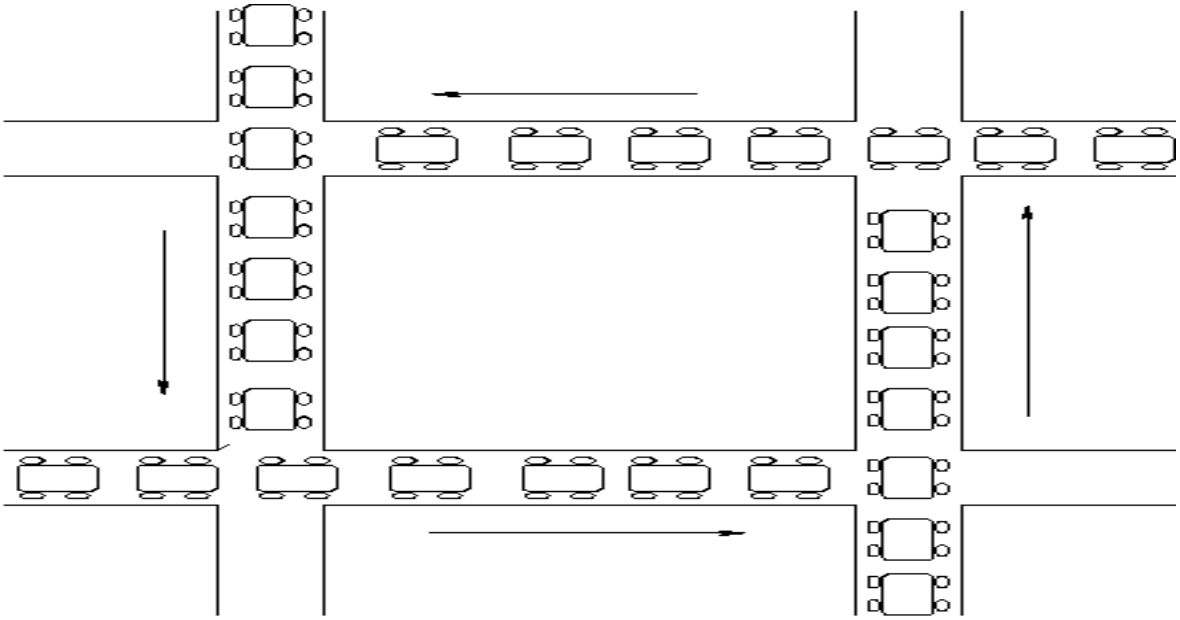


Figure 2.11 Traffic gridlock is a real time example for deadlock situation

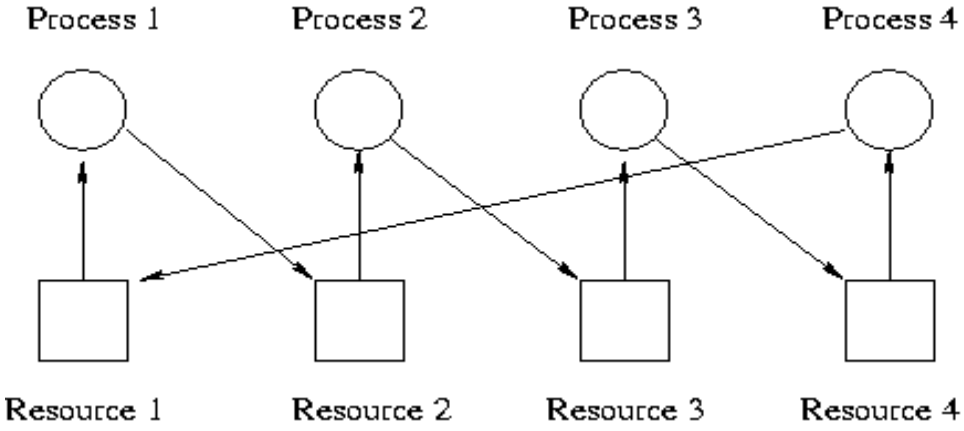


Figure 2.12 Deadlock in operating system

A process must request a resource before using it and must release the resource after using it. A process may request as many resources as it requires for completing its task. Obviously, the number of resources requested may not exceed the total number of resources available in the system.

Under the normal mode of operation, a process may utilize a resource in only the following sequence:

- Request. The process requests the resource. If the request cannot be granted immediately, then the requesting process must wait until it can acquire the resource.
- Use. The process can use resource.
- Release. Release the resource.

2.4.2 DEADLOCK CHARACTERISTICS

Deadlock can arise if four conditions hold simultaneously

1. Mutual exclusion.

At least one resource must be held in a non-sharable mode; If any other process requests this resource, then that process must wait for the resource to be released.

2. Hold and wait.

A process holding at least one resource is waiting to acquire additional resources held by other processes.

3. No preemption.

Resources cannot be preempted; that is, a resource can be released only voluntarily by the process holding it, after that process has completed its task.

4. Circular wait.

A set {P0, P1 … Pn} of waiting processes must exist such that P0is waiting for a resource held by P1, P1 is waiting for a resource held by P2, ... , Pn-1 is waiting for a resource held by Pn, and Pn is waiting for a resource held by P0.

2.4.3 DEAD LOCK PREVENTION

Deadlocks can be prevented by preventing at least one of the four required conditions.

1. Mutual Exclusion

The mutual-exclusion condition must hold for non-sharable resources. For example printers and tape drives cannot be simultaneously shared by several processes. Shared resources such as read-only files do not require mutually exclusive access and thus cannot lead to deadlocks.

2. Hold and Wait

To ensure that the hold-and-wait condition never occurs in the system, we must guarantee that, whenever a process requests a resource, it does not hold any other resources.

One way used is each process must collect all the resources before it begins execution. Another way allows a process to request resources only when it has none. A process may request some resources and use them. Before it can request any additional resources, however, it must release all the resources that it is currently allocated.

Both these methods have two main disadvantages.

1. Resource utilization may be low, since resources may be allocated but unused for a long period.
2. Starvation is possible. A process that needs several popular resources may have to wait indefinitely.

3. No preemption

If a process is holding some resources and requests another resource that cannot be immediately allocated to it then all resources the process is currently holding are preempted. The preempted resources are added to the list of resources for which the process is waiting. The process will be restarted only when it can regain its old resources in addition with the new ones that it is requesting.

4. Circular wait

One way to prevent the circular wait condition is by linear ordering of different types of resources. In this, the resources are divided into different classes. The following rules are used to prevent the possibility of circular wait.

- Assign a unique integer number to each resource type.
- Each process requests resources in an increasing order of enumeration.

For Example, consider the following resources

Tape drive = 1
Disk drive = 5
Printer = 12

A process that wants to use the tape drive and printer at the same time must first request the tape drive and then request the printer.

2.4.4 DEADLOCK AVOIDANCE

Deadlock-prevention algorithms prevent deadlocks by limiting how requests can be made. An alternative method for avoiding deadlocks is to require additional information about how resources are to be requested. A deadlock-avoidance algorithm dynamically examines the resource-allocation state to ensure that a circular wait condition can never exist. The resource-allocation *state* is defined by the number of available and allocated resources and the maximum demands of the processes.

2.4.4.1 SAFE STATE

A safe state is a state in which the system can allocate resources to each process in some order and still avoid a deadlock. A sequence of processes <P1, P2, ... , *Pn*>is a safe sequence for the current allocation state if, for each *Pi,* the resource requests that *Pi* can still make can be satisfied by the currently available resources plus the resources held by all *Pj,* with *j <i.*

A safe state is not a deadlocked state. An unsafe state *may* lead to a deadlock. As long as the state is safe, the operating system can avoid deadlocked. In an unsafe state, the operating system cannot prevent processes from requesting resources in such a way that a deadlock occurs.

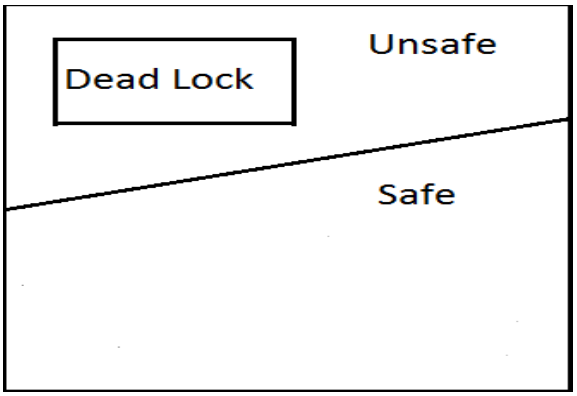


Figure 2.13 Safe, unsafe, and deadlocked state spaces

Let us we consider a system with 12 magnetic tape drives and 3 processes namely P0*,* P1and P2. The maximum needs, number of resources allocated and the current needs of each process are given in the table.

Process	Max Needs	Allocated	Current Needs	Available
P0	10	5	5	3
P1	4	2	2	
P2	9	2	7	

At time *t0,* the system is in a safe state. The sequence <P1, P0, P2> satisfies the safety condition. Process P1 can immediately be allocated all its tape drives and then return them, then process *Po* can get all its tape drives and return them and finally process *P2* can get all its tape drives and return them. The safe sequence is shown in the following table.

Process	Max Needs	Initially Allocated	Currently allocated	Available	No. of resources returned by processes after execution	Total available
P1	4	2	2	1	4	5
P0	10	5	5	0	10	10
P2	9	2	7	3	9	12

2.4.4.2 RESOURCE - ALLOCATION GRAPH

If we have only one instance of each resource type, then resource allocation graph is used for deadlock avoidance. The resource allocation graph normally contains the request and assignment edges. In addition to that, a new type of edge, called a claim edge is added. A claim edge Pi →Rj indicates that process Pi may request resource Rj at some time in the

future. The claim edge is shown by dotted lines. Figure 2.14 shows the deadlock avoidance with resource allocation graph.

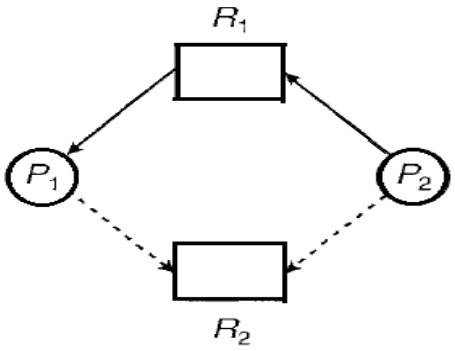


Figure 2.14 Resource allocation graph for deadlock avoidance

When process Pi requests resource Rj, the claim edge Pi →Rj is converted to a request edge. Similarly, when a resource Rj is released by Pi the assignment edge Rj→ Pi is reconverted to a claim edge Pi →Rj.

Cycle detection algorithm is used for detecting cycle in the graph. If no cycle exists, then the allocation of the resource will leave the system in a safe state. If a cycle is found, then the allocation will put the system in an unsafe state.

2.4.4.3 BANKER'S ALGORITHM

Resource allocation graph is not applicable for deadlock avoidance if we have several instance of each resource type. Banker's algorithm is the best known deadlock avoidance strategies for several instances of resources.

When a new process enters the system, it must declare the maximum number of instances of each resource type that it may need. This number may not exceed the total number of resources in the system. When a user requests a set of resources, the system must determine whether the allocation of these resources will leave the system in a safe state. If it will, the resources are allocated; otherwise, the process must wait until some other process releases enough resources.

Several data structures must be maintained to implement the banker's algorithm. We need the following data structures, where n is the number of processes in the system and mis the number of resource types.

- **Available**: A vector of length m indicates the number of available resources of each type.
- **Max:** An n x m matrix defines the maximum demand of each process.
- **Allocation:** An n x m matrix defines the number of resources of each type currently allocated to each process.
- **Need:** An n x m matrix indicates the remaining resource need of each process.

2.4.4.3.1 Safety Algorithm

Safety algorithm is used to find whether or not a system is in a safe state. This algorithm can be described as follows

1. Let *Work* and *Finish* be vectors of length *m* and *n,* respectively. Initialize*Work*= *Available* and *Finish[i] =false* for *i* = 0, 1, ... ,*n* - 1.
2. Find an I such that both

   a. *Finish[i] ==false*

   b. $Need_i$<= *Work*

   If no such *i* exists, go to step 4.
3. *Work = Work + Allocation*$_i$

   Finish[i] = true

   Go to step 2.
4. 4If *Finish[i] ==true* for all *i,* then the system is in a safe state

2.4.4.3.2 Resource – Request Algorithm

Next, we describe the algorithm for determining whether requests can be safely granted. 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 *Ri.* When a request for resources is made by process P_i, the following actions are taken:

1. If $Request_i$<= $Need_i$, go to step 2. Otherwise, raise an error condition, since the process has exceeded its maximum claim.
2. If $Request_i$<=*Available,* go to step 3. Otherwise, P_i must wait, since the resources are not available.
3. Have the system pretend to have allocated the requested resources to process $P_i$by modifying the state as follows:

   Available= Available - Request$_i$

   Allocation$_i$ =*Allocation*$_i$ +*Request*$_i$

   $Need_i$ =$Need_i$- $Request_i$

If the resulting resource-allocation state is safe, the transaction is completed, and process P_i is allocated its resources. However, if the new state is unsafe, then P_i must wait for $Request_i$, and the old resource-allocation state is restored.

2.4.5 DEADLOCK DETECTION AND RECOVERY

2.4.5.1 DEADLOCK DETECTION

If a system does not employ either a deadlock-prevention or a deadlock avoidance algorithm, then a deadlock situation may occur. The operating system periodically performs the deadlock detection algorithm to detect the deadlock.

2.4.5.1.1 Single Instance of Each Resource Type

If all resources have only a single instance, then a *wait-for* graph is used to detect the deadlock.

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. An edge $P_i \rightarrow P_j$ exists in a wait-for graph if and only if the corresponding resource allocation graph contains two edges $P_i \rightarrow R_q$ and $R_q \rightarrow P_j$ for some resource R_q.

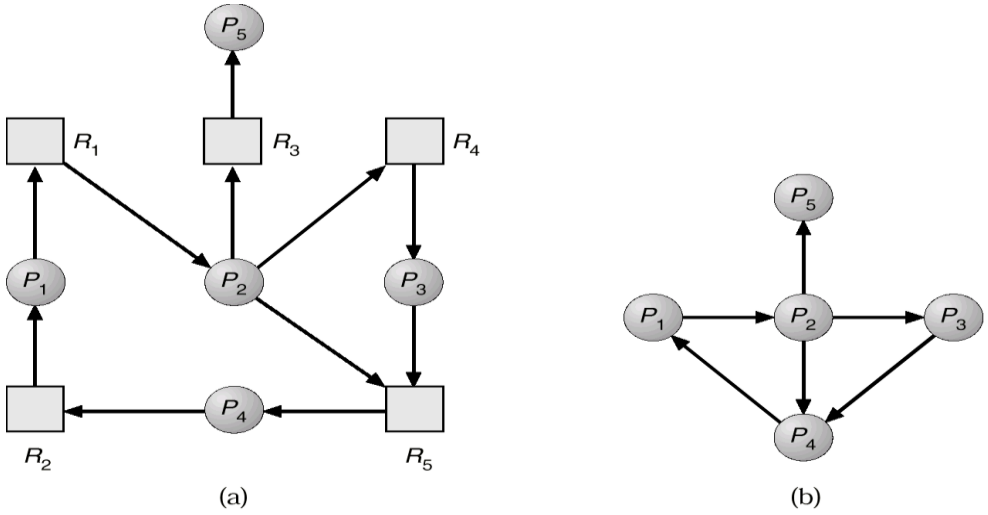


Figure 2.15 Resource allocation graph with corresponding wait for graph

If the wait-for graph contains a cycle, then there is a deadlock. To detect deadlocks, the system needs to *maintain* the wait-for graph and periodically *invoke an algorithm* that searches for a cycle in the graph.

2.4.5.1.2 Several Instances of Resource Type

The wait-for graph scheme is not applicable to a resource-allocation system with multiple instances of each resource type. The data structures used are

- **Available:** A vector of length m indicates the number of available resources of each type.
- **Allocation:** An n x m matrix defines the number of resources of each type currently allocated to each process.
- **Request:** An *n* x *m* matrix indicates the current request of each process.

1. Let *Work* and *Finish* be vectors of length *m* and *n,* respectively. Initialize *Work= Available. For i*= 0, 1, ... ,*n-1,* if $Allocation_i$ <>0, then *Finish[i] =false;* otherwise, *Finish[i] = true.*

2. Find an index *i* such that both
   a. *Finish[i] ==false*
   b. $Request_i$<=*Work*
   If no such *i* exists, go to step 4.

3. *Work= Work+* $Allocation_i$
   Finish[i] = true
   Go to step 2.

4. If *Finish[i] ==false* for some *i,* 0<= *i* <*n,* then the system is in a deadlocked state. Moreover, if *Finish[i] ==false,* then process P_i is deadlocked.

2.5.4.2 DEADLOCK RECOVERY

Once deadlock has been detected, some strategy is needed for recovery.

Process Termination:

To eliminate deadlocks by aborting a process, we use one of two methods

- Abort all deadlocked processes. This method clearly will break the deadlock cycle.
- Abort one process at a time until the deadlock cycle is eliminated

.

Resource Preemption:

To eliminate deadlocks using resource preemption, we successively preempt some resources from processes and give these resources to other processes until the deadlock cycle is broken.

- Select a victim - which process and which resource to preempt.
- Rollback to previously defined "safe" state.
- Prevent one process from always being the one preempted(starvation).

In general, it's easier to preempt the resource, than to terminate the process.

Summary

- Process is an instance of program running on computer.
- In Operating system, system processes and user processes are the major categories of process.
- The process has different states-New, Ready, Running, Waiting and Terminated states.
- The thread is a dispatchable unit of work within the process
- The major difference between process and thread are

S.No	Process	Thread
1.	Process is heavy weight	Thread is light weight
2.	Process switching needs interaction with operating system.	Thread switching does not need to interact with operating system
3.	In multiple processing environments, each process executes the same code but has its own memory and file resources.	All threads can share same set of open files, child processes.
4.	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.

- The different types of threads are user level and kernel level threads
- Process scheduling is the method of determining which process in the ready state should be moved to running state.
- There are three types of schedulers-Long term,Short term and medium term
- Different scheduling algorithms are present which are following the criteria--- CPU utilisation, throughput, turnaround time, waiting time and response time

- The scheduling algorithms are first come first served, shortest job first scheduling, round robin scheduling, multiprocessor scheduling and gang scheduling.
- The algorithms are evaluated by the following method--Deterministic model, Queueing model and simulation
- Inter process communication is a mechanism that allows the exchange of data between processes the two models are shared memory and message passing
- Race condition is a situation where more than one processes access and manipulate shared data at a time then the result of the execution depends on the order in which the access takes place.
- Mutual exclusive is only one process can be in the critical section at a time, i.e if one process is executing in its critical section then other processes are not allowed to execute their critical section.
- Semaphore is a hardware based solution used to solve critical section problem.
- Deadlock is a situation if a waiting process is never again able to change state because the resources it has requested are held by other waiting processes.
- Deadlocks can be prevented by preventing at least one of the four required conditions-Mutual Exclusion, hold and wait, No pre-emption and circular wait.
- Deadlock avoidance can be done by safe state, Resource allocation graph, bankers Algorithm.
- Deadlock Detection algorithm is done by operating system periodically – there are two types of algorithm –single instance of each resource type ,several instances of resource type
- Deadlock recovery is done by process termination and resource pre-emption.

REVIEW QUESTIONS

PART - A

1. Define Process.
2. What is process control block?
3. List the states of the process.
4. Draw the structure of PCB.
5. Define context switching.
6. What is thread?
7. List the types of threads.
8. Write any two advantages of threads.
9. Define Throughput.
10. Define Turnaround time.
11. What is the objective of scheduling?
12. List the types of scheduler.
13. Define CPU scheduler.
14. Define swapping.
15. What do you mean by preemptive and non-preemptive scheduling?
16. What is multiprocessor scheduling?
17. When gang scheduling is used?
18. Define race condition.
19. Define critical section.
20. What is semaphore?
21. Define deadlock.
22. List the characteristics of deadlock.

23. What is mutual exclusion
24. What is a resource allocation graph?
25. What is a wait for graph?

PART – B

1. What are the benefits of threads?
2. Write short notes on user level threads.
3. Write brief notes on the concept of multithreading.
4. Differentiate process and threads.
5. Write any three scheduling criteria.
6. What is space scheduling?
7. Explain the following transition between process states
    a. Running → Ready
    b. Running → Terminated
    c. Running → Waiting
8. Briefly write about semaphore.
9. Write about deadlock recovery.

PART – C

1. Explain process state transition with neat diagram.
2. With neat sketch, write briefly about PCB.
3. Explain the types of threads.
4. Describe the various types of schedulers.
5. Explain FCFS scheduling algorithm with an example.
6. Explain non-preemptive SJF scheduling algorithm with an example.
7. Explain preemptive SJF scheduling algorithm with an example.
8. Explain round robin scheduling algorithm with an example.
9. Explain various types of multiprocessor scheduling.
10. Explain the various method for evaluating the performance of the scheduling algorithm
11. What is shared memory? Explain it with an example
12. Explain various message passing techniques.
13. Explain critical section in detail.
14. Describe the characteristics of deadlock.
15. What are the methods to prevent a deadlock? Explain
16. Explain how safe state is used to avoid deadlock with an example
17. Explain any one deadlock avoidance strategy.
18. Write the deadlock detection algorithm for several instances of resource system.

UNIT – III

MEMORY MANAGEMENT

OBJECTIVES

After studying this unit, the student should be able to:

- Provide a detailed description of various ways of organizing memory hardware.
- Understand the reason for memory partitioning and explain the various types that are used.
- Understand and explain the concept of fragmentation and its types.
- Understand and explain the concept of paging its advantages .
- Summarize key security issues related to memory management.
- Define virtual memory.
- Describe the hardware and control structures that support virtual memory.
- Describe the benefits of a virtual memory system.
- Explain the concepts of demand paging, page-replacement algorithms.

Introduction:

Memory is a valuable system resource which must be carefully **managed and shared** between programs and processes . Also it should be **protected** against access by other programs.

Memory management is the process of managing the computer memory which consist of primary memory or secondary memory. In this, we allocate the memory portions to programs and softwares after freeing the space of the computer memory. Basically, **memory management** is of critical importance for operating system because the multi-tasking can take place in the system which switches the memory space from one process to another. Moreover, the concept of the **virtual memory** is used in the system according to which programs from main memory are loaded to the secondary memory when the space is not large enough to hold the programs.

. 3.1 Basic Memory Management

3.1.1 Definition

The main memory is central to the operation of a modern computer system. It consists of large array of words or bytes. ***Memory management*** is a process of operating system which manages primary memory and moves processes between main memory and disk during execution. The CPU fetches instructions from memory according to the value of the program counter. Both instructions and the data must be brought to the main memory during execution. If the data are not in memory, they must be moved there before the CPU can operate on them .

Memory management keeps track of each and every memory location, regardless of either it is allocated to some process or it is free. It checks how much memory is to be allocated to processes. It tracks whenever some memory gets freed or unallocated and correspondingly it updates the status.

The main ***functions of memory management*** are:

- Keeps track of the status of each memory location, either *allocated* or *free*.
- Determines how memory is allocated among processes.
- Decides which process will get memory at what time .
- Determines which memory locations will be assigned.
- Keeps track of when memory is freed or *unallocated* and updates the status.

3.1.1.2 Types of memory :

Basically computer memory is classified as

- **Primary Memory (Main Memory)**
- **Secondary Memory (Auxillary Memory)**

Primary Memory (Main Memory)

Primary memory holds only those data and instructions on which computer is currently working. It has limited capacity and data is lost when power is switched off. Hence it is volatile. It is generally made up of semiconductor device. The data and instruction required to be processed reside in main memory. It is divided into two subcategories : RAM and ROM.

Secondary Memory (Auxiliary Memory)

This type of memory is also known as external memory or non-volatile. It is slower than main memory. It is ***used for storing data/Information permanently.*** CPU directly does not access these memories instead they are accessed via input-output routines. Contents of secondary memories are first transferred to main memory, and then CPU can access it. For example : disk, CD-ROM, DVD etc.

3.1.2 Logical & Physical Address Map

Logical address otherwise called as Virtual address is an address used by software which is generated by the CPU. It is also referred to as virtual address. During execution of the program, the CPU requests the needed data from main memory using logical address.

Physical address actual memory address which denotes a memory area in the storage device is the one that is loaded into the memory address register of the memory. Physical memory may be mapped to different logical addresses for various purposes.

The process of converting logical (virtual) address into physical address at run time is called **memory mapping**. This run-time mapping is done by a hardware device called the memory-management unit (MMU).

The MMU has two special registers that are accessed by the CPU's control unit. A data to be sent to main memory or retrieved from memory is stored in the *Memory Data Register* (MDR). The desired logical memory address is stored in the *Memory Address Register* (MAR). The address translation is also called ***address binding*** and uses a memory

map that is programmed by the operating system. Before memory addresses are loaded on to the system bus, they are translated to physical addresses by the MMU.

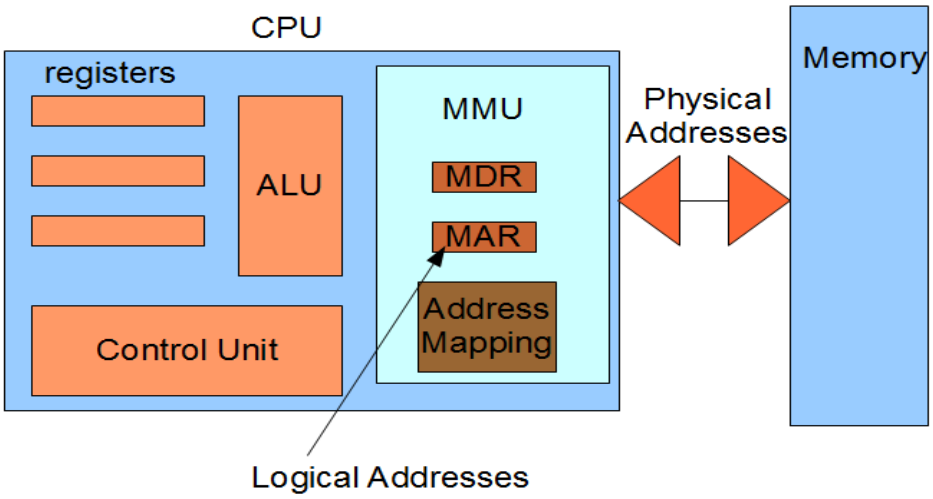


Fig 3.1 Memory Management Unit (MMU)

In compile-time and load-time address-binding schemes of logical and physical addresses are same , whereas in execution-time the address-binding schemes will differ.

The set of all logical addresses (generated by a program) is referred to as Logical address space. The set of all Physical addresses corresponding to these logical addresses is referred to as physical address space.

For example, P2 is a user program, with size 256 KB. But program is loaded in the main memory from 13000 to 13256 KB; this address is called physical address. The user program deals with logical addresses.

The value in the relocation register is added to every address generated by a user process and it is then sent to memory. Mapping is done as follows: Physical address space = Logical address space + relocation register value ie., 13256 = 256 + 13000

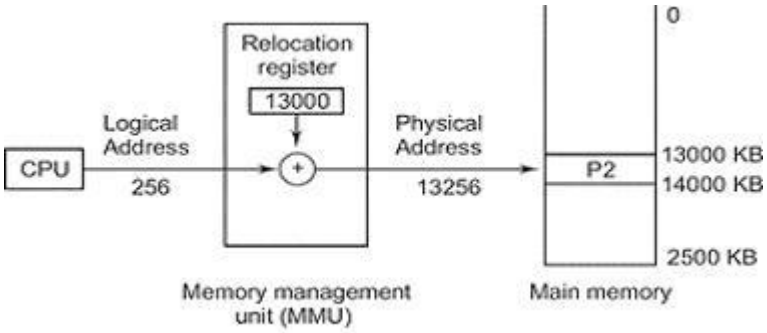


Fig 3.2 Memory Mapping

3.1.3 Memory Allocation

Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space. It is the process of reserving a partial or complete portion of computer memory for the execution of programs and processes. Programs and services are assigned with a specific memory as per their requirements when they are executed. Once the program has finished its operation or is idle, the memory is released and allocated to another program or merged within the primary memory.

Memory allocation methods:

- **Contiguous allocation**
- **Fixed partition allocation**
- **Variable partition allocation**

3.1.3.1 Contiguous allocation :

Contiguous memory allocation is a method that assigns a user process in memory blocks having consecutive addresses.

The main memory is usually divided into two partitions: one for the resident operating system and one for the user processes.

- **Low Memory** − Operating system resides in this memory.
- **High Memory** − User processes are held in high memory.

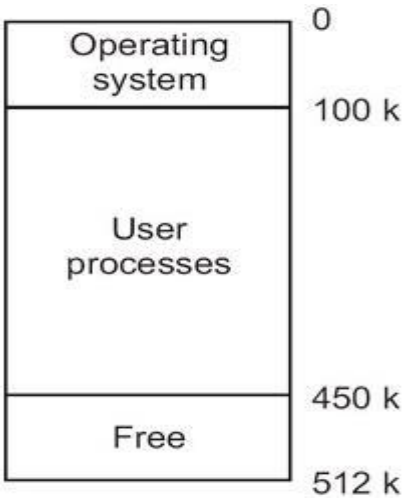


Fig 3.3 Contiguous memory allocation

In contiguous memory allocation, each process is contained in a single contiguous section of memory. In this type of allocation, relocation-register scheme is used to protect user processes from each other, and from changing operating-system code and data. Relocation register contains value of smallest physical address whereas limit register contains range of logical addresses. With relocation and limit registers, each logical address must be less than the limit register; the MMU maps the logical address dynamically by adding the value in the relocation register. This mapped address is sent to memory. This is depicted in the following diagram.

Hardware Support for Relocation and Limit Registers

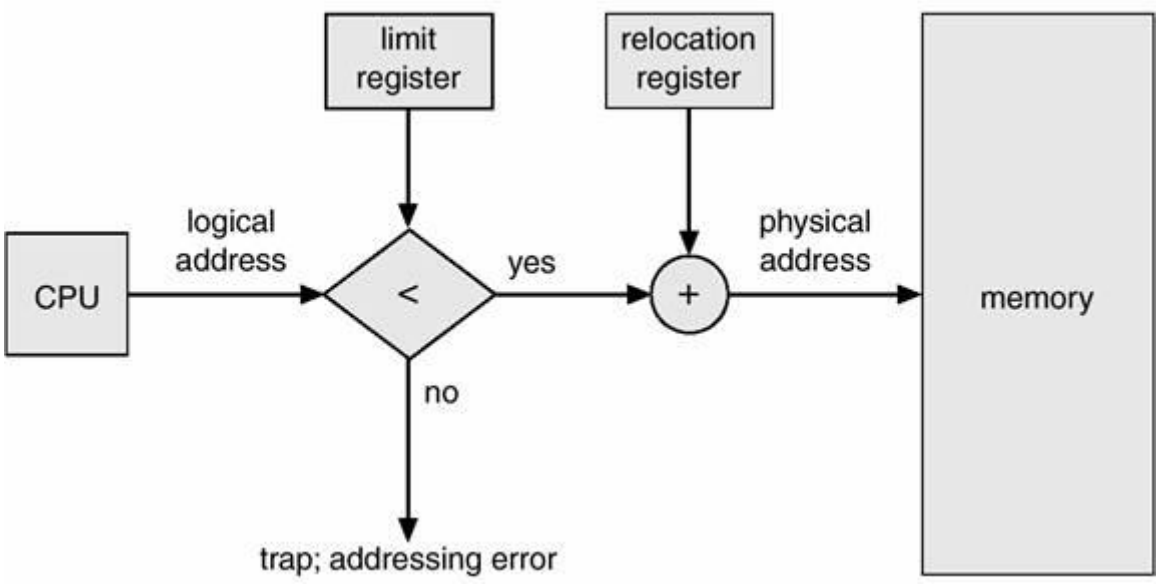


Fig 3.4 Protection using Relocation & limit registers

Advantages

- It is simple.
- It is easy to understand and use.

Disadvantages

- It leads to poor utilization of processor and memory.
  - User process is limited to the size of available memory

3.1.3.2 Fixed and variable partition

3.1.3.2.1FixedpartitionAllocation

One of the simplest methods for allocating memory is to divide memory into several fixed sized, non overlapping partitions. This method divides the main memory into equal number of fixed sized partitions, operating system occupies some fixed portion and remaining portion of main memory is available for user processes. Any process whose size is less than or equal to a partition size can be loaded into the partition. If all partitions are occupied, the operating system can swap a process out of a partition. If a program is too large to fit in a partition, then the programmer must redesign the program .

In this method, the main memory use is inefficient. Any program, no matter how small, occupies an entire partition. The left over space in partition, after program assignment, is called internal fragmentation.Unequal-size partitions will decrease these problems. Equal-size partitions was used in early IBMs OS/MFT (Multiprogramming with a Fixed number of Tasks)

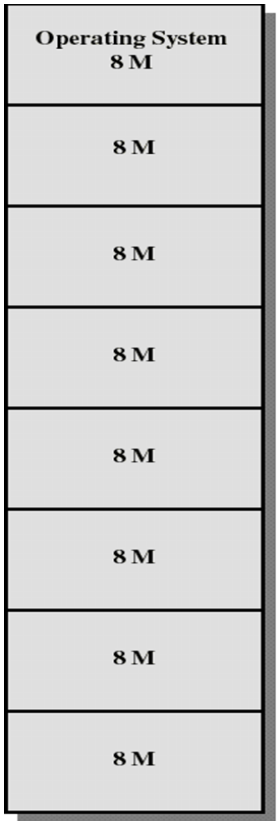


Fig 3.5 Fixed Partition - Example

Advantages

1. Any process whose size is less than or equal to the partition size can be loaded into any available partition.

2 . It supports multiprogramming.

Disadvantages

1.If a program is too big to fit into a partition use overlay technique.

2, Memory use is inefficient, i.e., block of data loaded into memory may be smaller than the partition. It is known as internal fragmentation.

3.1.3.2.2 Variable Size Partitions

In this type of allocation, main memory is divided into a number of fixed-sized partitions where each partition should contain only one process. 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.

Each partition may contain exactly one process. 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. The operating system keeps a table indicating which parts of memory are available and which are occupied. Finally, when a process arrives and needs memory, a memory section large enough for this process is provided.

By using variable size partitions, we can overcome the disadvantages present in fixed size partitioning. This is shown in the figure below:

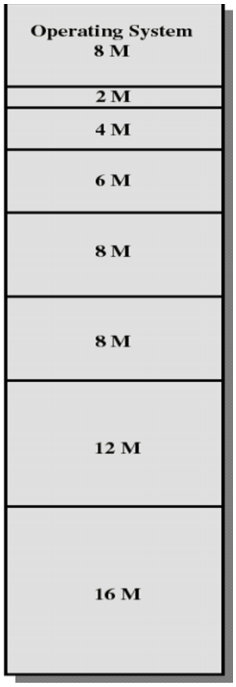


Fig 3.6 Variable Partition - Example

3.1.4 Internal ,External Fragmentation & Compaction

Fragmentation occurs in memory allocation system when many of the free blocks are too small to satisfy any request.

As processes are loaded and removed from memory, the free memory space is broken into little pieces. It happens after sometimes that processes cannot 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**
2. ***External fragmentation***

3.1.4.1 Internal & External fragmentation:

Internal Fragmentation:

Internal fragmentation is the space wasted inside of allocated memory blocks because of restriction on the allowed sizes of allocated blocks. Allocated memory may be slightly larger than requested memory; this size difference is memory internal to a partition, but not being used. Internal fragmentation occurs when memory allocation is based on fixed-size partitions where after a small size application is assigned to a slot and the remaining free space of that slot is wasted. Internal fragmentation occurs when more storage is allocated than is actually requested. This left over space, known as slack space, causes a degradation of system performance

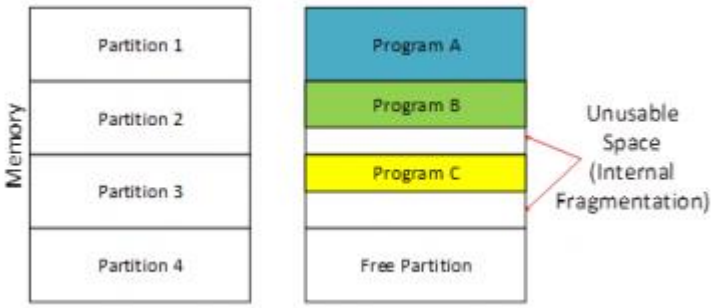
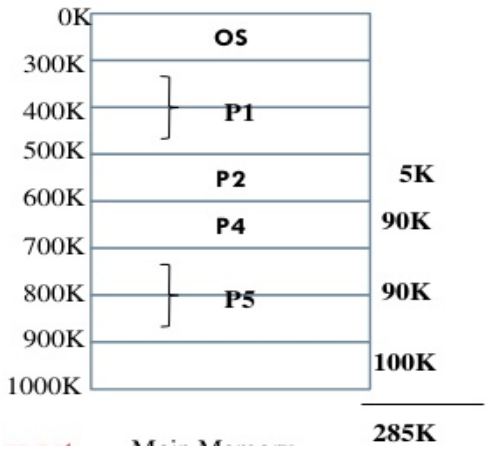


Fig 3.7 Internal Fragmentation

Example: Internal Fragmentation

Jobs	Memory Requirement
P1	200K
P2	95K
P3	540K
P4	10K
P5	110K
P6	250



Main Memory

We have 285 k memory available, but we can not fix p6 process, due to Internal fragmentation.

External Fragmentation:

External Fragmentation happens when a dynamic memory allocation algorithm allocates some memory and a small piece is left over that cannot be effectively used. If too much external fragmentation occurs, the amount of usable memory is drastically reduced. Total memory space is enough to satisfy a request or to reside a process in it, but it is not contiguous so it can not be used. For example, this can cause problems when an application requests a block of memory of 1000bytes, but the largest contiguous block of memory is only 300bytes. Even if ten blocks of 300 bytes are found, the allocation request will fail because they are not contiguous.

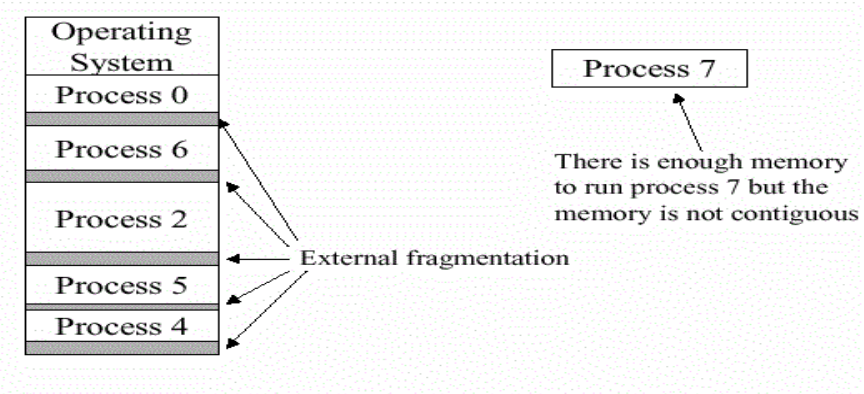
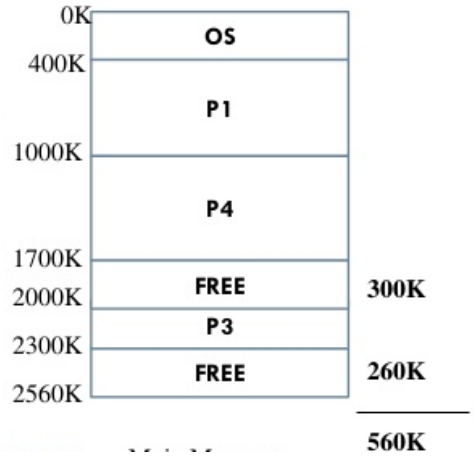


Fig: 3.8 External fragmentation

Example : External Fragmentation

Jobs	Memory Requirement
P1	600K
P2	1000K
P3	300K
P4	700K
P5	500K
P6	200

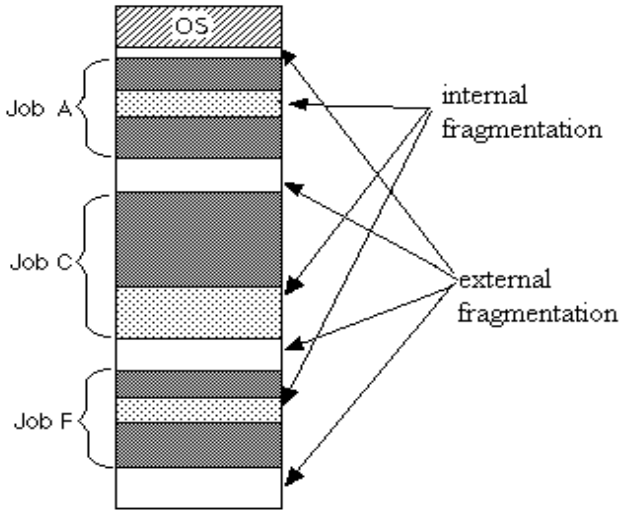


Main Memory

We have 560k of available memory but we cannot fix P5 process, due to External fragmentation.

Difference between Internal and External Fragmentation:

Internal Fragmentation	***External Fragmentation***
It is found in Fixed partition scheme.	*It is found in variable partition scheme.*
This can be reduced using External Fragmentation. However this solution suffer from external fragmentation.	*This can be solved using compaction where all the empty spaces are combined together.*



.

Fig 3.9 showing the difference between Internal and External Fragmentation

3.1.4.2 Compaction:

Memory compaction is the process of moving allocated objects together, and leaving empty space together. Swapping creates multiple fragments in the memory because of the processes moving in and out. Memory compaction refers to combining all the empty spaces together and then combining all the processes together. The disadvantage of memory compaction is that it requires too much of CPU time.

The following diagram shows how fragmentation can cause waste of memory and a compaction technique can be used to create more free memory out of fragmented memory

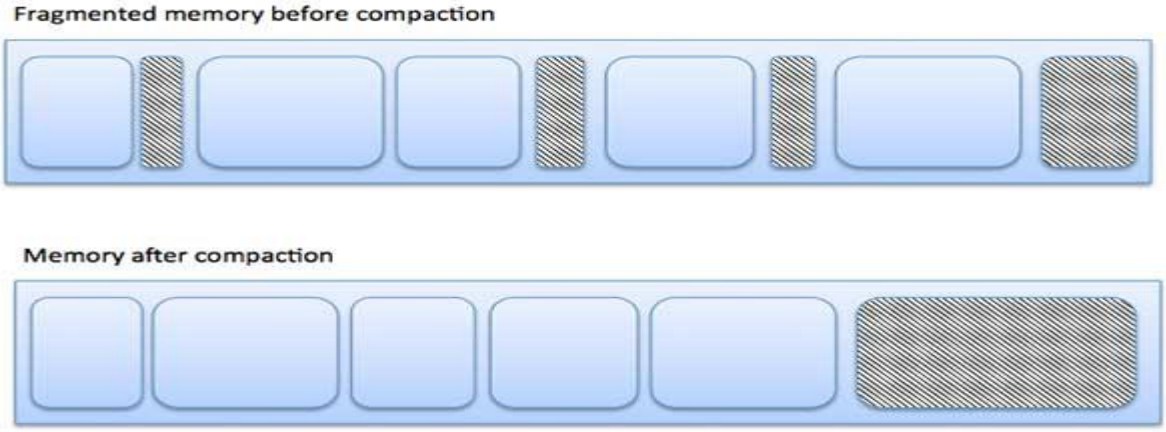


Fig 3.10 Compaction

External fragmentation can be reduced by compaction or shuffle memory contents to place all free memory together in one large block. To make compaction feasible, relocation should be dynamic.

3.1.5. **Paging**

3.1.5.1 Basic principle

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called virtual memory.

Paging technique plays an important role in implementing virtual memory.

Paging is a memory management technique in which process (logical) address space is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes). The size of the process is measured in the number of pages.

Similarly, main memory is divided into small fixed-sized blocks of (physical) memory called **frames** (size is power of 2, between 512 bytes and 8192 bytes). and the size of a frame is kept the same as that of a page to have optimum utilization of the main memory and to avoid external fragmentation

3.1.6.2 Address Translation

Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

Frame address is called **physical address** and represented by a **frame number** and the **offset**.

Physical Address = Frame number + page offset

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

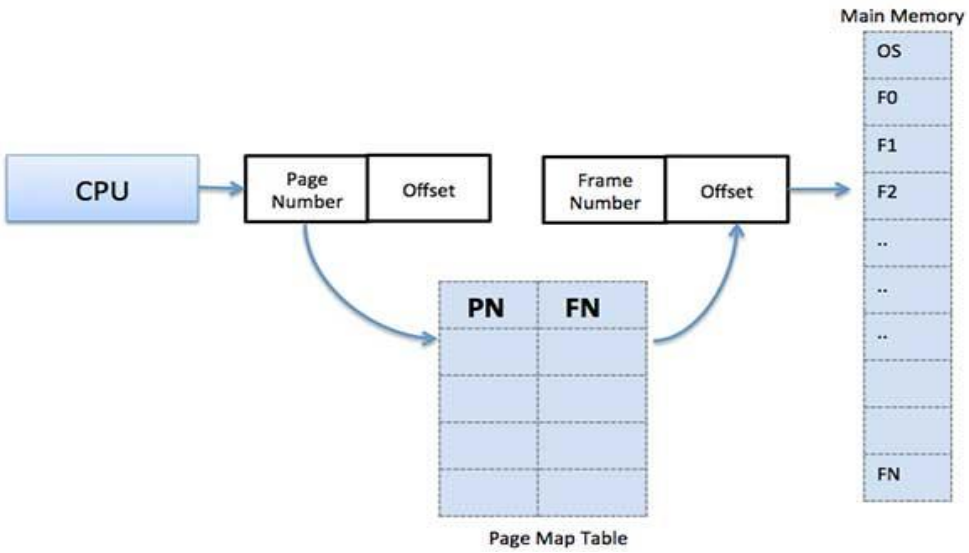


Fig 3.11 Page Map Table

When the system allocates a frame to any page, it translates this logical address into a physical address and create entry into the page table to be used throughout execution of the program.

When a process is to be executed, its corresponding pages are loaded into any available memory frames. Suppose you have a program of 8Kb but your memory can accommodate only 5Kb at a given point in time, then the paging concept will come into picture. When a computer runs out of RAM, the operating system (OS) will move idle or unwanted pages of memory to secondary memory to free up RAM for other processes and brings them back when needed by the program.

This process continues during the whole execution of the program where the OS keeps removing idle pages from the main memory and write them onto the secondary memory and bring them back when required by the program.

Address Translation Architecture

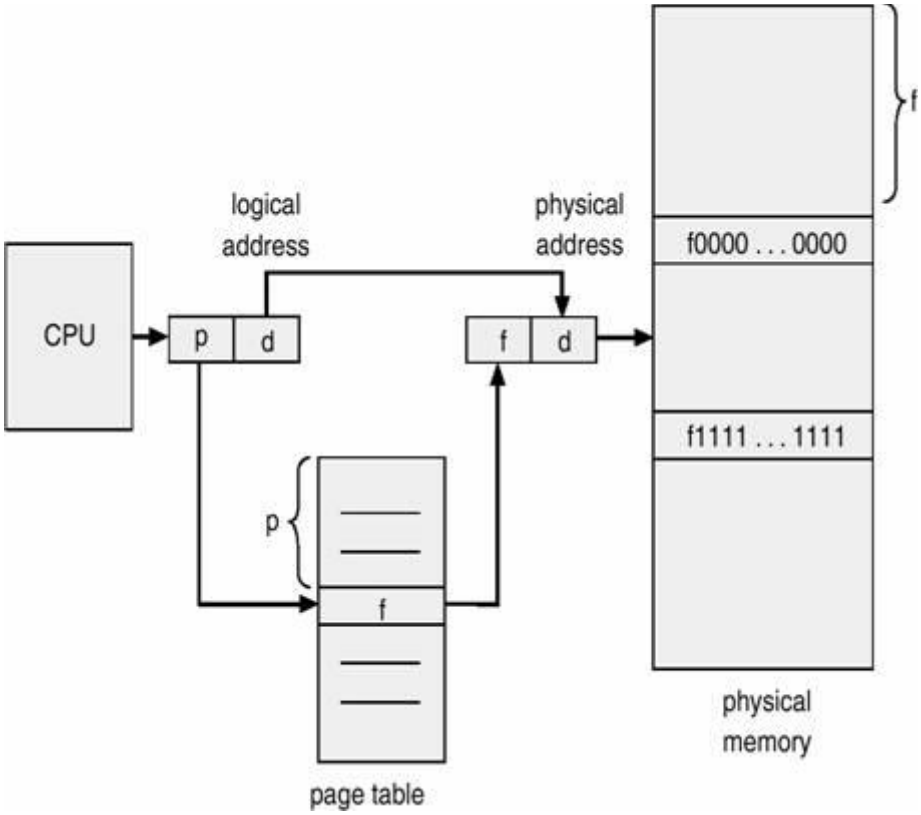


Fig. 3.11 Address translation in Paging

Every address generated by the CPU is divided into two parts: a **page number (p)** and a **page offset (d).** The page number is used as an index into a **page table.** The 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 to the memory unit.

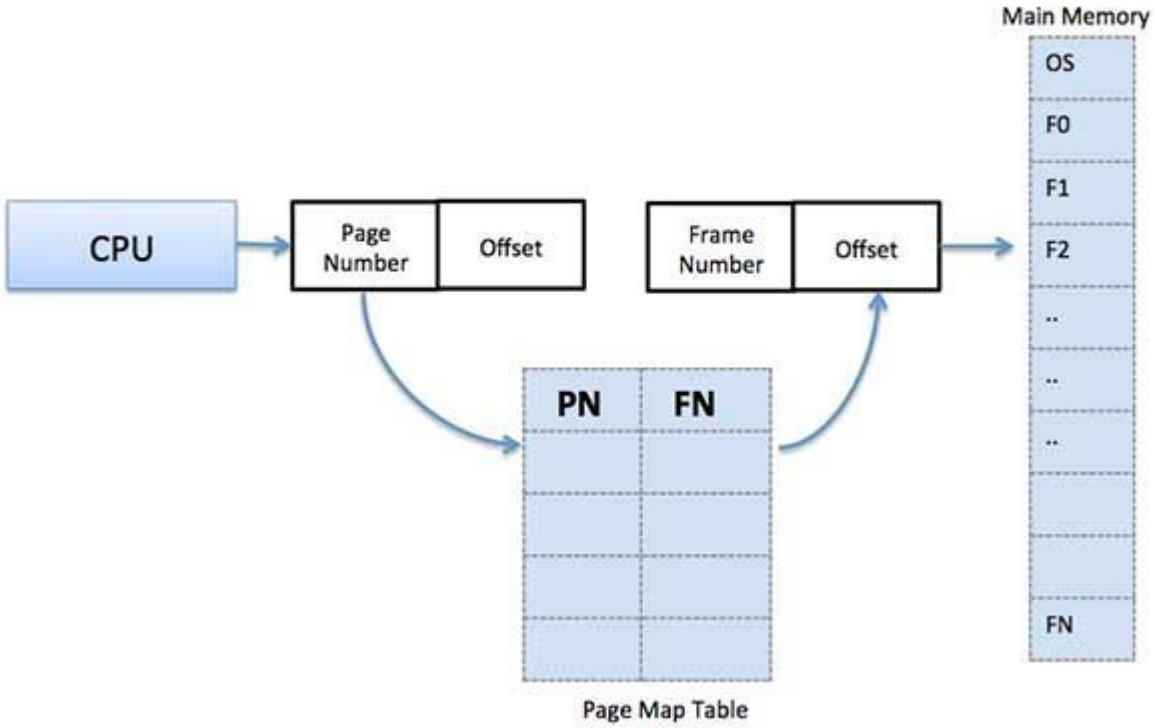
Page address is called **logical address** and represented by **page number** and the **offset**.

Logical Address = Page number + page offset

Frame address is called **physical address** and represented by a **frame number** and the **offset**.

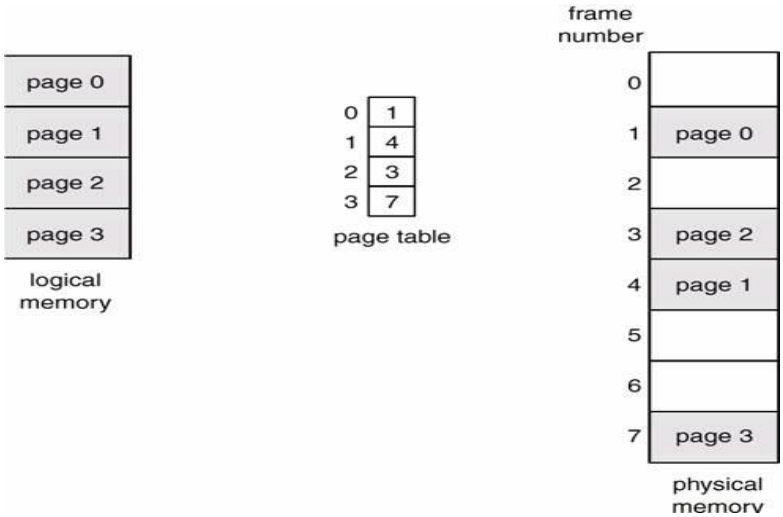
Physical Address = Frame number + page offset

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



The paging model of memory is shown below.

Paging Example 1:



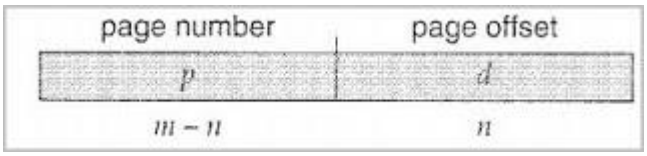
The page size (like the frame size) is defined by the hardware. The size of a page is typically a power of 2, varying between 512 bytes and 16 MB per page, depending on the computer architecture. The selection of a power of 2 as a page size makes the translation of a logical address into a page number and page offset particularly easy. If the size of logical address space is 2^m and a page size is 2^n addressing units (bytes or words), then the high-order *m* –

n bits of a logical address designate the page number, and the *n* low-order bits designate the page offset.

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

1. ***Page number (p)* – used as an index into a *page table* which contains base address of each page in physical memory.**

2. ***Page offset (d)* – combined with base address to define the physical memory address that is sent to the memory unit.**



where *p* is an index into the page table and *d* is the displacement within the page.

Fig 3.13 Logical address

Paging Example 2:

Assume a page size of 1K and a 15-bit logical address space. Given : Size of logical address = 2 ^ 15 = 32788 (m=15)

No. of bits required to address each byte within a 1024-byte page = 10 bits (n=10, 2^10 = 1024).

Given : Size of logical address = 2 ^ 15 = 32788 so, m=15 & n = 10

This leaves 5 bits for page number. (m-n)

So, No. of pages are in the system = 32 pages of 1K each (Since 2^5 = 32)

Page number (p) = m-n = 5 bits;

page offset (d) = 10 bits

Assuming a 15-bit address space with 8 logical pages. How large are the pages?

Answer: 2^12 = 4K. It takes 3 bits to reference 8 logical pages (2^3 = 8).

This leaves 12 bits for the page size and thus pages are 2^12.

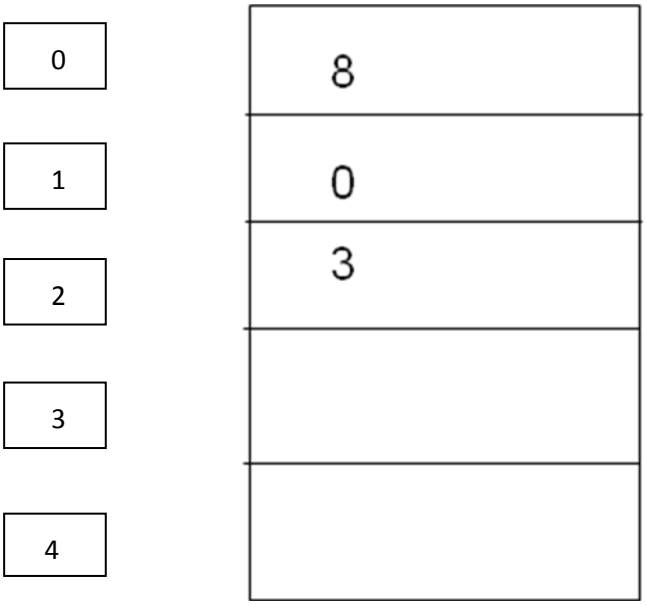
Consider logical address 2049 and the following page table for some process P0.

Assume a 15-bit address space with a page size of 1K.

What is the physical address to which logical address 2049 will be mapped?

Solution :

Logical Pages



Step 1. Convert logical address to binary:

Logical address: **2049** ⟶ **000100000000001**

Step2. Determine the logical page number:

Since there are 5-bits allocated to the logical page, the address is broken up as follows:

00010 **0000000001**

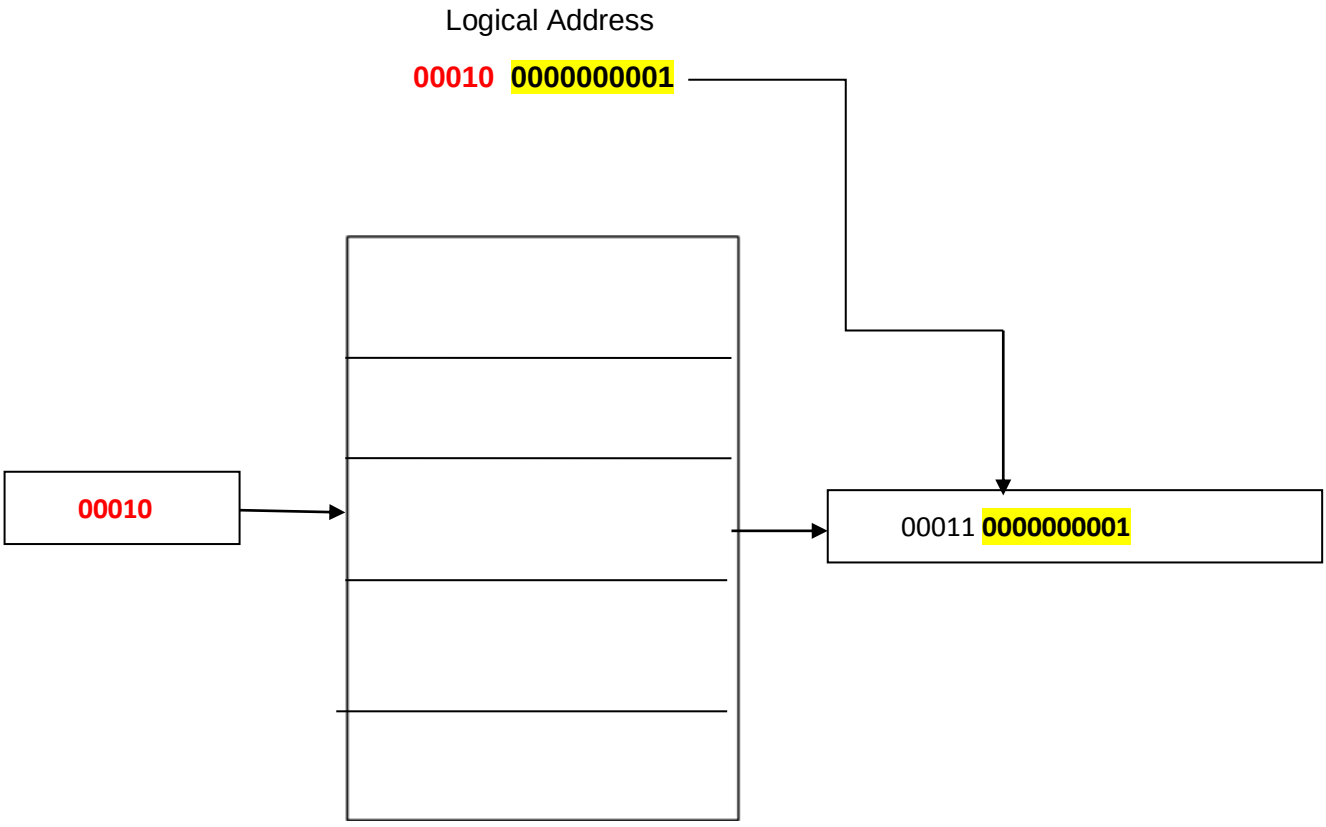
Logical page number offset within page

Step 3. Use logical page number as an index into the page table to get physical page number.

Logical Address:

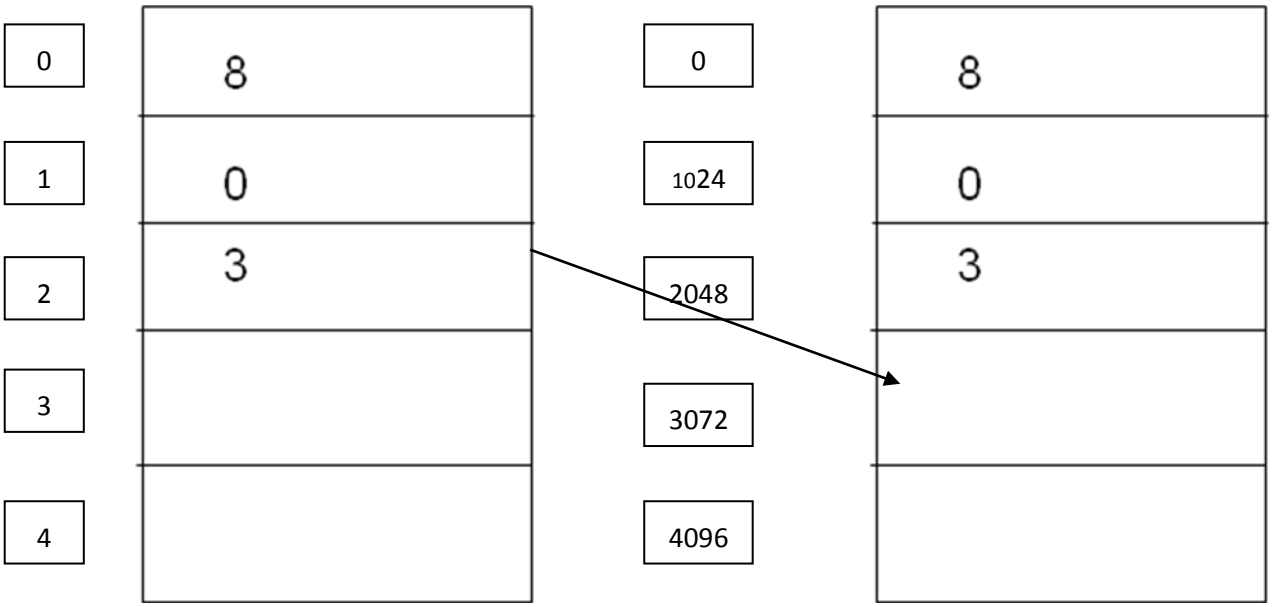
00010 0000000001

Step 4. Concatenate offset with physical page frame number



Step 5:

Logical address Physical address



000110000000001 = 3073

Logical address 2049 is mapped to ---------- > physical address 3073

3.1.6.3 Hardware support for paging:

The additional hardwares used in paging are :

- **i) Page table**
- **ii) *Translation look-aside buffers (TLB)***
- iii) **Page table is kept in main memory.**
- ***iv) Page-table base register (*PTBR) points to the page table.**
- ***v) Page-table length register* (PRLR) indicates size of the page table.**

The diagram below shows the Hardware support for Paging Hardware With TLB.

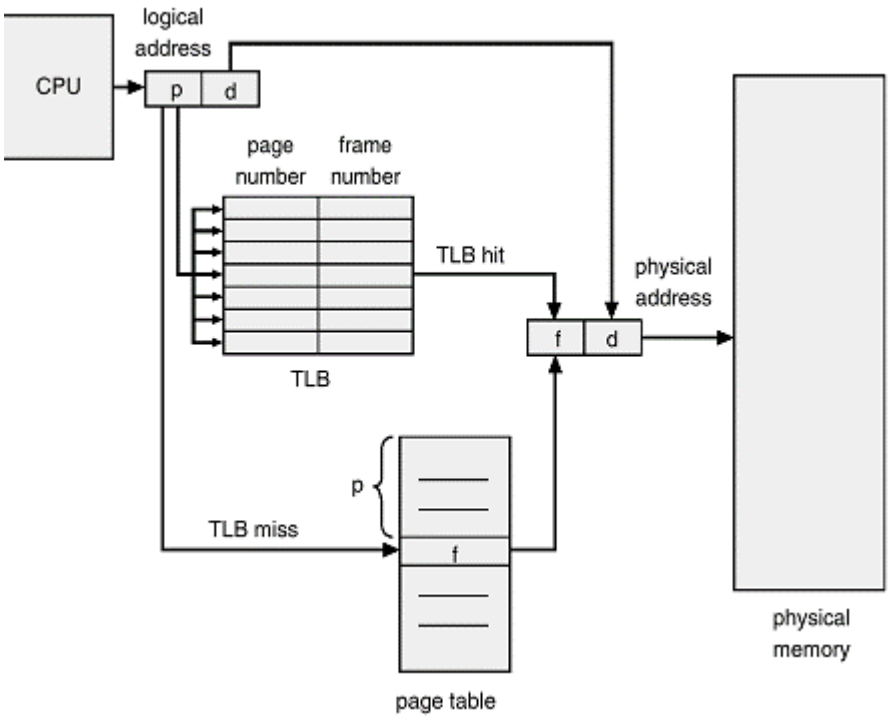


Fig 3.14 Paging using TLB

In this scheme every data/instruction access requires two memory accesses. One for the page table and one for the data/instruction. The two memory access problem can be solved by the use of a special fast-lookup hardware cache called *associative memory* or *translation look-aside buffers (TLBs)*

Working:

- If the page number is not in the TLB (TLB miss) a memory reference to the page table must be made.
- In addition, we add the page number and frame number into TLB
- If the TLB already full, the OS have to must select one for replacement
- Some TLBs allow entries to be wire down, meaning that they cannot be removed from the TLB, for example kernel codes
- The percentage of times that a particular page number is found in the TLN is called hit ratio
- If it takes 20 nanosecond to search the TLB and 100 nanosecond to access memory
- If our hit ratio is 80%, the effective memory access time equal to:
  0.8*(100+20) + 0.2 *(100+100)=140
- If our hit ratio is 98%, the effective memory access time equal:
  0.98*(100+20) + 0.02 *(100+100)=122

3.1.6.4 Protection and sharing

Paging hardware typically also contains protection mechanism by associating ***protection bit*** (or valid/invalid bit) with each frame. **Valid-invalid bit** attached to each entry in the page table:

1. “valid -v” indicates that the associated page is in the process’ logical address space, and is thus a legal page.

2. “invalid – i ” indicates that the page is not in the process’ logical address space.

3. Sharing code and data takes place if two page table entries in different processes point to same physical page, the processes share the memory. If one process writes the data, other process will see the changes. It is a very efficient way to communicate.

4. Sharing must also be controlled to protect modification and accessing data in one process by another process. Programs using procedures and data that are non-modifiable can be shared.

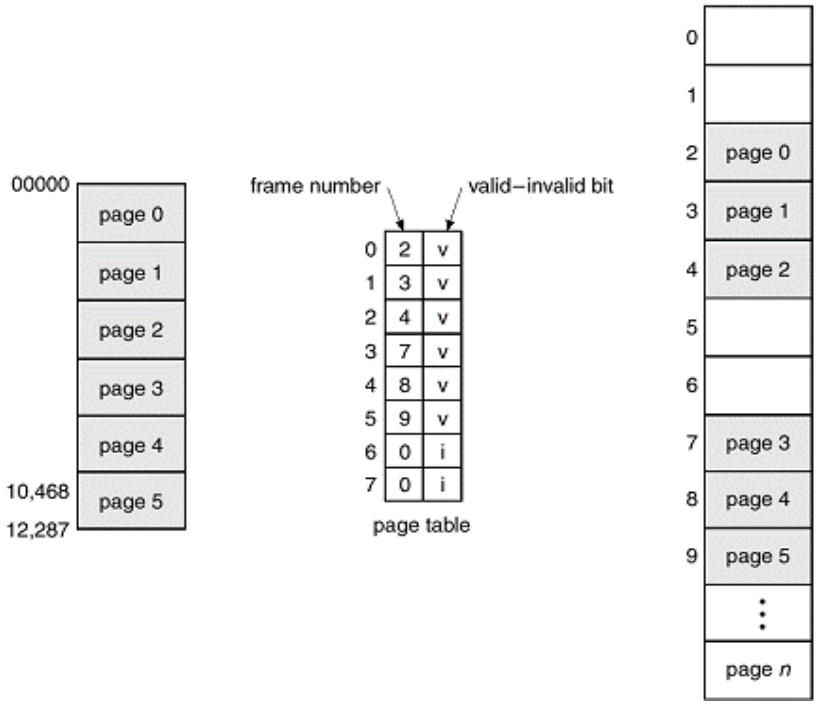


Fig 3.15 Valid (v) or Invalid (i) Bit in a Page Table

3.1.6.5 Advantages and Disadvantages of Paging

Advantages

- reduces external fragmentation, but still suffer from internal fragmentation.
- simple to implement and assumed as an efficient memory management technique.
- Due to equal size of the pages and frames, swapping becomes very easy.

Disadvantages

- extra resource consumption,
- memory overhead for storing page tables.
- translation overhead.

3.2 Virtual Memory

Introduction

The goal of various memory management techniques is to keep many processes in memory simultaneously to allow multiprogramming. However, they tend to require that an entire process be in memory before it can execute.

Virtual memory is a technique that allows the execution of processes that are not completely in memory. One major advantage of this scheme is that programs can be larger than physical memory. Further, virtual memory abstracts main memory into an extremely large, uniform array of storage, separating logical memory as viewed by the user from physical memory. This technique frees programmers from the concerns of memory-storage limitations. Virtual memory also allows processes to share files easily and to implement shared memory. In addition, it provides an efficient mechanism for process creation.

3.2.1 Basics of Virtual memory

A computer can address more memory than the amount physically installed on the system. This extra memory is actually called **virtual memory** and it is a section of a hard disk that's set up to emulate the computer's RAM. **Virtual memory** is a **memory** management capability of an **OS** that uses hardware and software to allow a computer to compensate for physical **memory** shortages by temporarily transferring data from random access **memory** (RAM) to disk storage. Virtual Memory is a space where large programs can store themselves in form of pages while their execution and only the required pages or portions of processes are loaded into the main memory. Thus we can have large virtual memory on a small physical memory.

Table 3.1 Virtual Memory Terminology

Virtual address	: The address assigned to a location in virtual memory to allow that location to be accessed as though it were part of main memory.
Virtual address space	: The virtual storage assigned to a process.
Address space	: The range of memory addresses available to a process.
Physical address	: The address of a storage location in main memory.

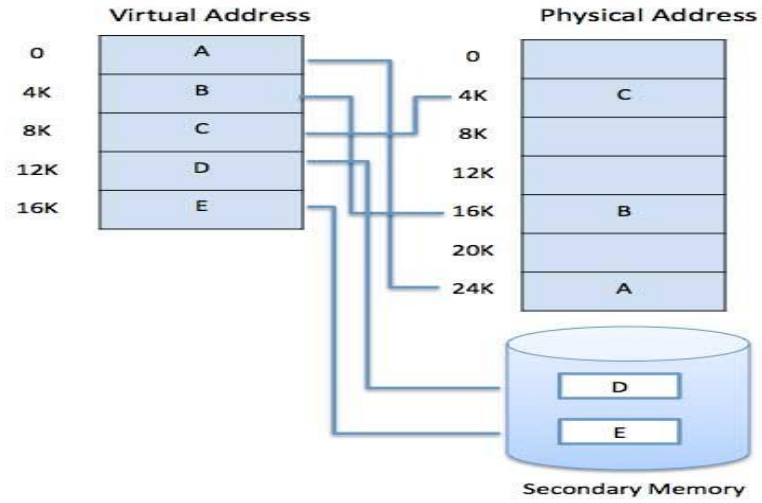


Fig. 3.16 Virtual memory Terminologies

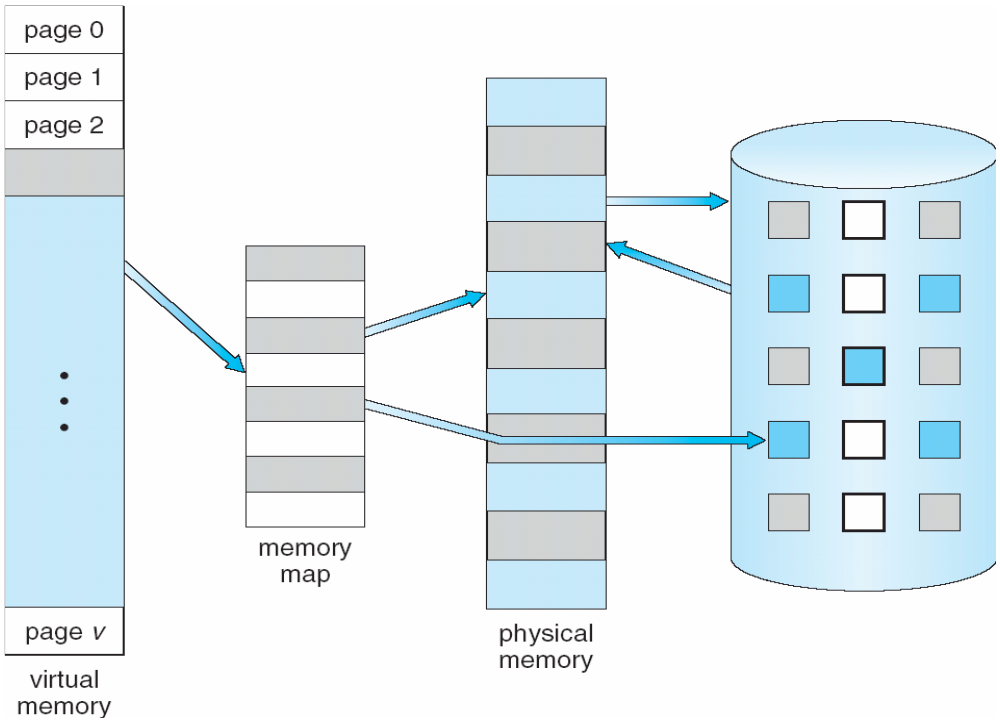


Fig . 3.17 Virtual Memory

Benefits of having Virtual Memory :

1. Programs can be larger than physical memory since virtual memory allows us to extend the use of physical memory by using disk.

2. It allows us to have memory protection, because each virtual address is translated to a physical address.
3. Less I/O required, leads to faster and easy swapping of processes.
4. Higher degree of multiprogramming is possible since only portions of programs are in memory

3.2.2 Hardware and Control Structures : Locality of Reference :

Many applications continually reference large amounts of data. Whenever the data are to be linked like primary or secondary memory references, database queries , the process become slow and this leads to poor performance in the application.

A common solution to such applications is ***Locality of Reference.*** This principle observes that an application does not access all of its data at once with equal probability. Instead, it accesses only a small portion of it at any given time. An application can exhibit temporal and/or spatial locality. If some data is referenced, then there is a high probability that it will be referenced again in the near future. This is called ***temporal locality***. If some data is referenced, then there is a high probability that data nearby will be referenced in the near future. This is called ***spatial locality.***

The O.S keeps more often used data in main memory, and everything else in secondary memory. Because of the principle of Locality of Reference, we can be sure that most memory references will be to locations already stored in main memory, thereby improving efficiency and providing a flat memory model. This scheme is used in modern operating systems and is called virtual memory. Virtual memory gives users the appearance of unlimited primary memory by transparently utilizing secondary memory.

Hardware and Control Structures :

The Virtual memory manager (VMM) maintains the following data structures to manage the virtual memory.

1. ***Page*** : Each program is divided into equal sized partitions called pages. It is a unit of transfer from program to memory and back. Each page is assigned a unique page number.
2. ***Page frame*** : The main memory is divided into equal sized partitions called page frames. Each partition is of the same size as a page of the program so that a page from the program can be accommodated in a page frame of the main memory. Each frame is assigned a unique page frame number. The VMM allocates page frames to incoming pages of the program.
3. ***Page table base register (PTBR)*** : This holds the base address for the page table of the current process. Each process running on a processor needs its own logical address space. Each process has its own page table. The operating system maintains information about each process in a process control block.The page table base address for the process is stored there. The operating system loads this address into the PTBR whenever a process is dispatched.

4. ***Page Table :*** Each running program, plus the data structures needed to manage it, is called a *process*. For every active process, the O.S assigns a page table. This table is used for recording the information about the page frames allocation to

the various pages brought in from the hard disk. and is used by a virtual memory system map between physical frames and virtual pages. Each page table entry contains information about a single page. The most important part of this information is a frame number — where the page is located in physical memory.

A page table consists of :

Page no : Number of the page brought in from the hard disk.

Fr***ame no.:*** Number of the page frame allotted from the main memory.

Valid bit (v): A valid bit (v) tells if the page is currently in main memory or if it must be retrieved from virtual memory. If the page is in main memory v is set to 1. When a page is taken from disc and put back into main memory, v is set to 0 and then it indicates a page fault.

Dirty Bit or Modified Bit (m) : . A Dirty or modifed bit (m) tells if a page has been written to while in main memory. If it hasn't been modified, m is set to 1. If it hasn't been modified, and a copy of it is in virtual memory, it doesn't need to be written to disc,hence the system speeds up. If it has modified, m is set to 0 and the page must be written to virtual memory.

5. Working set : This is the set of pages of the program which are currently active in the main memory. A process will never be executed unless its working set is resident in main memory. Pages outside the working set may be discarded at any time. The working set contains only pageable memory allocations; When a process references pageable memory that is not currently in its working set, a *page fault* occurs. The system page fault handler attempts to resolve the page fault and, if it succeeds, the page is added to the working set.

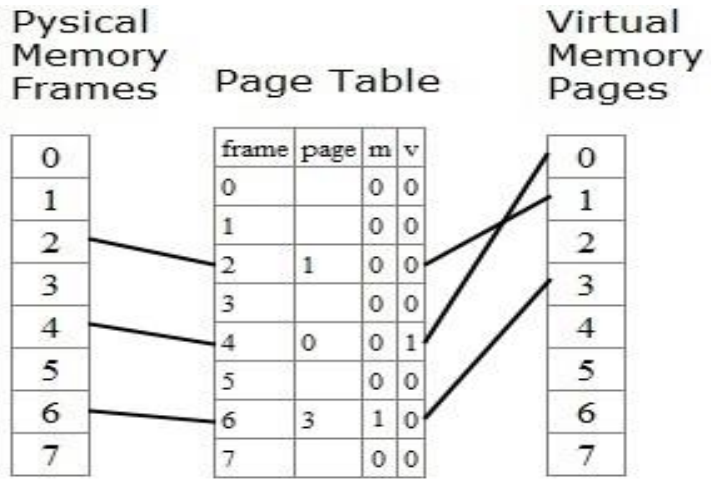


Fig. 3.18 Page Table Entries (PTE's) in Virtual memory

Page fault : *An interrupt that occurs when a program requests data that is not currently in real memory. The interrupt triggers the operating system to fetch the data from a virtual memory and load it into RAM.*

An invalid page fault or page fault error occurs when the operating system cannot find the data in virtual memory. This usually happens when the virtual memory area, or the table that maps virtual addresses to real addresses, becomes corrupt.

Principle of Working of Virtual Memory : Address Translation using Paging:

When the page is needed, the operating system copies it from disk to main **memory**, translating the **virtual** addresses into real addresses. The process of translating **virtual** addresses into real addresses is called mapping. The copying of **virtual** pages from disk to main **memory** is known as **paging** or swapping.

Virtual memory address translation uses page tables. These are simple arrays in memory indexed by page number. Address translation combines the frame number with the offset part of a logical address to form a physical address.The addresses that appear in programs are the virtual addresses or program addresses. For every memory access, either to fetch an instruction or data, the CPU must translate the virtual address to a real physical address. A virtual memory address can be considered to be composed of two parts: a page number and an offset into the page. The page number determines which page contains the information and the offset specifies which byte within the page. The size of the offset field is the log base 2 of the size of a page.

Consider an example system with:

16MB Maximum Virtual Address space (24 bits) ; 8MB Maximum Physical Address space (23 bits) ; 1024byte Page size (10 bits)

The virtual addresses can be represented as

13 bits	10 bits
page number	offset

To convert a virtual address into a physical address, the CPU uses the page number as an index into the page table. If the page is resident, the physical frame address in the page table is concatenated in front of the offset to create the physical address.

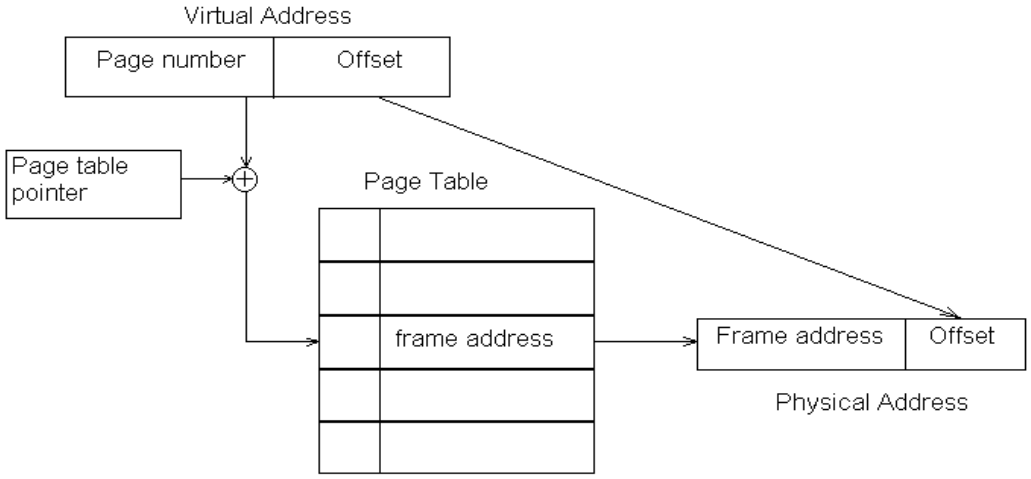


Fig. 3.19 Address Translation in a Paging System

3.2.3 Demand Paging

In virtual memory systems, demand paging is a type of *swapping* in which pages of data are not copied from disk to RAM until they are needed***. A demand paging system is a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance.*** When a context switch occurs, the operating system does not copy any of the old program's pages out to the disk or any of the new program's pages into the main memory Instead, it just begins executing the new program after loading the first page and fetches that program's pages as they are referenced.

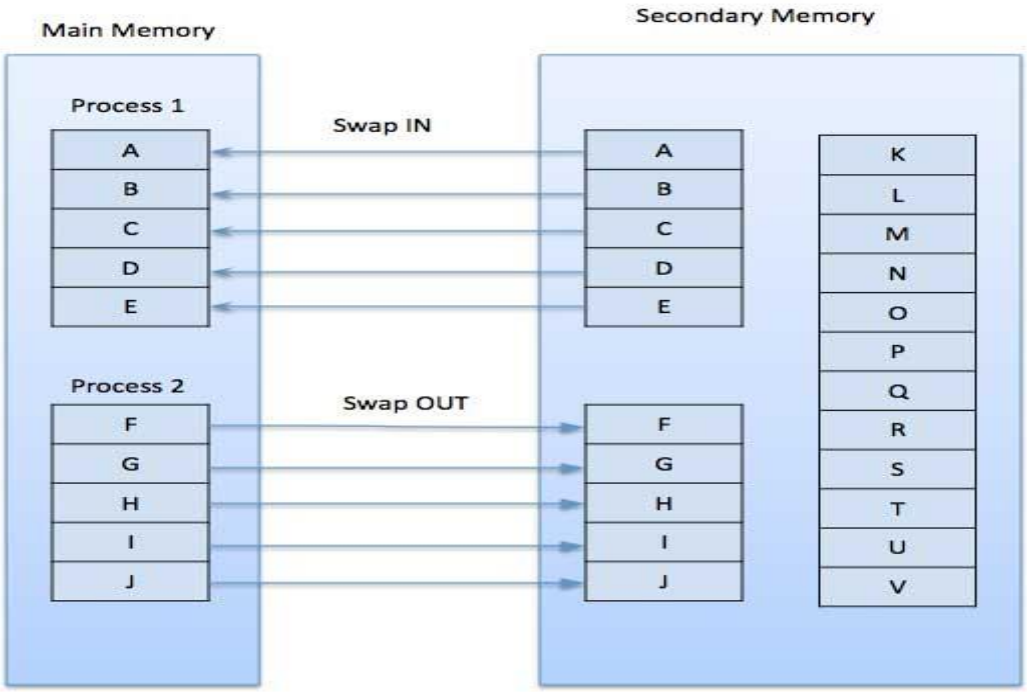


Fig 3.20 Demand Paging

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

Advantages

Following are the advantages of Demand Paging −

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

Disadvantages

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

3.2.4 Page Replacement Algorithms

In a computer operating system that uses paging for virtual memory management, **page replacement** algorithms are techniques that decide which memory pages to **page** out (swap out, write to disk) when a **page** of memory needs to be allocated.

Paging happens whenever a page fault occurs and a free page cannot be used for allocation purpose accounting to reason that pages are not available or the number of free pages is lower than required pages.

When the page that was selected for replacement and was paged out, is referenced again, it has to read in from disk, and this requires for I/O completion. This process determines the quality of the page replacement algorithm: the lesser the time waiting for page-ins, the better is the algorithm.

A page replacement algorithm looks at the limited information about accessing the pages provided by hardware, and tries to select which pages should be replaced to minimize the total number of page misses, while balancing it with the costs of primary storage and processor time of the algorithm itself. There are many different page replacement algorithms. We evaluate an algorithm by running it on a particular string of memory reference and computing the number of page faults,

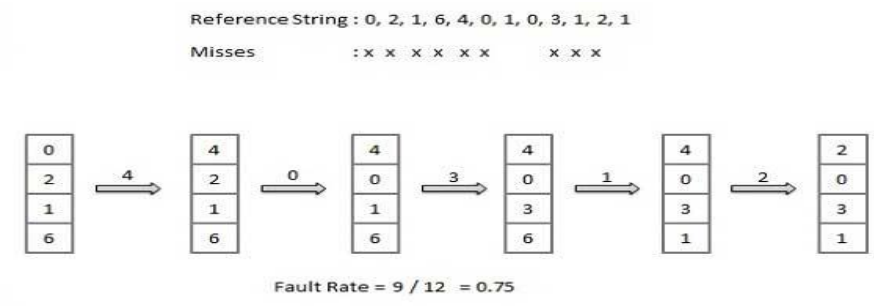
Reference String

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

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

3.2.4.1 First In First Out (FIFO) algorithm

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

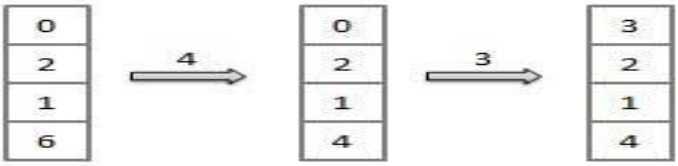


3.2.4.2 Optimal Page algorithm

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

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

Misses : x x x x x x



Fault Rate = 6 / 12 = 0.50

3.2.4.3 NRU(Not Recently Used) Page Replacement Algorithm -

This algorithm requires that each page have two additional status bits 'R' and 'M' called reference bit and change bit respectively. The reference bit(R) is automatically set to 1 whenever the page is referenced. The change bit (M) is set to 1 whenever the page is modified. These bits are stored in the PMT and are updated on every memory reference. When a page fault occurs, the memory manager inspects all the pages and divides them into 4 classes based on R and M bits.

- **Class 1: (0,0)** − neither recently used nor modified - the best page to replace.
- **Class 2: (0,1)** − not recently used but modified - the page will need to be written out before replacement.
- **Class 3: (1,0)** − recently used but clean - probably will be used again soon.
- **Class 4: (1,1)** − recently used and modified - probably will be used again, and write out will be needed before replacing it.

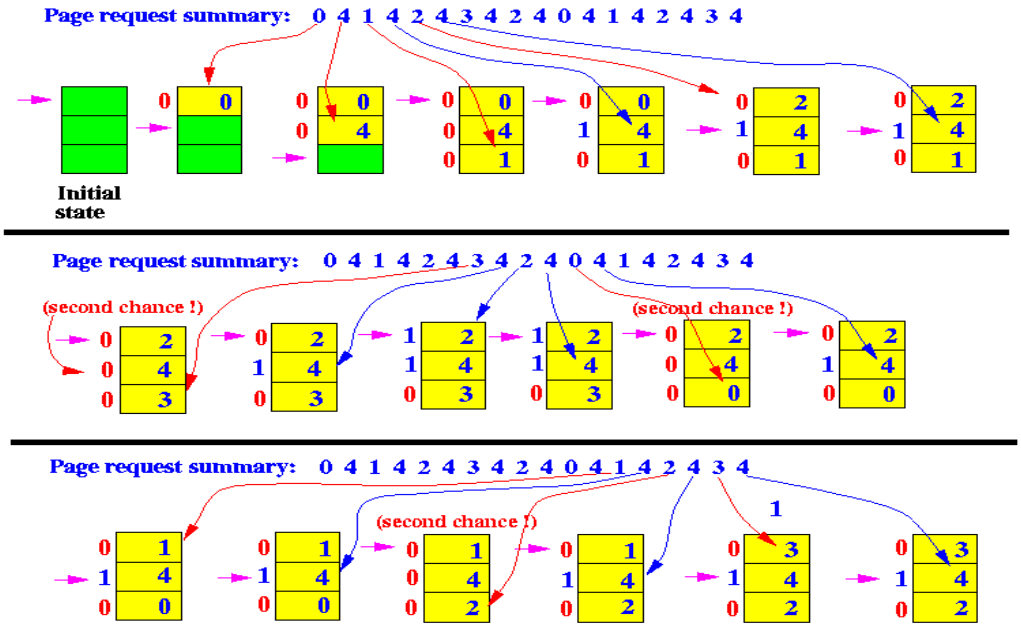
This algorithm removes a page at random from the lowest numbered non-empty class. The main attraction of NRU is that it is easy to understand, moderately efficient to implement.

3.2.4.4 The Second Chance Page Replacement

- The **Second Chance** replacement policy is called the **Clock** replacement policy...
- In the Second Chance page replacement policy, the pages for removal are **consider** in a round robin matter, and a page that has been **accessed between consecutive considerations** will not be replaced.
- Implementation:
    - Add a "second chance" bit to each memory frame.
    - Each time a memory frame is referenced, set the "second chance" bit to ONE (1) - this will give the frame a second chance...
    - A new page read into a memory frame has the second chance bit set to ZERO (0)

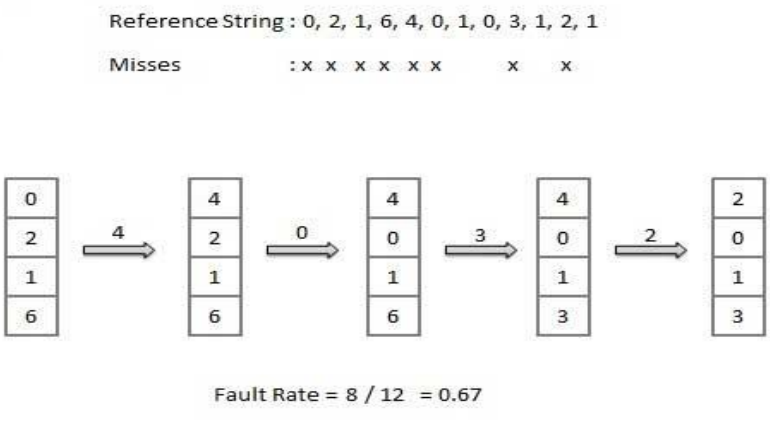
- When you need to find a page for removal, look in a round robin manner in the memory frames:
  - If the second chance bit is ONE, reset its second chance bit (to ZERO) and continue.
  - If the second chance bit is ZERO, replace the page in that memory frame.

- The following figure shows the behaviour of the program in paging using the Second Chance page replacement policy:



3.2.4.5 Least Recently Used (LRU) algorithm

- Page which has not been used for the longest time in main memory is the one which will be selected for replacement.
- Easy to implement, keep a list, replace pages by looking back into time.
- This algorithm suffers from the situation in which a page is used heavily during the initial phase of a process, but then is never used again.



Summary

- Memory management is the process of managing the computer memory which consist of primary memory or secondary memory.
- Basically computer memory is classified as Primary Memory (Main Memory),Secondary Memory (Auxillary Memory)
- Primary memory holds only those data and instructions on which computer is currently working.
- Secondary memory is used for storing data/Information permanently.
- The process of converting logical (virtual) address into physical address at run time is called memory mapping.

- Logical address otherwise called as Virtual address is an address used by software which is generated by the CPU
- Physical address actual memory address which denotes a memory area in the storage device.
- Memory allocation is a process by which computer programs and services are assigned with physical or virtual memory space
- Memory allocation methods: Contiguous allocation,Fixed partition allocation,Variable partition allocation
- Fragmentation is called when as processes are loaded and removed from memory, the free memory space is broken into little pieces which will not be able to use.
- Fragmentation is of two types---. Internal fragmentation , External fragmentation
- Internal fragmentation is the space wasted inside of allocated memory blocks because of restriction on the allowed sizes of allocated blocks
- External Fragmentation happens when a dynamic memory allocation algorithm allocates some memory and a small piece is left over that cannot be effectively used.
- Memory compaction is the process of moving allocated objects together, and leaving empty space together to avoid fragmentation.
- Paging is a memory management technique in which process (logical) address space is broken into blocks of the same size called pages (size is power of 2, between 512 bytes and 8192 bytes.)
- A data structure called page map table is used to keep track of the relation between a page of a process to a frame in physical memory.
- The additional hardwares used in paging are :Page table ,Translation look-aside buffers (TLB) , Page table is kept in main memory, Page-table base register (PTBR) points to the page table. Page-table length register (PRLR) indicates size of the page table.
- Virtual memory is a memory management capability of an OS that uses hardware and software to allow a computer to compensate for physical memory shortages by temporarily transferring data from random access memory (RAM) to disk storage
- When the page is needed, the operating system copies it from disk to main memory, translating the virtual addresses into real addresses. The process of translating virtual addresses into real addresses is called mapping. The copying of virtual pages from disk to main memory is known as paging or swapping.
- A demand paging system is a paging system with swapping where processes reside in secondary memory and pages are loaded only on demand, not in advance

- page replacement algorithms are techniques that decide which memory pages to page out (swap out, write to disk) when a page of memory needs to be allocated.
- The different algorithms are --First In First Out (FIFO) algorithm, Optimal Page algorithm, NRU(Not Recently Used) Page Replacement Algorithm , The Second Chance Page Replacement , Least Recently Used (LRU) algorithm

REVIEW QUESTIONS

PART A (2 Marks)

1. What are the main tasks of Memory management.
2. List the two types of memory available in a computer system.
3. Define : Physical address.
4. Define : Logical address.
5. What are the features of Main memory?
6. What is a secondary memory?
7. What is the use of memory mapping hardware?
8. What do you mean by memory allocation?
9. What is Internal fragmentation?
10. what is External fragmentation?
11. What is Memory compaction?
12. What is Paging?
13. Define address translation?
14. What is protection?
15. Define Virtual memory?
16. What is Demand paging?
17. What is the use of dirty bit?
18. What is Working set?
19. Define Page fault.
20. What is Page replacement?

PART B (3 Marks)

1. Explain Logical and physical address mapping.
2. Draw the structure of contiguous memory allocation.
3. Give the merits and demerits of variable partitioned allocation in memory.
4. What is basic concept of paging?
5. What are the components of page map table?
6. List the advantages of paging.
7. What is memory sharing?
8. Give the concept of page replacement policy?

PART C (5 Marks)

1. Briefly explain the logical and physical address map .
2. Explain fixed and variable partitioned allocation.

3. Discuss Internal and External fragmentation with example.
4. With diagram , explain the concept of compaction.
5. Discuss with example , the principle of paging.
6. Discuss various protection and sharing mechanisms.
7. Explain the hardware needed in paging with diagram.
8. Discuss the basic concept of Demand paging with diagram.
9. Write notes on : page frame, page fault, working set and dirty bit.
10. Explain any two page replacement policy.

UNIT IV
I/O AND FILE MANAGEMENT, SECURITY & PROTECTION

***Objectives*:**

- Understand the structure and performance of disk drives.
- Discuss about Disk scheduling and its various algorithms.
- Explain the concept of RAID and describe the various levels.
- Describe the basic concepts of files and file systems.
- Study Directory structures.
- Understand the principal techniques for file organization and access methods.
- Define Disk formatting.
- To explore file-system security and protection through various mechasisms.

Introduction

Computers operate a great many kinds of I/O devices. *General categories are storage devices (disks, tapes), transmission devices (network cards, modems), and human-interface devices (screen, keyboard, mouse).* I/O management deals with the control of I/O devices connected to the computer and this is a major concern of operating-system since I/O devices vary so widely in their function and speed .

The most important form of I/O is the Magnetic disk I/O. Since there is a vast difference in speed between the Disks and processor/main memory which slows down the performance, disk management enforce various disk scheduling policies to improve it.

In most applications, the file is the central element. And the input to the application is by means of a file; but in all applications, output is saved in a file for long term storage which is later accessed by the user and programs. The file system permits users to create data collections, called files and access files through the file management system. The main objective for a file management system is to meet the data management needs and requirements of the user, which include storage of data and the ability to perform the various file operations.

Protection mechanisms control access to a system by limiting the types of file access permitted to users. Protection ensure that only authorised processes can operate on memory segments, the CPU, and other resources.

Security ensures the authentication of system users to protect the integrity of the information stored in the system (both data and code), as well as the physical resources of the computer system.The security system prevents unauthorized access, malicious destruction or alteration of data.

4.1 Disk Management

- Disks provide the bulk of secondary storage for modern computer systems. Magnetic tape was used as an early secondary storage medium, but the access time is much slower than for disks.
- Modern disk drives are addressed as large one dimensional arrays of logical blocks, where the logical block is the smallest unit of transfer. The size of a logical block is usually 512 bytes, although some disks can be low level formatted to choose a different logical block size, such as 1024 bytes. The one dimensional array of logical blocks is mapped onto the sectors of the disk sequentially.

4.1.1 Disk Structure

Hard disks have the following basic structure:

- Hard disks drives are organized as a concentric stack of disks or platters' which rotate on about a central spindle.
- Each platter has 2 working surfaces.
- Platter is made from aluminum, ceramic, or class, coated with a magnetic materials such as iron oxide on both sides.
- Each working surface is divided into a number of concentric rings called ***tracks.*** The collection of all tracks that are the same distance from the edge of the platter, (i.e. all tracks immediately above one another in the following diagram) is called a ***cylinder***.
- Each track is further divided into ***sectors,*** each containing 512 bytes of data , although some modern disks use larger sector sizes.

- The data on a hard drive is read by read-write ***heads.*** The standard configuration uses one head per surface, each on a separate ***arm***, and controlled by a common ***arm assembly*** which moves all heads simultaneously from one cylinder to another.
- Each storage unit on a disk can be identified by a 3-coordinate system.
  1. Cylinder (C) 2. Head/Side (H) 3. Sector (S)
- During operation the disk rotates at high speeds such as 7200 rpm etc.,

Disk Capacity and speed :

The storage capacity of a disk drive =

No. of working surfaces(or heads) x No. of tracks per surface x No. of sectors per track x No. of bytes per sector.

E.g. 12,495 cylinders x16 heads x 63 sectors x 512 bytes = approx. 6GB

Disk speed has the following three parameters :

- The **transfer rate** is the rate at which data flow between the drive and the computer.
- The **positioning time,** sometimes called the **access time**, consists of the time to move the disk arm to the desired cylinder, called the **seek time**, and the time for the desired sector to rotate to the disk head, called the **rotational latency.**
- The **disk bandwidth** 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.

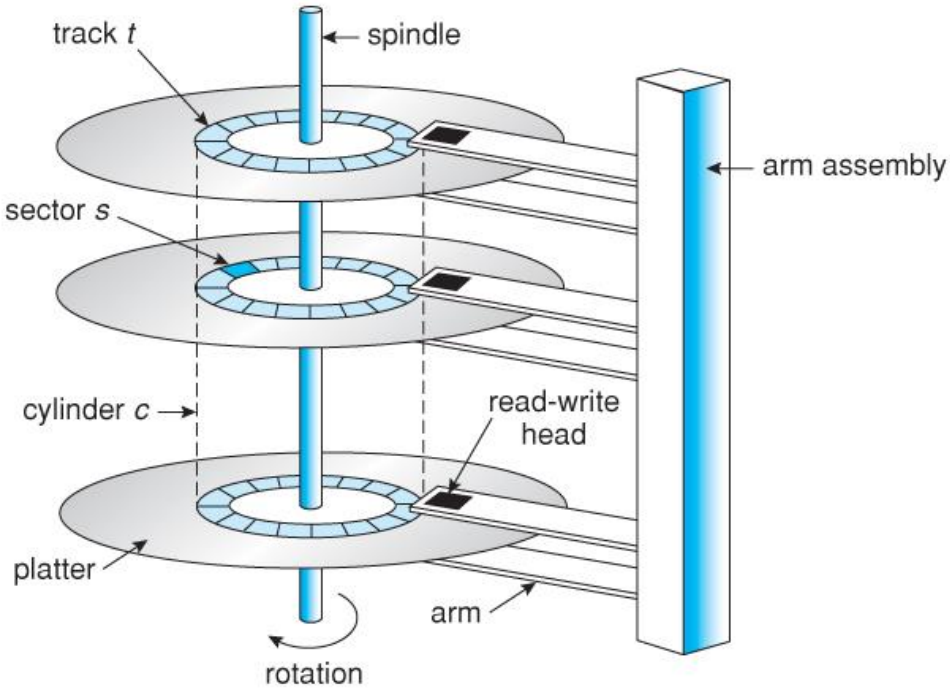


Figure 4.1 – Hard disk Moving-head disk mechanism.

4.1.2 Disk scheduling and its algorithms

In multiprogramming systems several different processes may want to use the system's resources simultaneously.The important task of the operating system is to use the disk drives efficiently. When the disk is in use, a drive motor spins it at high speed. Most drives rotate 60 to 200 times per second.

4.2.1.1 **Disk scheduling**

Disk scheduling *is done by operating systems to schedule I/O requests arriving for disk. Disk scheduling is also known as I/O scheduling. In this process, I/O requests are serviced in such a way to minimize the movement of the read/write head and minimizing the seek time of the disk drive.*

Need for Disk scheduling:

- Multiple I/O requests may arrive by different processes and only one I/O request can be served at a time by disk controller. Thus other I/O requests need to wait in waiting queue and need to be scheduled.
- Two or more request may be far from each other so can result in greater disk arm movement.
- Hard drives are one of the slowest parts of computer system and thus need to be accessed in an efficient manner.

Disk Drive Performance parameters:

The Disk's speed and performance are based on the following parameters which categorises the type of Disk Scheduling Algorithms.

- **Seek Time:**Seek time is the time taken to locate the disk arm to a specified track where the data is to be read or write. So the disk scheduling algorithm that gives minimum average seek time is better.
- **Rotational Latency:** Rotational Latency is the time taken by the desired sector of disk to rotate into a position so that it can access the read/write heads. So the disk scheduling algorithm that gives minimum rotational latency is better.
- **Transfer Time:** Transfer time is the time to transfer the data. It depends on the rotating speed of the disk and number of bytes to be transferred.
- **Disk Access Time:** Disk Access Time = Seek Time + Rotational Latency + Transfer Time
- **Disk Response Time:** Response Time is the average of time spent by a request waiting to perform its I/O operation. *Average Response time* is the response time of the all requests. *Variance Response Time* is measure of how individual request are serviced with respect to average response time. So the disk scheduling algorithm that gives minimum variance response time is better.

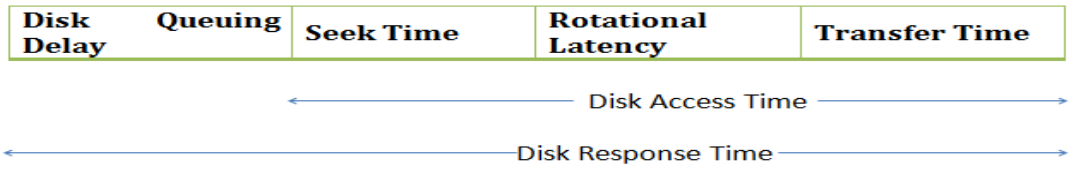


Fig 4.2 Disk performance parameters

4.1.2.2 Disk Scheduling Algorithms

Whenever a process needs I/O to or from the disk, it issues a system call to the operating system. If the desired disk drive and controller are available, the request can be serviced immediately. If it is busy, any new requests for service will be placed in the pending queue. For a multiprogramming system with many processes, the disk queue may often have several pending requests. Thus, when one request is completed, the operating system

chooses which pending request to service next. For this, the operating system use the following various disk-scheduling algorithms.

The following disk scheduling algorithms are used to reduce the seek time of all requests and hence the head movement:

i) First Come-First Serve (FCFS) ii) Shortest Seek Time First (SSTF) iii) Elevator (SCAN) iv) Circular SCAN (C-SCAN) v) LOOK vi) C-LOOK

To explain the above algorithms, let us take the following example:

Consider a disk queue with requests for I/O to blocks on cylinders as follows:

- **Disk Queue**: 23, 89, 132, 42, 189
- There are 200 cylinders numbered from 0 - 199
- The disk head stars at number 100

i) FCFS Scheduling: (First-Come, First-Served)

This is the simplest form of disk scheduling . In the first-come, first-served (FCFS) algorithm, requests are processed in the order that they arrive. This is very easily implemented with a FIFO queue; when Processes come in, they are put at the end of the queue.

The order of processes after serviced as per FCFS : 100,23,89,132,42,189

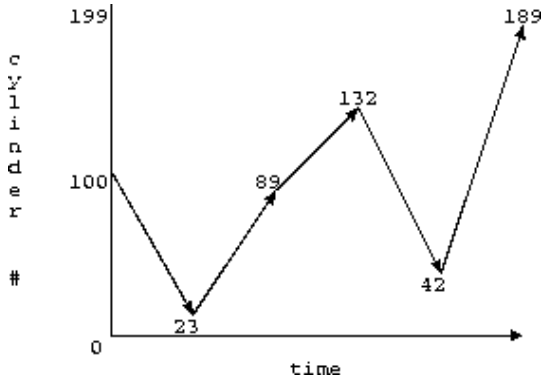


Fig. 4.3 FCFS scheduling

Total time is estimated by total arm motion :

| 100-23|+|23-89|+|89-132|+|23-132|+|132-42|+|42-189|=77+66+43+109+90+147 = 532

Advantages: 1) Every request gets a fair chance
2) No indefinite postponement

Disadvantages: 1) Does not try to optimize seek time
2) May not provide the best possible service

ii) SSTF Scheduling: (Shortest Seek Time First)

In SSTF (Shortest Seek Time First), requests having shortest seek time are executed first. So, the seek time of every request is calculated in advance in queue and then they are scheduled according to their calculated seek time. As a result, the request near the disk arm will get executed first. SSTF is certainly an improvement over FCFS.

The order of processes after serviced as per SSTF : 100,89,132,189,42,23

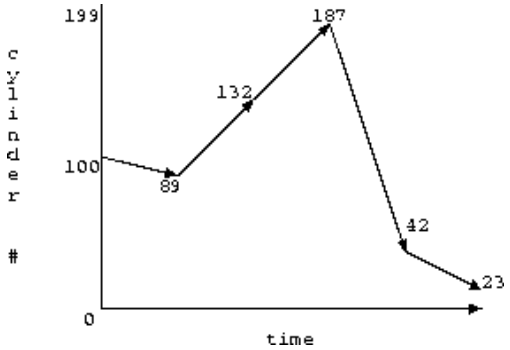


Fig. 4.4 SSTF scheduling

Total time is estimated by total arm motion :

$$| 100 - 89 | + | 89 - 132 | + 132 - 187 | + | 187 - 42 | + | 42 - 23 | = 11 + 43 + 55 + 145 + 19 = 273$$

Advantages: 1) Average Response Time decreases
2) Throughput increases

Disadvantages: 1) Overhead to calculate seek time in advance
2) Can cause Starvation for a request if it has higher seek time as compared to incoming requests
3) High variance of response time as SSTF favours only some requests.

iii) SCAN: Elevator Algorithm

In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works like an elevator and hence also known as **elevator algorithm.** As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Assume we are going inwards (i.e., towards 0), we have

The order of processes after serviced as per SCAN : 100,89,42,23,132,187

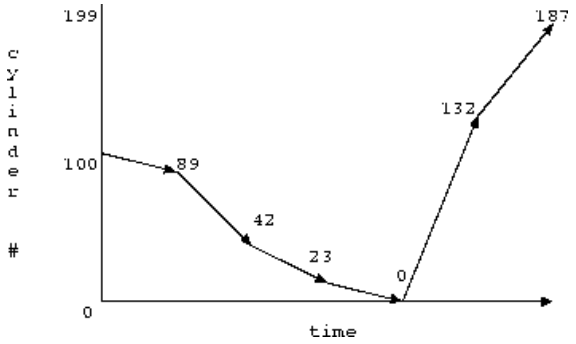


Fig 4.5 SCAN Scheduling

Total time is estimated by total arm motion :

| 100-89|+|89-42|+|42-23|+|23-0|+|0-132|+|132-189|=11+47+19+23+132+57 = 257

Advantages: 1) High throughput.

2) Low variance of response time.
3) Average response time.

Disadvantages: 1) Long waiting time for requests for locations just visited by disk arm.

iv) C-SCAN: (Circular –SCAN)

Circular scanning works just like the elevator to some extent. In circular SCAN algorithm, when the edge of the disc is reached, it returns to the opposite edge without dealing with any requests, and then starts again from there. This provides a slight speedup over the SCAN algorithm, and is thus preferable to it. The C-SCAN scheduling algorithm essentially treats the cylinders as a circular list that wraps around from the final cylinder to the first one.

The order of processes after serviced as per C-SCAN : 100,89,42,23,199,187,132

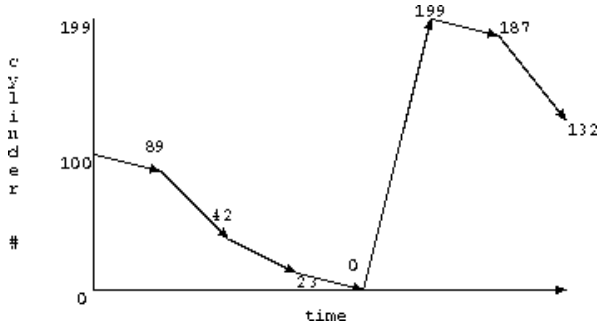


Fig 4.6 C-Scan scheduling

|100-89| + |89-42| + |42-23| + |23-0| + |0-199| + |199-187|+|187-132|=11+47+19+23+199+12= 311

Advantages: Provides more uniform wait time compared to SCAN

v) LOOK:

It is similar to the SCAN disk scheduling algorithm except the difference that the disk arm in spite of going to the end of the disk goes only to the last request to be serviced in front of the head and then reverses its direction from there only. Thus it prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

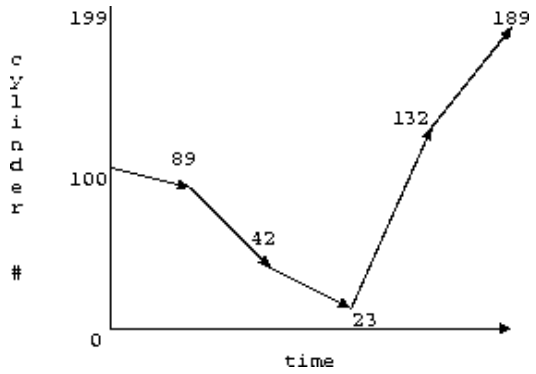


Fig.4.7 Look Scheduling

$$| 100 - 89 | + | 89 - 42 | + | 42 - 23 | + | 23 - 132 | + | 132 - 187 | = 11 + 47 + 19 + 109 + 55 = 241$$

vi) CLOOK: As LOOK is similar to SCAN algorithm, in similar way, CLOOK is similar to CSCAN disk scheduling algorithm. In CLOOK, the disk arm inspite of going to the end goes only to the last request to be serviced in front of the head and then from there goes to the other end's last request. Thus, it also prevents the extra delay which occurred due to unnecessary traversal to the end of the disk.

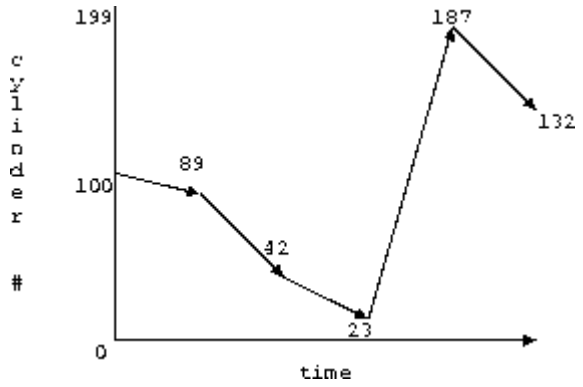


Fig. 4.8 C-LOOK scheduling

$$| 100 - 89 | + | 89 - 42 | + | 42 - 23 | + | 23 - 187 | + | 187 - 132 | = 11 + 47 + 19 + 164 + 55 = 296$$

4.1.3 RAID Technology

RAID stands for **R**edundant **A**rray of **I**nexpensive (or sometimes "Independent") **D**isks.

RAID is a method of combining several hard disk drives into one logical unit. RAID arrays appear to the operating system as a single logical hard disk. Here the disk drives are independent, and are multiple in number.

The main advantage of RAID, is that RAID is fault tolerant. In most of the RAID level's data is redundant in multiple disks, so even if one disk fails, or even two sometimes, the data will be safe and the operating system will not be even aware of the failure. DATA loss is prevented due to the fact that data can be recovered from the disk that are not failed.

Characteristics used in RAID are:

- Stripping
- Mirroring
- Parity

1. Striping :

RAID is collection of multiple disk's and in these disk ,predefined number of contiguously addressable disk blocks are defined which are called as *strips* and collection of such strips in aligned in multiple disk is called *stripe*.

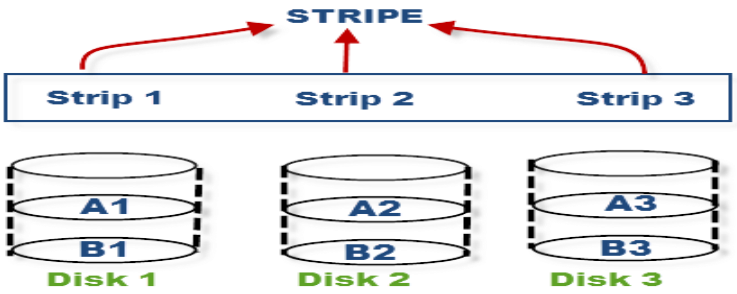


Fig. 4.9 Striping

Hard disk is a collection of multiple addressable blocks and these blocks are stacked together and called strip and multiple such hard disks are placed parallel or serially. Then such combination of disk is called **stripe**.

2. Mirroring :

Mirroring is a mechanism in which the same data is written to another disk drive. "*mirroring*" is simply a pair of disk drives which store duplicate data, but appear to the computer as a single drive; the number of drives in the array will always be an ***even*** number. The main advantage of mirroring(multiple sets of same data on two disks), is that it provides 100 percent redundancy.

Suppose there are two drives in mirroring mode, then both of them will contain an exact same copy of data. So even if one disk fails, the data is safe on the other.

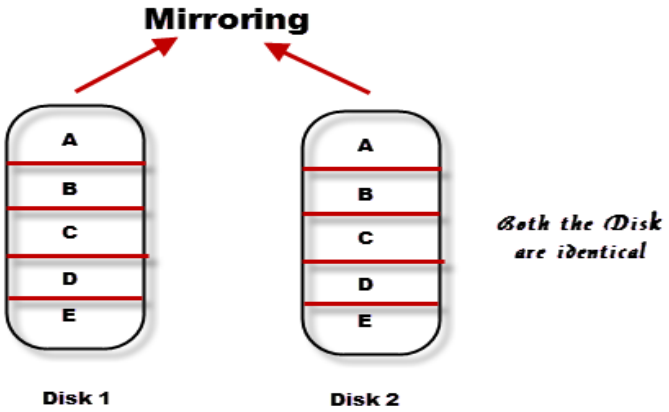


Fig. 4.10 Mirroring

When the failed disk is replaced with a new disk, the controller copies the data from the surviving disk of the **mirrored pair**. Data is simultaneously recorded on both the disk. Though this type of RAID gives you highest availability of data but it is costly as it requires double amount of disk space and thus increasing the cost.

3. Parity :

Mirroring involves high cost, so to protect the data new technique is used with striping called parity. This is reliable and **low cost solution for data protection**. In this method and additional HDD or disk is added to the stripe width to hold parity bit.

Parity is a redundancy check that ensures full protection of data without maintaining a full set of duplicate data.

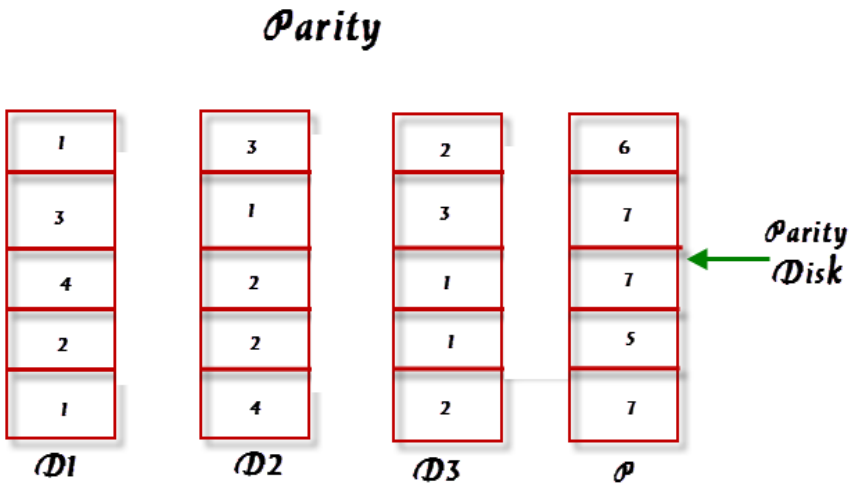


Fig.4.11 Parity

The parity bits are used to re-create the data at the time of failure. Parity information can be stored on separate, **dedicated HDDs** or distributed across all the drives in a RAID set. In the above image, parity is stored on a separate disk.

The first three disks, labeled D, contain the data. The fourth disk, labeled P, stores the parity information, which in this case is the sum of the elements in each row. Now, if one of the Disks (D) fails, the missing value can be calculated by subtracting the sum of the rest of the elements from the parity value.

Standard RAID levels

RAID 0: In a RAID 0 system data are split up in blocks that get written across all the drives in the array. By using multiple disks (at least 2) at the same time, this offers superior I/O performance. This configuration has striping but no redundancy of data. It offers the best performance but no fault-tolerance.

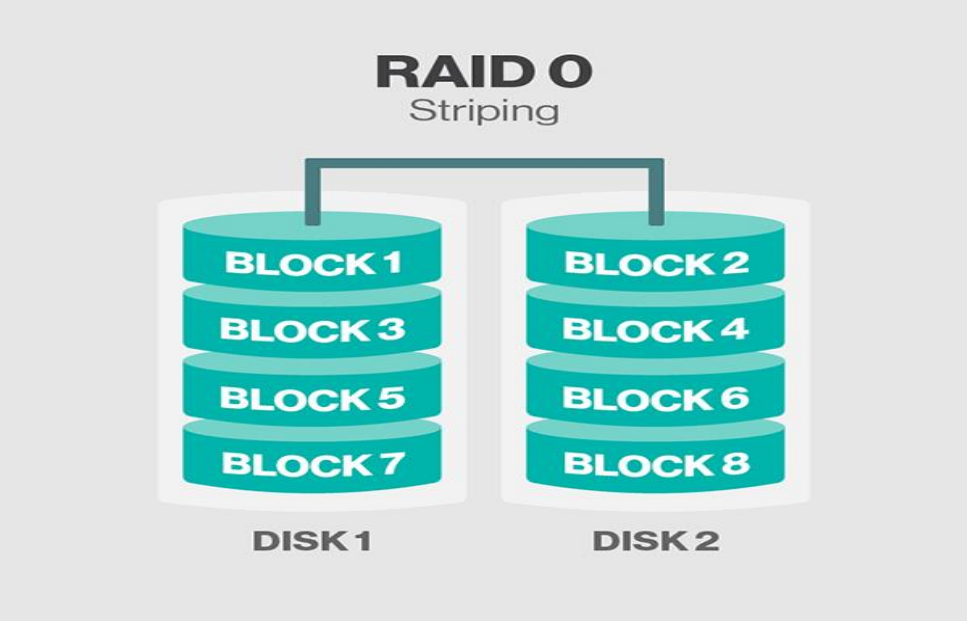


Fig 4.12 RAID 0

Advantages

- RAID 0 offers great performance, both in read and write operations. There is no overhead caused by parity controls.
- All storage capacity is used, there is no overhead.
- The technology is easy to implement.

Disadvantages

- RAID 0 is not fault-tolerant. If one drive fails, all data in the RAID 0 array are lost. It should not be used for mission-critical systems.

RAID 1: Also known as *disk mirroring.* Data are stored twice by writing them to both the data drive (or set of data drives) and a mirror drive (or set of drives) . If a drive fails, the controller uses either the data drive or the mirror drive for data recovery and continues operation. You need at least 2 drives that duplicate the storage of data for a RAID 1 array.There is no striping. Read performance is improved since either disk can be read at the same time. Write performance is the same as for single disk storage.

In a RAID 0 system data are split up in blocks that get written across all the drives in the array. By using multiple disks (at least 2) at the same time, this offers superior I/O performance.

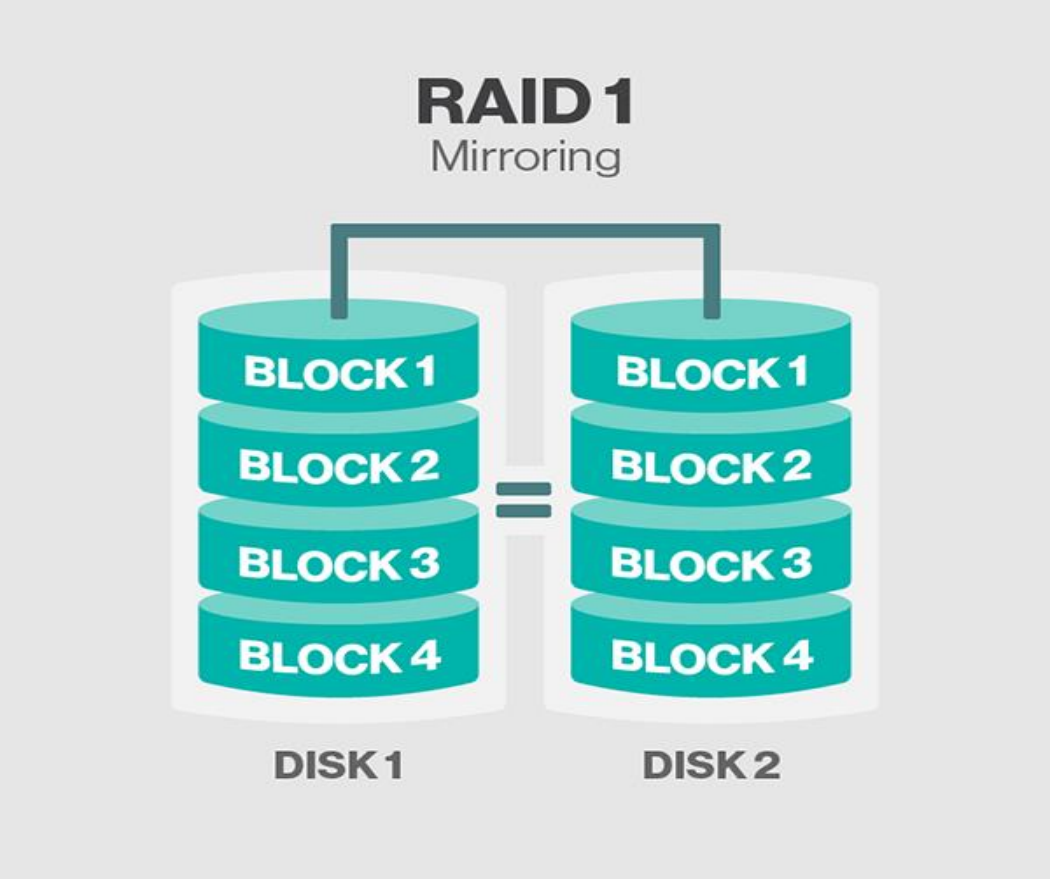


Fig 4.13 RAID 1

Advantages

- RAID 1 offers excellent read speed and a write-speed that is comparable to that of a single drive.
- In case a drive fails, data do not have to be rebuild, they just have to be copied to the replacement drive.
- RAID 1 is a very simple technology.

Disadvantages

- The main disadvantage is that the effective storage capacity is only half of the total drive capacity because all data get written twice.
- Software RAID 1 solutions do not always allow a hot swap of a failed drive. That means the failed drive can only be replaced after powering down the computer it is attached to. For servers, this is not suitable .

RAID 2: This configuration uses striping across disks with some disks storing error checking and correcting (ECC) information. It has no advantage over RAID 3 and is no longer used.

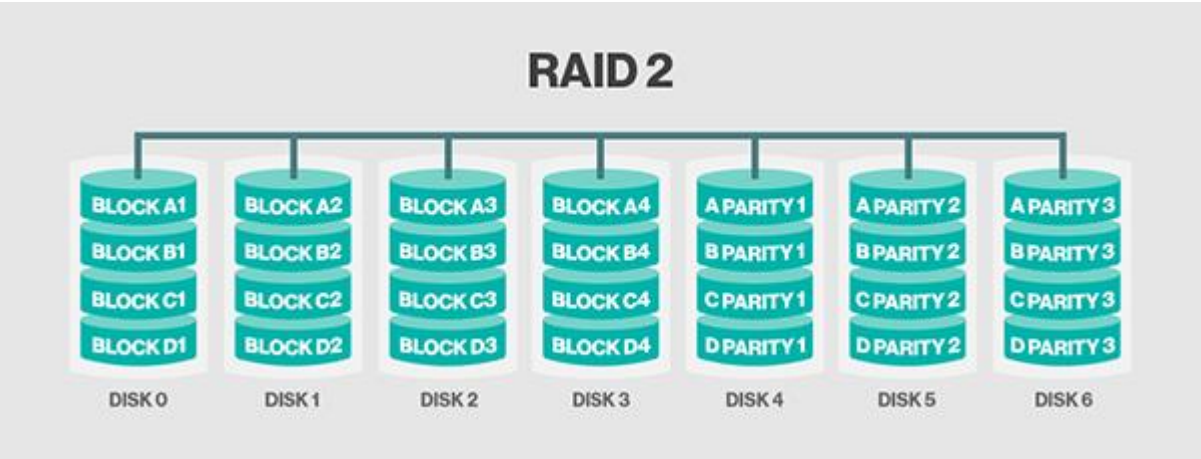


Fig. 4.14 RAID 2

RAID 3: This technique uses striping and dedicates one drive to storing parity information. The embedded ECC information is used to detect errors. Data recovery is accomplished by calculating the exclusive OR (XOR) of the information recorded on the other drives. Since an I/O operation addresses all drives at the same time, RAID 3 cannot overlap I/O. For this reason, RAID 3 is best for single-user systems with long record applications.

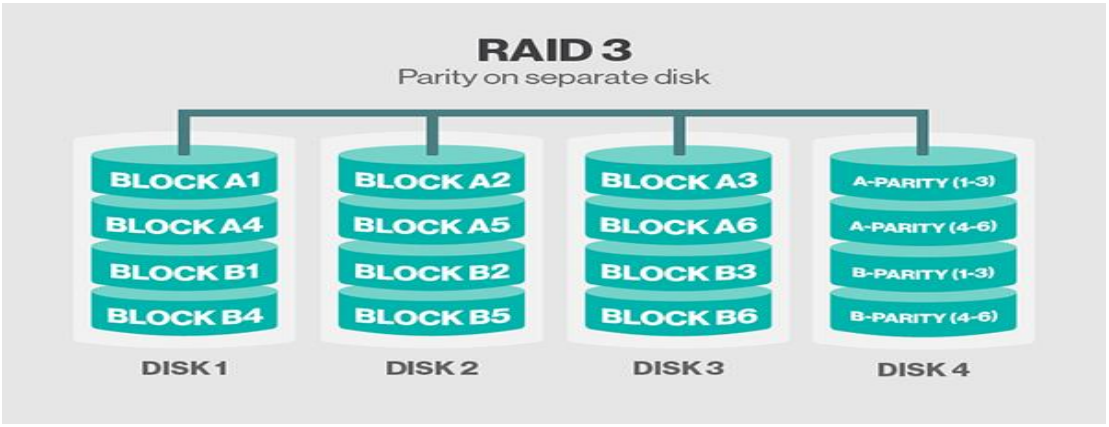


Fig 4.15 RAID 3

RAID 4: This level uses large stripes, which means you can read records from any single drive. This allows you to use overlapped I/O for read operations. Since all write operations have to update the parity drive, no I/O overlapping is possible. RAID 4 offers no advantage over RAID 5.

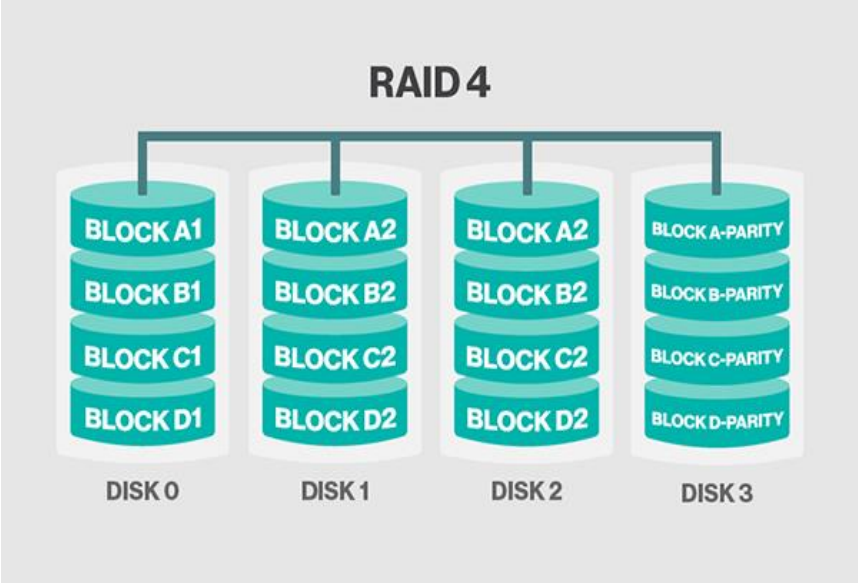


Fig. 4.16 RAID 4

RAID 5: This level is based on block-level striping with parity. This is the most common secure RAID level. It requires at least 3 drives but can work with up to 16. Data blocks are striped across the drives and on one drive a parity checksum of all the block data is written. The parity data are not written to a fixed drive, they are spread across all drives, as the drawing below shows. Using the parity data, the computer can recalculate the data of one of the other data blocks, should those data no longer be available. That means a RAID 5 array can withstand a single drive failure without losing data or access to data. Although RAID 5 can be achieved in software, a hardware controller is recommended. Often extra cache memory is used on these controllers to improve the write performance.

Advantages

- Read data transactions are very fast while write data transactions are somewhat slower (due to the parity that has to be calculated).

- If a drive fails, you still have access to all data, even while the failed drive is being replaced and the storage controller rebuilds the data on the new drive.

Disadvantages

- Drive failures have an effect on throughput, although this is still acceptable.
- This is complex technology. If one of the disks in an array using 4TB disks fails and is replaced, restoring the data (the rebuild time) may take a day or longer, depending on the load on the array and the speed of the controller. If another disk goes bad during that time, data are lost forever.

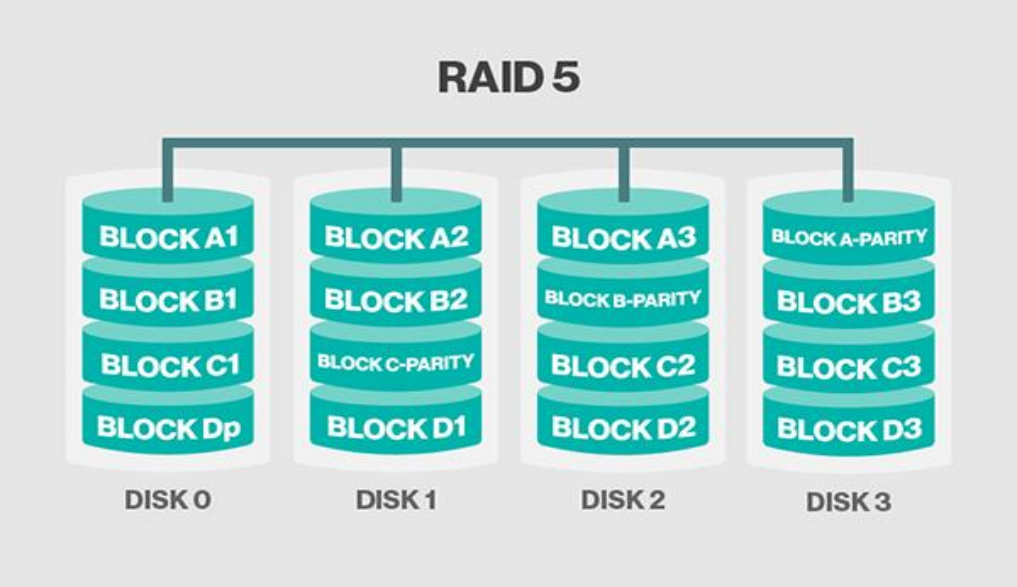


Fig. 4.17 RAID 5

RAID 6: This technique is similar to RAID 5 but includes a second parity scheme that is distributed across the drives in the array. The use of additional parity allows the array to continue to function even if two disks fail simultaneously. However, this extra protection comes at a cost. RAID 6 arrays have a higher cost per gigabyte (GB) and often have slower write performance than RAID 5 arrays.

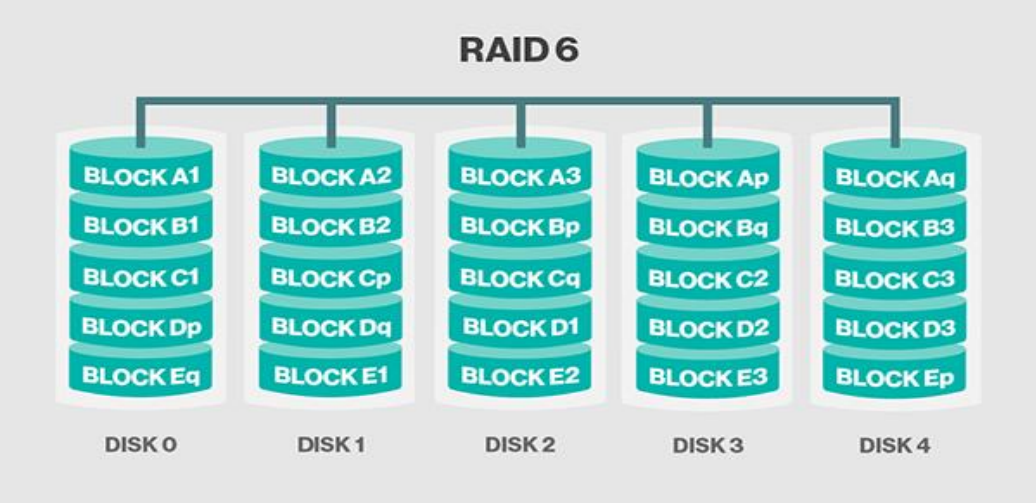


Fig. 4.18 RAID 6

Advantages

- Like with RAID 5, read data transactions are very fast.
- If two drives fail, you still have access to all data, even while the failed drives are being replaced. So RAID 6 is more secure than RAID 5.

Disadvantages

- Write data transactions are slowed down due to the parity that has to be calculated.
- Drive failures have an effect on throughput, although this is still acceptable.
- This is complex technology. Rebuilding an array in which one drive failed can take a long time.

4.2 File Management

Most computer systems employ secondary storage devices such as magnetic disk, magnetic tape, optical media, flash drives etc. to provide cheap, non-volatile storage for programs and data. The programs, and the user data they work with, are stored in discrete storage units called *files*.

Important Tasks of the File management system of the operating system :

i) allocating space for files on secondary storage media as and when required.
ii) keeping track of creating, destroying, organizing, reading, writing, modifying, moving, and controlling access to files; and
iii) management of resources used by files.
iv) keeping *access times* (the time required to write data to or read data from secondary storage) to a minimum.

4.2.1 File concept

4.2.1.1 File definition:

A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks. In general, a file is a sequence of bits, bytes, lines or records which is created by users.

4.2.1.2 File Structure

A File Structure should be according to a required format that the operating system can understand.

- A file has a certain defined structure according to its type.
- A text file is a sequence of characters organized into lines.
- A source file is a sequence of procedures and functions.
- An object file is a sequence of bytes organized into blocks that are understandable by the machine.
- When operating system defines different file structures, it also contains the code to support these file structure. Unix, MS-DOS support minimum number of file structure.

4.2.1.3 File Type

File type refers to the ability of the operating system to distinguish different types of file such as text files, source files and binary files etc. Many operating systems support many types of files. Operating system like MS-DOS and UNIX have the following types of files −

Ordinary files These are the files that contain user information. These may have text, databases or executable program. The user can apply various operations on such files like add, modify, delete or even remove the entire file.

Directory files These files contain list of file names and other information related to these files.

Special files These files are also known as device files. These files represent physical device like disks, terminals, printers, networks, tape drive etc.

These files are of two types −

- **Character special files** − data is handled character by character as in case of terminals or printers.
- **Block special files** − data is handled in blocks as in the case of disks and tapes.

4.2.2 File Attributes

A file has generally the following attributes but vary from operating system to another:

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

Identifier: This unique tag, usually a number, identifies the file within the file system; it is the non-human readable name for the file

Type: This information is needed for those systems that support different types.

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

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

Protection: Access-control information determines who can do reading, writing, executing etc.,.

Time, date, and user identification: This information may be kept for creation, last modification, and last use. These data can be useful for protection, security, and usage monitoring.

The information about all files is kept in the directory structure.

4.2.3 File Operations

A file is an abstract data type. To define a file properly, we need to consider the operations that can be performed on files.

Six basic file operations. The OS can provide system calls to create, write, read, reposition, delete, and truncate files.

- **Creating a file.** Two steps are necessary to create a file.
  i) Space in the file system must be found for the file.
  ii) An entry for the new file must be made in the directory.
- **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. The system must keep a write pointer to the location in the file where the next write is to take place. The write pointer must be updated whenever a write occurs.
- **Reading a file.** To read from a file, we use a system call that specifies the name of the file and where (in memory) the next block of the file should be put. The system needs to keep a read pointer to the location in the file where the next read is to take place.

- Because a process is usually either reading from or writing to a file, the current operation location can be kept as a per-process current-file-position pointer.
- Both the read and write operations use this same pointer, saving space and reducing system complexity.
- **Repositioning within a file.** The directory is searched for the appropriate entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seek.
- **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.
- **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 (except for file length) but lets the file be reset to length zero and its file space released.

4.2.4 Directory Structure

4.2.4.1 Directory

A **directory** is a location for storing files on the computer. Directories are found in a hierarchical file system, such as Linux, MS-DOS, OS/2, and Unix. A directory is an organizational unit, or container, used to organize folders and files into a hierarchical structure. Directories contain bookkeeping information about files. A directory is considered as a file cabinet that contains folders that contain files. Many graphical user interfaces (GUI) use the term *folder* instead of *directory.*

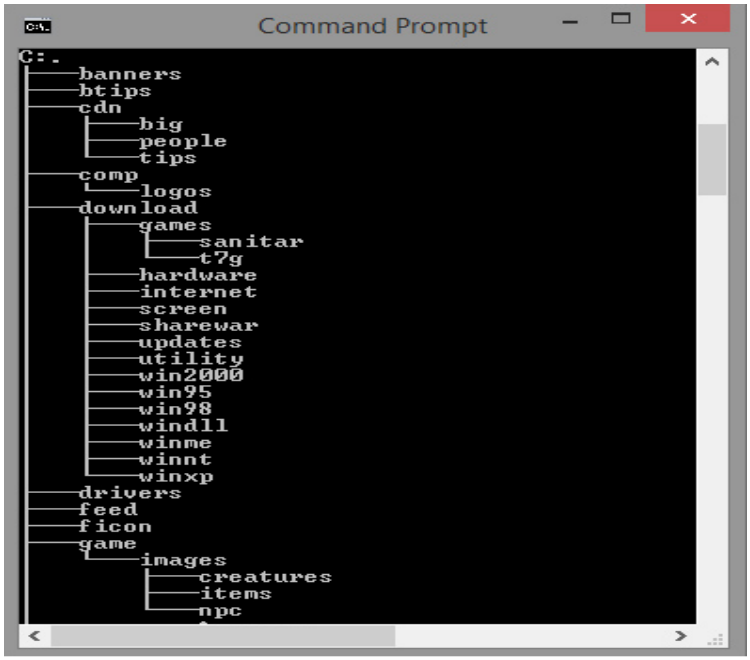
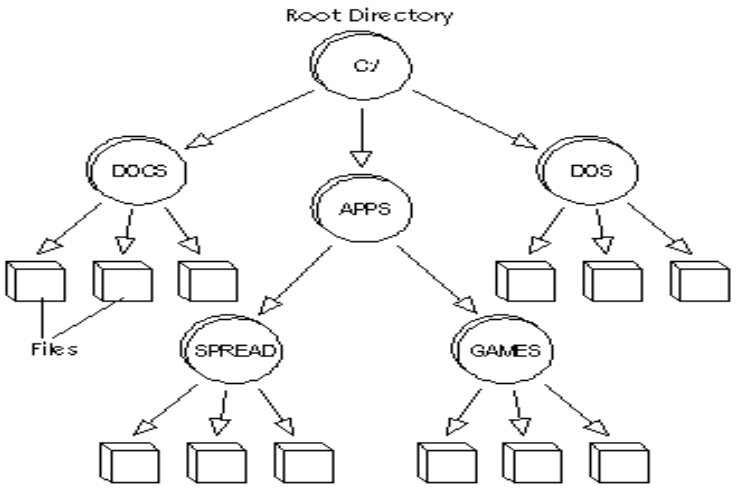


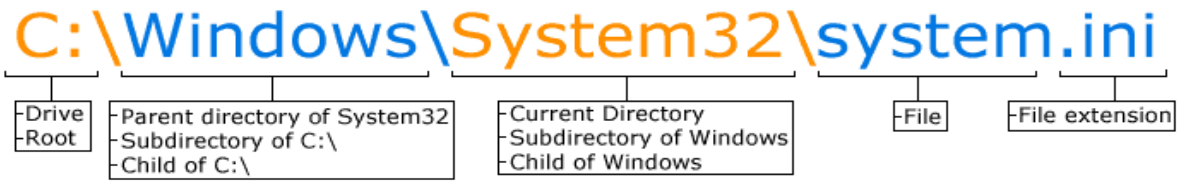
Fig 4.19 example for a Directory

4.2.4.2 Directory related terms and path:



- Root Directory is the top directory in a file system. For example, in DOS systems the root directory is called \ . The directory at the highest level is called the **root directory**
- To indicate a particular file using text, we specify that file's **path**, which is the series of directories through which you must go to find the file
- An **absolute path** name begins at the root and specifies each step down the tree until it reaches the desired file or directory
- A **relative path** name begins from the current working directory

Below is an example of what a directory path would look like in MS-DOS.



- In the above example, C: is the drive letter and the current directory is System32, which is a subdirectory of the Windows directory.

Uses of Directories :

- ❑ A directory entry provides the info needed to find the disk data blocks of a file
  - disk address of first block and size
  - address of first block
  - number of associated i-node
- ❑ File attributes and file names can be stored in the directory entry (Windows) or in its i-node (Unix).
- ❑ File name may have variable length and long file names contain 255 chars.

4.2.4.3 **Types of Directories :** There are many types of directory structure in Operating System. They are as follows :-

1. Single Level Directory 2. Two Level Directory 3. Tree Structured Directory

i) Single Level Directory :

In Single Level Directory , all files are in the same directory. Simple to implement, but each file must have a unique name.

- ***Limitations of Single Level Directory***
  a) Since all files are in the same directory, they must have unique name.
  b) If two user call their data free test, then the unique name rule is violated.
  c) Files are limited in length.
  d) Even a single user may find it difficult to remember the names of all files as the number of file increases.
  e) Keeping track of so many file is a difficult task.

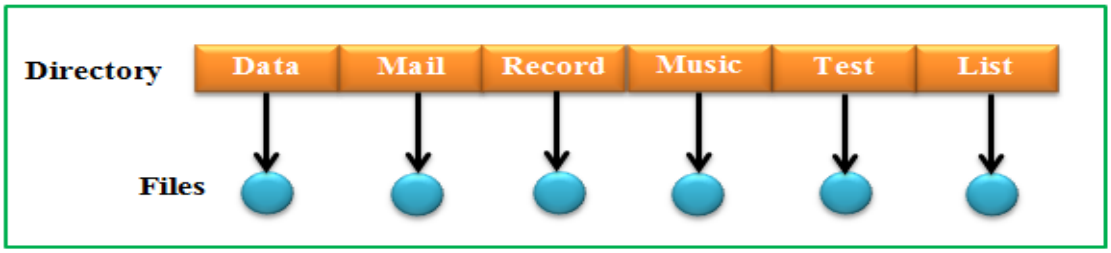


Fig. 4.20 Single Level Directory

ii) T**wo Level Directory :**

1. Each user has its own User File Directory (UFD).
2. File names only need to be unique within a given user's directory.
3. A master file directory is used to keep track of each users directory, and must be maintained when users are added to or removed from the system.
4. A separate directory is generally needed for system (executable) files.
5. When the user job start or user log in, the system Master File Directory (MFD) is searched. MFD is indexed by user name or Account Number.
6. When user refers to a particular file, only his own UFD is searched.
   Thus different users may have files with same name. To have a particular file uniquely, in a two level directory, we must give both the user name and file name.

7. *A two level directory can be a tree or an inverted tree of height 2*

8. The root of a tree is Master File Directory (MFD). Its direct descendents are User File Directory (UFD). The descendents of UFD's are file themselves. The files are the leaves of the tree.

Limitations of Two Level Directory
The structure effectively isolates one user from another.

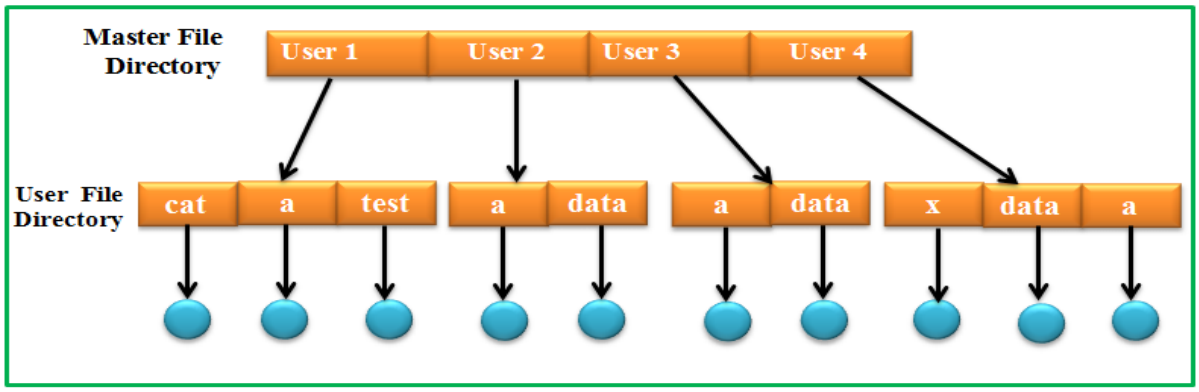


Fig. 4.21 Two Level Directory

iii) Tree Structured Directory :

- A directory (or Sub directory) contains a set of files or sub directories. All directories has the same internal format. Each user / process has the concept of a ***current directory*** from which all (relative) searches take place.
- Files may be accessed using either absolute pathnames (relative to the root of the tree) or relative pathnames (relative to the current directory.)
- Directories are stored the same as any other file in the system, except there is a bit that identifies them as directories, and they have some special structure that the OS understands.

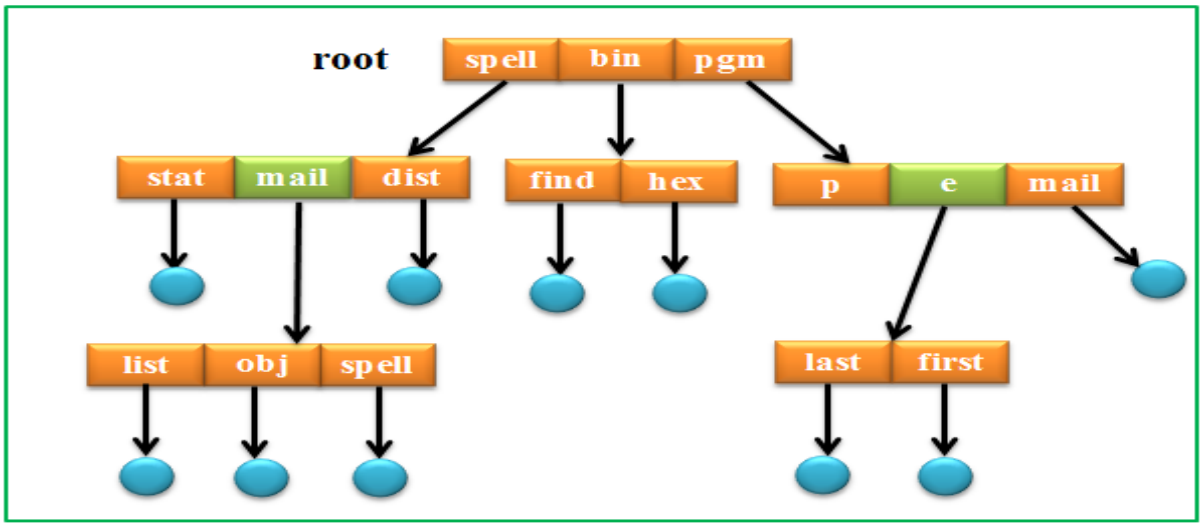


Fig. 4.22 Tree structured Directory

4.2.5 File Allocation Methods

File Allocation :

- File allocation is a technique used to allocate space for files so that disk space is utilized effectively and files can be accessed quickly.
- Disks are divided into physical blocks (sectors on a track).
- Files are divided into logical blocks (subdivisions of the file).
- Logical block size = some multiple of a physical block size.

- The operating system or file management system is responsible for allocating blocks to files.
- Space is allocated to a file as one or more *portions* (contiguous set of allocated ***disk blocks***). A portion is the logical block size.
- ***File allocation table (FAT) -*** data structure used to keep track of the portions assigned to a file.
- A preallocation policy requires that the maximum size of a file be declared at the time of the file creation request.
- Dynamic allocation allocates space to a file in portions as needed.

Three major ***methods*** of ***allocating disk space*** are:

1. ***Contiguous Allocation***

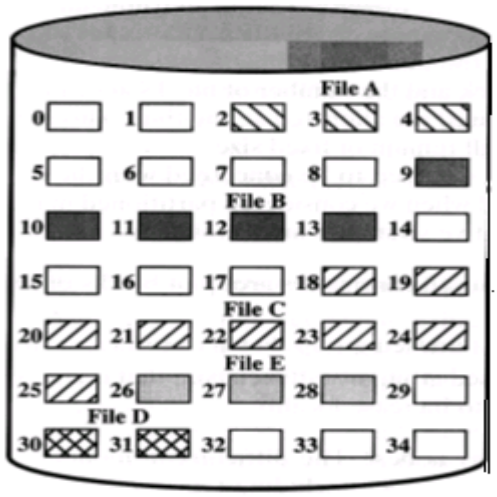
2. ***Non – contiguous : i) Linked or chained Allocation***
   ii) Indexed Allocation.

1. Contiguous File Allocation:

- It store each file as a contiguous run of disk blocks. Thus on a disk with 1-KB blocks, a 50-KB file would be allocated 50 consecutive blocks.
- With this allocation method, a user must indicate the file size before creating the file.
- Then, the operating system searches the disk to find contiguous disk blocks for the file.
- The directory entry is easy. It contains the starting disk address of this file and the number of disk blocks.
- Therefore, if the initial address is b and the number of blocks is n, the file will occupy blocks b, b+1, b+2, …, b+n-1.

File Allocation Table

File Name	Start Block	Length
FileA	2	3
File B	9	5
File C	18	8
File D	30	2
File E	26	3



4.23 Contiguous File Allocation

Advantage:

- Contiguous allocation is ***easy to implement*** because keeping track of file's blocks is done by knowing the disk address of the first block and the number of blocks in the file.
- It has ***high performance*** because the entire file can be read from the disk in a single operation since only one seek is needed.

Disadvantage:

- It can be considered as a form of dynamic memory allocation, and ***external fragmentation*** may ***occur*** and ***compaction*** may be ***needed.***
- ***It is difficult to estimate the file size.*** The size of a file may grow at run time and may be larger than the specified number of allocated blocks. In this case, the OS must move the blocks in order to provide more space.

2. Non- Contiguous Allocation :

i) Linked (Chained) Allocation:

- Typically, allocation is on an individual block basis.
- Each block contains a pointer to the next block in the chain with a linked list.
- Again, the file allocation table needs just a single entry for each file, showing the starting block and the length of the file.
- Although preallocation is possible, it is more common simply to allocate blocks as needed.
- The selection of blocks is made easy . Any free block can be added to a chain.
- To select an individual block of a file requires tracing through the chain to the desired block.

There is no external fragmentation to worry about because only one block at a time is needed. This type of physical organization is

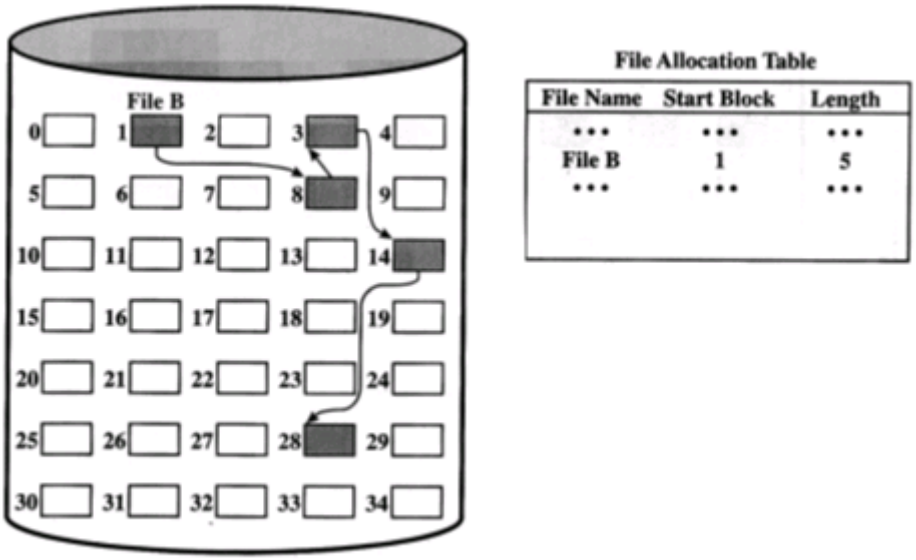


Fig 4.24 Linked File Allocation

Advantages:

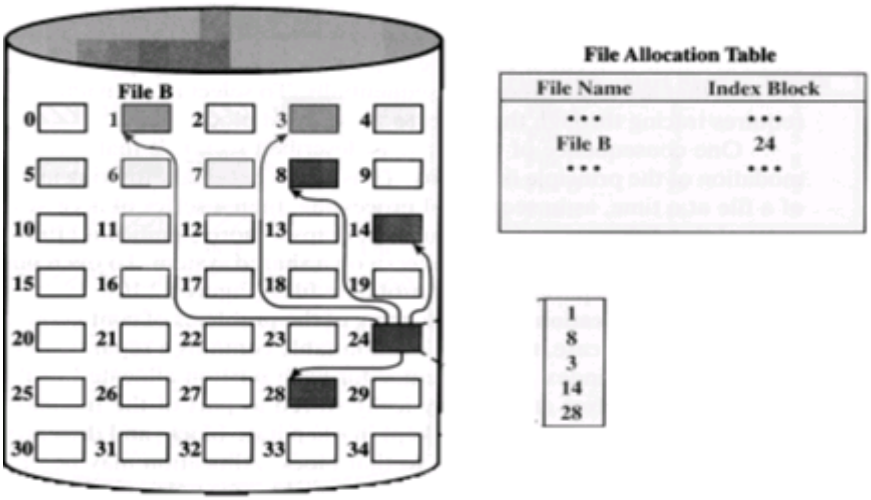
- File size does not have to be specified.
- No external fragmentation.
- best suited to sequential files that are to be processed sequentially.

Disadvantages:

- It is not used for direct access files.
- Each block contains a pointer, wasting space.
- Blocks scatter everywhere and a large number of disks seek may be necessary.

ii) Indexed Allocation:

- Each file has an index block that is an array of disk block addresses.
- The i-th entry in the index block points to the i-th block of the file.
- A file's directory entry contains a pointer to its index.
- Hence, the index block of an indexed allocation plays the same role as the page table. Index allocation supports both sequential and direct access without external fragmentation.
- The indexed allocation suffers from wasted space. The index block may not be fully used (i.e., internal fragmentation).
- The number of entries of an index table determines the size of a file. To overcome this problem, we can have multiple index blocks and chain them into a linked list.
- We can also have multiple index blocks, but make them a tree just like the indexed access method.
- Another alternative is that we can have a combination of both.



4.25 Indexed File Allocation

Advantages:

- File size does not have to be specified.
- No external fragmentation.

4.2.6 File Access Methods

File access mechanism refers to the manner in which the records of a file may be accessed. There are several ways to access files −

- Sequential access
- Direct/Random access
- Indexed sequential access

Sequential access

A sequential access is that in which the records are accessed in some sequence, i.e., the information in the file is processed in order, one record after the other. This access method is the most primitive one. Example: Compilers usually access files in this fashion.

Direct/Random access

- Random access file organization provides, accessing the records directly.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing.
- The records need not be in any sequence within the file and they need not be in adjacent locations on the storage medium.

Indexed sequential access

- This mechanism is built up on base of sequential access.
- An index is created for each file which contains pointers to various blocks.
- Index is searched sequentially and its pointer is used to access the file directly.

Some file systems provide different access methods that specify ways the application will access data

- **Sequential access:**Read bytes one at a time, in order
- **Direct access:**Random access given a block/byte #
- **Record access:**File is array of fixed- or variable-sized records
- **Indexed access:**FS contains an index to a particular field of each record in a file • apps can find a file based on value in that record (similar to DB)

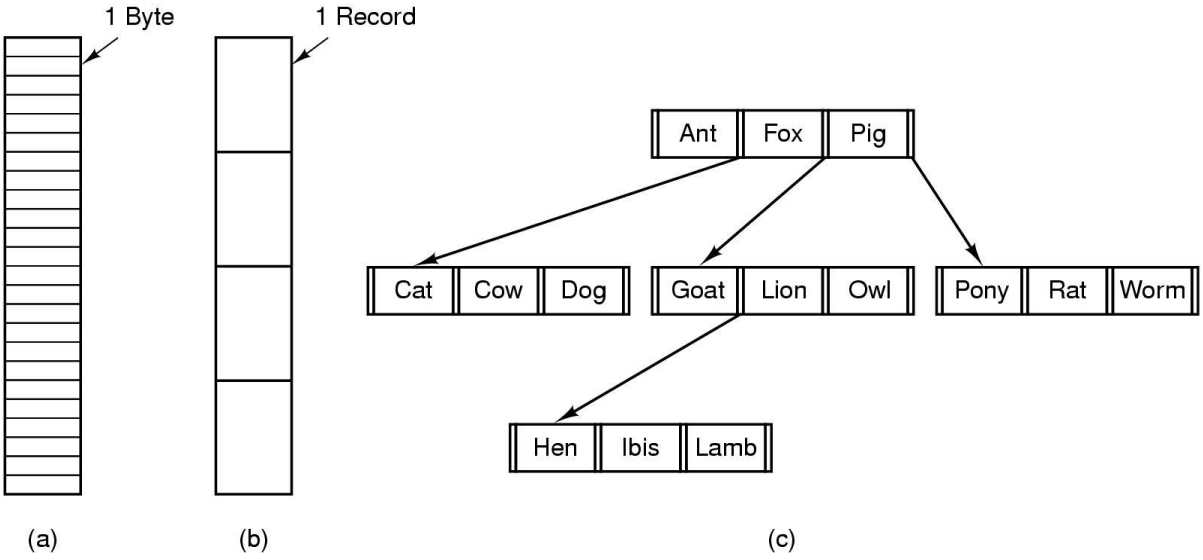
4.2.7 File System Structure

Disks provide the bulk of secondary storage on which a file system is maintained. They have two characteristics that make them a convenient medium for storing multiple files:

- They can be rewritten in place; it is possible to read a block from the disk, to modify the block, and to write it back into the same place.
- They can access directly any given block of information on the disk. Thus, it is simple to access any file either sequentially or randomly, and switching from one file to another requires only moving the read write heads and waiting for the disk to rotate.

Rather than transferring a byte at a time, to improve I/O efficiency, I/O transfers between memory and disk are performed in units of blocks. Each block is one or more sectors. Depending on the disk drive, sectors vary from 32 bytes to 4,096 bytes, they are 512 bytes.

File Structure



Three kinds of files

- byte sequence
- record sequence
- tree

a. Byte Sequence:

The file in Fig. (a) is just an unstructured sequence of bytes. In effect, the operating system does notknow or care what is in the file. All it sees are bytes. Any meaning must be imposed by user-level programs. Both UNIX and Windows 98 use this approach.

b. Record Sequence:

In this model, a file is a sequence of fixed-length records, each with some internal structure. Central to the idea of a file being a sequence of records is the idea that the read operation returns one record and the write operation overwrites or appends one record. As a historical note, when the 80-column punched card was king many (mainframe) operating systems based their file systems on files consisting of 80-character records, in effect, card images

c. Tree:

In this organization, a file consists of a tree of records, not necessarily all the same length, eachcontaining a key field in a fixed position in the record. The tree is sorted on the key field, to allow rapid searching particular key.

4.2.8 Disk formatting

Disk formatting is the process of preparing a data storage device such as a hard disk drive, solid-state drive, floppy disk or USB flash drive for initial use. It is an operation in which a new disk medium is fully prepared to store files. The formatting operation may also create one or more new file systems.

Disk Formatting involves three different processes :

i) Low-level formatting or Physical formatting - performs basic medium preparation.
ii) Partitioning - making the data storage device visible to an operating system.
iii) High-level formatting or Logical formatting - generating a new file system .

i) Low-level formatting : Low-level formatting is the process of marking out cylinders and tracks for a blank hard disk, and then dividing tracks into multiple sectors. This process creates physical format which defines where the data is saved and thus creating the structure of the disk. Low level formatting is performed by disk manufacturers itself. The disk drive's controller marks the surfaces of the disks indicating the start of a recording block and other control information to be used later to read or write data.

ii) Partitioning : This is a process which divides a disk into one or more regions, writing data structures to the disk to indicate the beginning and end of the regions. This level of formatting often includes checking for defective tracks or defective sectors.

iii) High-level formatting : After low-level formatting is complete, we have a disk with tracks and sectors--but nothing written on them. *High-level formatting* is the process of writing the file system structures , cluster size, partition label, etc., on the newly created partition on the disk that make the disk to be used for storing programs and data. This is a fast operation, and is sometimes referred to as *quick formatting*.. And we can also say high-level formatting just clears data on hard disk, generates boot information, initializes FAT, and labels logical bad sectors when the partition has completed .This formatting includes the data structures used by the OS to identify the logical drive or partition's contents. This may occur during operating system installation, or when adding a new disk. This formatting is made by users themselves.

4.3 Security and protection

Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system. If a computer program is run by an unauthorized user, then he/she may cause severe damage to computer or data stored in it. So a computer system must be protected against unauthorized access, malicious access to system memory, viruses, worms etc.

Protection refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer system. This mechanism must provide a means for specifying the controls to be imposed, together with a means of enforcement.

4.3.1 Security Threats

Security Threats can be classified into i) Program Threats ii) System Threats as follows.

4.3.1.1 Program Threats

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common example of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

- **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generatlly a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

4.3.1.2 System Threats

System threats refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats creates such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.

- **Worm** − Worm is a process which can choke down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.
- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.

- **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

4.3.2 Security Policies and Mechanisms

4.3.2.1 Security Policies

To protect a system, we must take security measures at four levels:

1. **Physical -** The site or sites containing the computer systems must be physically secured against harmful entry by intruders. Both the machine rooms and the terminals or workstations that have access to the machines must be secured.

2.**Human -** There is some concern that the humans who are allowed access to a system be trustworthy, and that they cannot be coerced into breaching security. However more and more attacks today are made via ***social engineering,*** which basically means fooling trustworthy people into accidentally breaching security.

- **Phishing** involves sending an innocent-looking e-mail or web site designed to fool people into revealing confidential information. E.g. spam e-mails pretending to be from e-Bay, PayPal, or any of a number of banks or credit-card companies.
- **Dumpster Diving** involves searching the trash or other locations for passwords that are written down. (Note: Passwords that are too hard to remember, or which must be changed frequently are more likely to be written down somewhere close to the user's station.)

- **Password Cracking** involves divining users passwords, either by watching them type in their passwords, knowing something about them like their pet's names, or simply trying all words in common dictionaries.

3.Operating System - The OS must protect itself from security breaches, such as runaway processes (denial of service), memory-access violations, stack overflow violations, the launching of programs with excessive privileges, and many others.

4. Network - Much computer data in modern systems travels over private leased lines, shared lines like the Internet, wireless connections, or dial-up lines. Intercepting these data could be just as harmful as breaking into a computer; and interruption of communications could constitute a remote denial-of-service attack, diminishing users' use of and trust in the system.

4.3.2.2 Authentication

Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways −

- **Username / Password** − User need to enter a registered username and password with Operating system to login into the system.
- **User card/key** − User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.
- **User attribute - fingerprint/ eye retina pattern/ signature** − User need to pass his/her attribute via designated input device used by operating system to login into the system.

User Authentication is carried through following methods:

i) Passwords

This method is the most commonly used for authentication. When the user identifies himself by user ID or account name, he is asked for a password. If the user-supplied password matches the password stored in the system, the system assumes that the account is being accessed by the owner of that account.

Passwords are often used to protect resources in the computer system like files. Whenever a request is made to use the resource, the passwordmust be given. If the password is correct, access is granted. Different passwords may be associated with different access rights. For example, different passwords may be used for reading files, appending files, and updating files.

ii) Encrypted Passwords

Passwords are extremely common because they are easy to understand and use. Unfortunately, passwords can often be guessed, accidentally exposed, sniffed, or illegally transferred from an authorized user to an unauthorized one. So, the operating system uses encryption to avoid the necessity of keeping its password list secret. Each user has a password. The system uses a function to encode all the passwords. Only encoded passwords are stored.

When a user presents a password, it is encoded and compared against the stored encoded password. Even if the stored encoded password is seen, it cannot be decoded, so the password cannot be determined. Thus, the password file does not need to be kept secret.

iii) One Time passwords

One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password are implemented in various ways.

- **Random numbers** − Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
- **Secret key** − User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
- **Network password** − Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

iv) Bio-metrics

The biometric technologies involved are based on the ways in which individuals can be uniquely identified through one or more distinguishing biological traits, such as fingerprints, hand geometry, earlobe geometry, retina and iris patterns, voice waves, keystroke dynamics, DNA and signatures. Biometric authentication is the application of that proof of identity as part of a process validating a user for access to a system. Biometric technologies are used to secure a wide range of electronic communications, including enterprise security, online commerce and banking -- even just logging in to a computer or smart phone.

Biometric authentication systems compare the current biometric data capture to stored, confirmed authentic data in a database. If both samples of the biometric data match, authentication is confirmed and access is granted. The process is sometimes part of a multifactor authentication system. For example, a smart phone user might log on with his personal identification number (PIN) and then provide an iris scan to complete the authentication process.

Summary

- Disks provide the bulk of secondary storage for modern computer systems.
- *storage capacity of a disk drive =*
- No. of working surfaces(or heads) x No. of tracks per surface x No. of sectors per track x No. of bytes per sector.
- Disk speed has the following three parameters :The transfer rate The positioning time, sometimes called the access time, The disk bandwidth
- *Disk scheduling is done by operating systems to schedule I/O requests arriving for disk.i.e. I/O scheduling.*
- Disk Drive Performance parameters:Seek Time,Rotational Latency,.Transfer Time,Disk Access Time, Disk Response Time.
- Disk Scheduling Algorithms are *First Come-First Serve (FCFS) ,Shortest Seek Time First (SSTF) , Elevator (SCAN) , Circular SCAN (C-SCAN) , LOOK and C-LOOK*
- RAID stands for Redundant Array of Inexpensive (or sometimes "Independent") Disks.
  RAID is a method of combining several hard disk drives into one logical unit.
- *Characteristics used in RAID are:*Stripping,Mirroring,Parity
  - RAID 0: In a RAID 0 system data are split up in blocks that get written across all the drives in the array.
  - RAID 1: known as *disk mirroring.* Data are stored twice by writing them to both the data drive (or set of data drives) and a mirror drive (or set of drives)
  - RAID 2: This configuration uses striping across disks with some disks storing error checking and correcting (ECC) information.
  - RAID 3: This technique uses striping and dedicates one drive to storing parity information.
  - RAID 4: This level uses large stripes, which means you can read records from any single drive
  - RAID 5: This level is based on block-level striping with parity
  - RAID 6: This technique is similar to RAID 5 but includes a second parity scheme that is distributed across the drives in the array.
- Files are discrete stored units used to store the programs, and the user data they work with.

- A file is a named collection of related information that is recorded on secondary storage such as magnetic disks, magnetic tapes and optical disks.
- There are different types of files--Ordinary files, Directory files,Special files
- File Attributes are Name, Identifier, Type, Location, Size,Protection, Time, date, and user identification:

- *File Operations areCreating a file,Writing a file. Reading a file. Repositioning within a file. Deleting a file. Truncating a file.*
- A directory is a location for storing files on the computer.
- Types of Directories Single Level Directory , Two Level Directory, Tree Structured Directory
- File allocation is a technique used to allocate space for files so that disk space is utilized effectively and files can be accessed quickly.
- *Three* major *methods* of *allocating disk space* are:*Contiguous Allocation,Non – contiguous i.e. i) Linked or chained Allocation ii) Indexed Allocation.*
- File Access Methods- refers to the manner in which the records of a file may be accessed. There are several ways to access files −Sequential access,Direct/Random access,Indexed sequential access
- Disk formatting is the process of preparing a data storage device such as a hard disk drive, solid-state drive, floppy disk or USB flash drive for initial use.
- Disk Formatting involves three different processes :Low-level formatting or Physical formatting - Partitioning,High-level formatting or Logical formatting.
- Security refers to providing a protection system to computer system resources such as CPU, memory, disk, software programs and most importantly data/information stored in the computer system..
- If a user program made these process do malicious tasks, then it is known as Program Threats- Trojan Horse Trap Door Logic Bomb Virus .
- System threats refers to misuse of system services and network connections to put user in trouble. Following is the list of some well-known system threats.Worm − Port Scanning − Denial of Service
- Security Policies-To protect a system, we must take security measures at four levels: Physical,.Human,Operating System ,Network -
- It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic.i.e.authentication.
- Operating Systems generally identifies/authenticates users using following three ways −Username / Password ,User card/key −User attribute - fingerprint/ eye retina pattern/ signature.
- User Authentication is carried through following methods: Passwords,Encrypted Passwords,One Time passwords, Bio-metrics

Review Questions

Part – A (2 marks)

1. List the Various types of I/O devices.

2. Define: Disk Scheduling.
3. List any four types of Disk Scheduling.
4. What are Sectors?
5. Define: Cylinder.
6. What is a Track?
7. How will you find the capacity of a Disk?
8. What is a RAID array?
9. Define: Data striping.
10. What do you mean by Disk mirroring?
11. What is Parity?
12. Define a file.
13. What are the main tasks of File Management.
14. List the File attributes.
15. What are the various operations that can be done on files?
16. What is a Directory?
17. List the various types of directories.
18. List the advantages of two level directories?
19. Draw a tree based Directory structure.
20. Define : Contiguous allocation of disk space.
21. List the advantages of Indexed Allocation.
22. Give the difference between sequential and random access methods.
23. What is record sequence file structure?
24. What is Low level Disk formatting?
25. Define: Security
26. Name some of the security threats.
27. Define: Authentication.
28. What is the use of Password?
29. Name some areas where onetime passwords are used?
30. Mention some of the biological traits that are used in Biometrics.

Part – B (3 marks)

1. Draw the structure of Disk drive.
2. Why disk scheduling is needed?
3. Write about the three characteristics of RAID.
4. Discuss the various types of Files.
5. Explain any three terms related with Directories.
6. Explain Disk Formatting.
7. What are the Program Threats you know?
8. Differentiate between the use of Passwords , encrypted passwords and onetime passwords.

Part- C (10 marks)

1. Explain the structure of Hard Disk Drive with neat diagram.
2. Discuss about any two Disk scheduling algorithms.
3. Explain the Concept of RAID.
4. Discuss about File attributes.
5. Write about the various File operations.

6. With examples, Explain the Directory structures.
7. Mention the various File Access methods . Explain any two in detail.
8. Discuss about File system structure.
9. Write about various Security Policies.
10. Discuss about various Authentication methods.

UNIT – V
LINUX – A CASE STUDY

LEARNING OBJECTIVES

At the end of this unit, the student will be able to

- Understand the history and the features of the Linux operating system.
- Know about FSF/GNU and the various flavors of Linux OS.
- Differentiate UNIX and Linux operating systems.
- Describe the architecture of Linux operating system.
- Explain the desktop environment of Linux OS.
- Explain EXT2 and VFS file systems.
- List and explain different types of file.
- Explain the concept of file permission and file security.
- Define and explain mounting and unmounting the file system.

UNIT – V
LINUX – A CASE STUDY

LEARNING OBJECTIVES

At the end of this unit, the student will be able to

- Understand the history and the features of the Linux operating system.
- Know about FSF/GNU and the various flavors of Linux OS.
- Differentiate UNIX and Linux operating systems.
- Describe the architecture of Linux operating system.
- Explain the desktop environment of Linux OS.
- Explain EXT2 and VFS file systems.
- List and explain different types of file.
- Explain the concept of file permission and file security.
- Define and explain mounting and unmounting the file system.

5.1 INTRODUCTION

Operating systems are an essential part of any computer system. An operating system is the first piece of software that the computer executes when you turn the machine on. The operating system loads itself into memory and begins managing the resources available on the computer. The most popular operating systems in use today are Windows, Mac OS, UNIX and Linux. Linux is a version of UNIX that has gained popularity in recent years.

5.1.1 HISTORY OF LINUX

Linux is a free and open source operating system. It was developed by **Linus Torvalds**a student of computer science in the University of Helsinki. He released the version of the UNIX operating system called '**Minix**'. It is a multiuser and multitasking operating system. It is released under the GPL (General Public License).

Torvalds posted an early versionof Linux in 1991. Since then, a number of people, collaborating over the Internet,have contributed to the development of Linux, all under the control of Torvalds. Today, Linux is a full-featured UNIX like system that runs on all platforms.

Key to the success of Linux has been the availability of free software packages under theFree Software Foundation (FSF). FSF's goal is stable, platform-independentsoftware that is free, high quality, and embraced by the user community. FSF's GNU projectprovides tools for software developers, and the GNU Public License (GPL) is the FSF seal ofapproval.

.

5.1.2 FEATURES OF LINUX

Following are some of the important features of Linux operating system.

- **Multi-User** − Linux is a multiuser operating system. At same time multiple users can access system resources like memory, ram and application programs.
- **Multitasking:** Linux has the ability to handle more than one job at a time. For example you have executed a command for sorting a huge list and simultaneously typing in a notepad.
- **Portable** − Portability was the one of the main features that made Linux so popular.Linuxand its application can works on different types of hardware. A Linux kernel and application program supports their installation on any kind of hardware platform.
- **Open Source** − Linux source code is freely available. Multiple teams work to enhance the capability of Linux operating system and it is continuously evolving.
- **Hierarchical File System** − Linux provides a standard file structure in which system files and user files are arranged.
- **Shell** − Linux provides a special interpreter program which can be used to execute commands of the operating system.
- **Security:**Security is a very important part of any operating system. Linux provides several security concepts for protecting their users from unauthorized access of their data and system.It provides user security using authentication features like password protection, controlled access to specific files and encryption of data.

- **Communication:** Linux has an excellent feature for communication. It can be within the network of a single main computer, or between two or more such computer networks.

Difference between UNIX and LINUX

S.No	L i n u x	U N I X
1	Linux is free and open source software.	Most UNIX like operating systems are not free. Licensed versions are used for commercial purpose.
2	Linux can be freely distributed, downloaded freely, distributed through magazines, books etc.	Different flavors of Unix have different cost structures.
3	L i n u x i s u s e r f r i e n d l y O S	U n i x i s n o t u s e r f r i e n d l y O S
4	General user can use Linux. For Example From home users to developers can use Linux	Unix operating systems were developed mainly for mainframes, servers and workstations. So normal users cannot easily use Unix.
5	Linux can be installed on a wide variety of computer hardware, ranging from mobile phones, tablet computers and video game consoles, to mainframes and supercomputers.	Unix is restricted. It cannot be installed in any hardware.
6	Linux typically provides two GUIs, KDE and Gnome	Initially Unix was a command based OS, but later a GUI was created called Common Desktop Environment.
7	Linux supports ext2, procfs, sysfs, ramfs and tmpfs file systems.	Unix supportsgpfs, jfs, hfs, zfs, xfs file systems.

5.1.3 LINUX ARCHITECTURE

Linux falls under the category of the layered architecture. It consists of the following layers

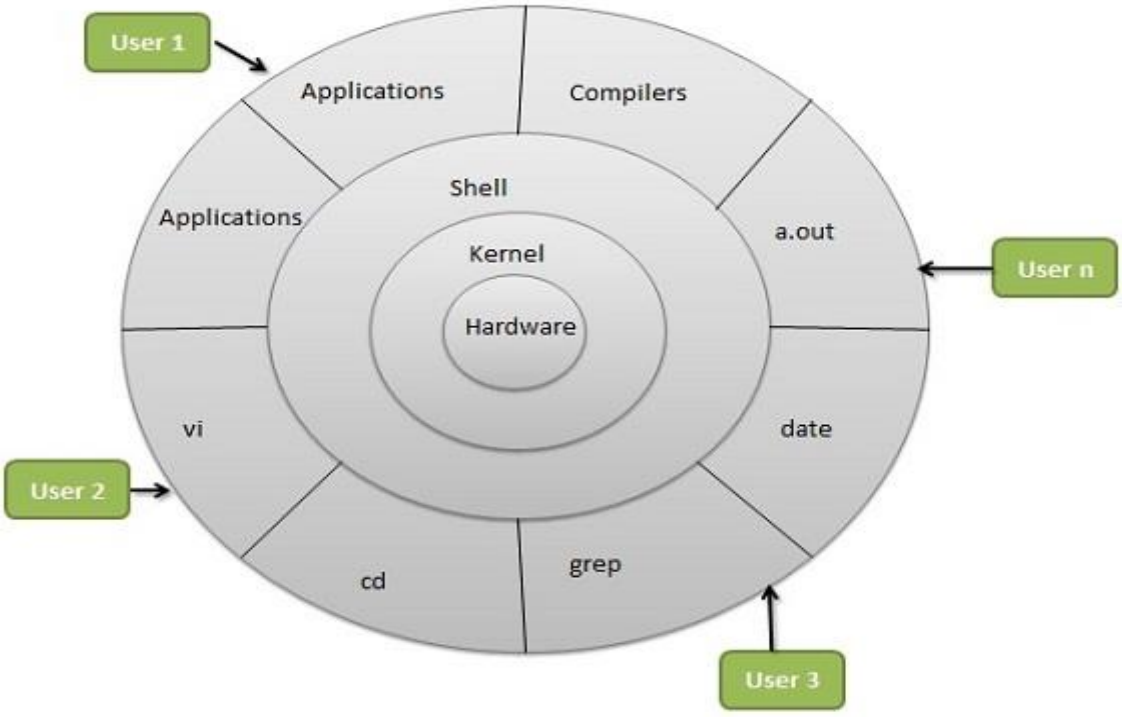


Figure 5.1 Architecture of Linux Operating System

- **Hardware layer –** Hardware consists of all peripheral devices (RAM/ HDD/ CPU etc.).
- **Kernel –** Core component of operating system, for all basic input/output management it interacts directly with the hardware.

- **Shell –**It is an interface between the user and the kernel. It hides the complexity of the kernel's functions from users. It receives commands from user and executes kernel's functions.
- **Application Software –**An utility programs that provide the user most of the functionalities of an operating systems
- **Users –** System users, who interacts directly with the system and application software's.

5.1.4 POPULAR FLAVORS OF LINUX

Linux is free and open source software. It can be modified and distributed freely. Various organizations are involved in modification and redistribution of Linux operating system freely. The different enhancements to the Linux are called flavors.

S.NO	LINUX FLAVORS	DESCRIPTION
1	**Ubuntu** http://www.ubuntulinux.org/	Ubuntu is a Debian-based Linuxoperating system for personal computers, tablets and smart phones. It hides the complexity of the underlying operating system from the user and to provide maximum reliability.
2	**Fedora** http://fedoraproject.org/	Fedora is the free distribution of Red Hat Linux. It includes a powerful desktop GUI based on GNOME and KDE. It is strong on security.
3	Linux Mint http://www.linuxmint.com/	It is a user friendly version of Linux, based on an Ubuntu core and sometimes described as "an improved Ubuntu". It has a full-featured desktop GUI.
4	Debian https://www.debian.com/	Debian is currently known as one of the well-tested and bug-free Linux distribution.
5	Mandriva www.mandriva.com/	Mandriva uses RPM for package management, and its design focuses on ease of installation and use.
6	PCLinuxOS www.pclinuxos.com/	It is a lighter-weight version of Mandriva with good support for graphics drivers, browser plugins and media codec's.
7	**Puppy Linux** http://puppylinux.org/	Small footprint Linux(100 MB once installed), suitable for old hardware or low specification machines. Can run easily from a USB memory stick or Live CD/DVD. It includes a full desktop GUI, browser, general purpose tools and minimal applications.
8	**TinyCore** http://www.tinycorelinux.com/	Very small footprint (10MB once installed) Linux, suitable for old hardware or low specification machines

5.1.5 FSF/GNU

5.1.5.1 FREE SOFTWARE FOUNDATION(FSF)

The **Free Software Foundation** (**FSF**) is founded by Richard Stallman on 4 October 1985 to support the free software movement, which promotes the universal freedom to study, distribute, create, and modify the computer software.

Free software means users of a program have the four essential freedoms:

- The freedom to run the program as your wish, for any purpose (freedom 0).
- The freedom to study how the program works, and adapt it to your needs (freedom 1).
- The freedom to redistribute copies so you can help your neighbor (freedom 2).
- The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3).

5.1.5.2 GNU

GNU is a Unix-like operating system. It is a collection of many programs like applications, libraries, developer tools, even games. The development of GNU, started in January 1984, is known as the GNU Project. The name “GNU” is a recursive acronym for “GNU's Not Unix.” “GNU” is pronounced as *g'noo*.

5.1.6 LINUX DESKTOP

Desktop is thearea of a displayscreenwhereimages,windows,iconsandothergraphicalitemsappear. There are two popular desktop environments supported by Linux operating system are GNOME and KDE.

5.1.6.1 GNOME

GNOMEstands for GNU Network Object Model Environment. It was founded in 1997. GNOME became extremely popular due to its simplicity and ease of use. It is a graphical desktop environment for Linux operating system. It composed entirely of free and open source software.

5.1.6.2 KDE

KDE stands for K Desktop Environment. It is a desktop environment for Linux based operation system. It is an open source graphical desktop environment. KDE provides Linux users a graphical interface to choose their own customized desktop environment.

5.2 FILE SYSTEM

On a Linux system, everything is a file. A Linux system makes no difference between a file and a directory.A directory is just a file containing names of other files. The Linux file system is usually represented in a tree structure.

Linux file system is a collection of files and directories. In a file system, a file is represented by inode. Inode is a kind of serial number containing information about the actual data that makes up the file.

5.2.1 THE SECOND EXTENDED FILE SYSTEM (EXT2)

The Second Extended File system is an extensible and powerful file system for Linux. It is also the most successful file system for Linux community. It is the basis for all of the currently used Linux distributions.

In EXT2 file systemthe data available in files is divided intonumber of data blocks. All the data blocks are of the same length. Every file's size is rounded up to an integral number of blocks. If a file with size 1025 bytes will occupy two 1024 byte blocks. All of the blocks in the file system doesn'thold the data. Some block contains the information about the structure of the file system.

The inode is the basic building block in the EXT2 file system. The inode number is a unique number used to identify the file or directory. All the inodes are kept in a table called inode table. An inode describes

- Which block of the file contains the data
- The access rights of the file
- The file's modification times and
- The type of the file.

Figure 5.2 shows the layout of the EXT2 file system.The inode number is used to read information or data from the file.

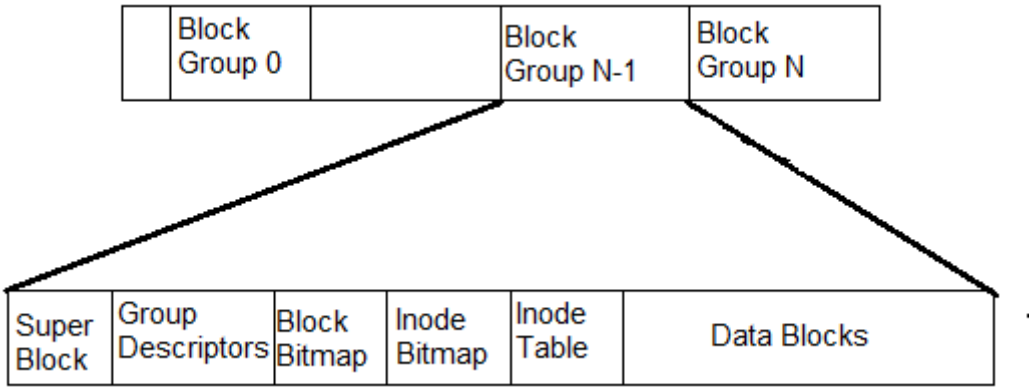


Figure 5.2 Physical Layout of the EXT2 File system

The EXT2 Inode

Figure 5.3 shows the format of an EXT2 inode, amongst other information, it contains the following fields:

Mode

This holds two pieces of information. They are

- what this inode describes
- The permissions that users have to it.

Owner Information

This filed indicates the owner of the file or directory.

Size

This field indicates the size of the file in bytes.

Timestamps

This filed indicates the time that the inode was created and the last modified.

Datablocks

This field indicates the pointers to the blocks that contain the data. The first twelve are pointers to the physical blocks containing the data described by this inode and the last three pointers contain more and more levels of indirection.

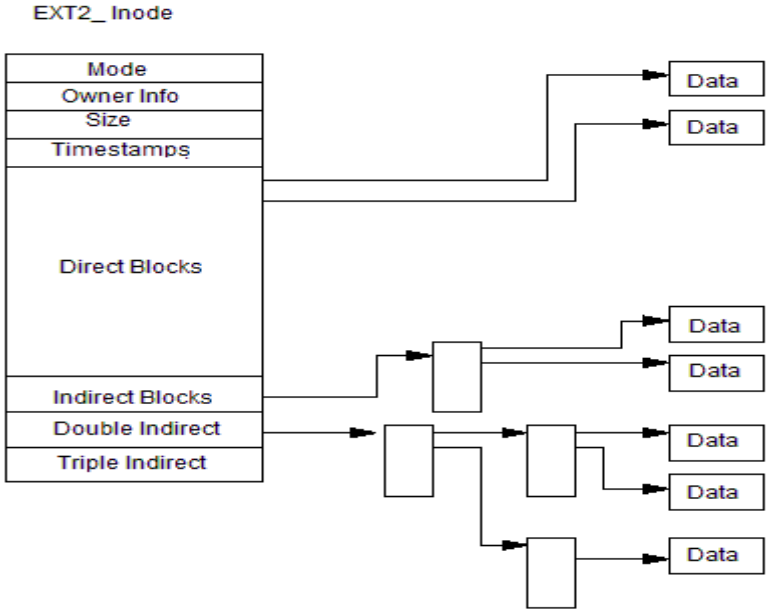


Figure 5.3 Format of an EXT2 inode

5.2.2 VIRTUAL FILE SYSTEM

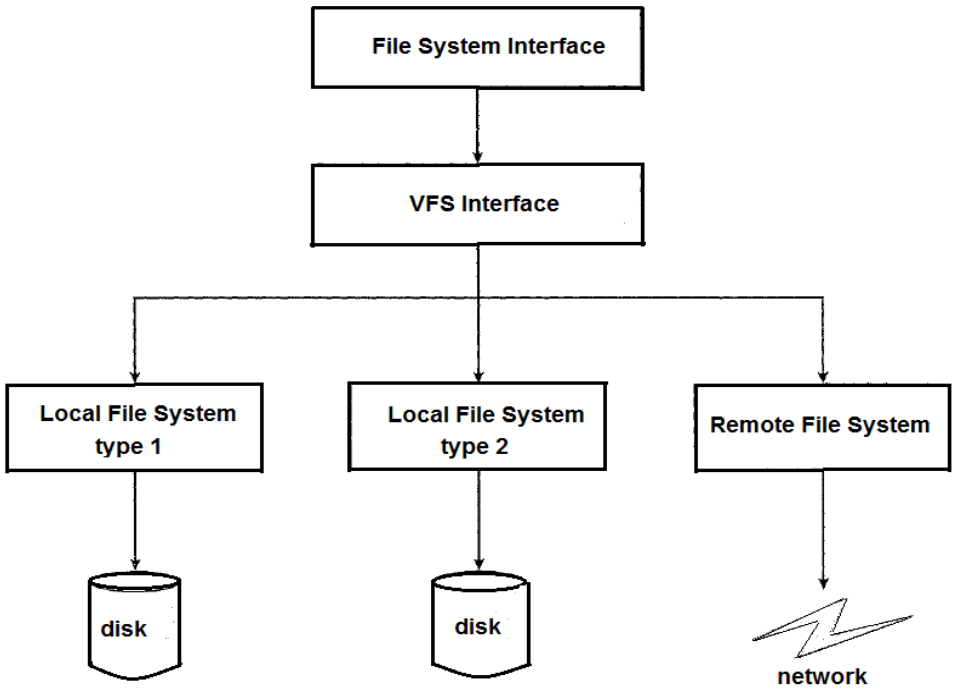


Figure 5.4: Linux Virtual File System

Linux includes a flexible and powerful file-handling facility called **virtual file system (VFS).** It is designed to support a wide variety of file management systems and file structures. VFS presents a single, uniform file system interface to user processes.

Figure 5.4 indicates the key ingredients of the Linux file system strategy. To access any file, a user process issues a requestto VFS file scheme. The VFSconverts the request into an internal file system call which allows the user to access the corresponding file system. So the user can access any file in the directory treewithout the knowledge of the location of the file where it is stored.

The Virtual File System (VFS) is used to manage all the file systems that are mounted in the Linux operating system. To do this, VFS maintains data structures that describe the whole (virtual) file system and the mounted file systems.VFS describes the files in terms of superblocks and inodes.

5.2.2.1 THE VFS SUPERBLOCK

Every mounted file system is represented by a VFS superblock.The VFS superblock contains the following information

The superblock object consists of a number of data items. Examples include the following:

iv) The device that this file system is mounted on
v) The basic block size of the file system
vi) Dirty flag, to indicate that the superblock has been changed but not written back to disk
vii) File system type
viii) Flags, such as a read-only flag
ix) Pointer to the root of the file system directory
x) List of open files

xi) Semaphore for controlling access to the file system
xii) List of superblock operations

5.2.2.2 THE VFS INODE

An inode is associated with each file. The inode holds all the informationabout a file except its name and the actual data contents of the file. Itemscontained in an inode include owner, group, permissions, access times for afile, size of data it holds, and number of links.

The inode also includes inode operations. The methods defined for the inode include the following

Create: Creates a new inode for a regular file.
Lookup: Searches a directory for an inode corresponding to a file name.
mkdir: Creates a new inode for a directory.

5.2.3 DIFFERENT TYPES OF FILES

The different types of files in Linux are

- Regular files
- Directory files
- Special files

    a. Block file(b)
    b. Character device file(c)
    c. Named pipe file or just a pipe file(p)
    d. Symbolic link file(l)
    e. Socket file(s)

○ **Regular files**

Regular file contains the normal data. These files are indicated with "-" at the starting of the line when we use ls –l command. The regular file may be

    a. A Readable file
    b. A Binary file
    c. An Image file
    d. A Compressed file

Example
$ ls - l

-r-xr-xr-x 1 root root 20986522 2015-03-07 ex1.txt
-rwxrwxrwx 1 root root 165 2015-02-18 10:16 abc.sh

The above two files are the regular files created by the user.

v) *Directory file*

Directory contains regular files, folders and special files.This type of files is normally blue in color. These files are indicated with "d" at the starting of the line when we use ls –l command.

Example
$ ls - l

drwxr-xr-x 2 bahratibharati 4096 2015-07-29 12:35 Documents
drwxr-xr-x5bharatibharati4096 2015-04-18 10:46 Desktop

vi) Special file

- *Block file*

These files are hardware files most of them are present in /dev. These files are indicated with "b" at the starting of the line when we use ls –l command.

Example
$ ls - l
brw-rw---- 1 root disk 8, 1 2010-02-15 09:35 sda1
brw-rw---- 1 root disk 8, 2 2010-02-15 09:35 sda2

- *Character device files*

This type of files provides a serial stream of input or output. These files are indicated with "c" at the starting of the line when we use ls –l command.

Example
$ ls–l

crw-rw-rw---1 roor tty 5,0 2010-02-15 16:52 tty
crw--w---- 1 root root 4, 0 2010-02-15 09:35 tty0

- *Pipe files*

The other name of pipe is a “named” pipe, which is sometimes called a FIFO. FIFO refers to the property that the order of bytes going in is the same coming out.These files are indicated with "p" at the starting of the line when we use ls –l command.

Example
$ ls–l

prw-r----- 1 root root 0 2010-02-15 09:35 /dev/.initramfs/usplash_outfifo
prw-r----- 1 root root0 2010-02-15 09:35 /dev/.initramfs/usplash_fifo

- *Symbolic link files*

These are linked files to other files. They are either directory or regular File. The inode number for this file and its parent files are same.These files are indicated with "l" at the starting of the line when we use ls –l command.

Example
$ ls–l

lrwxrwxrwx 1 root root 24 2010-02-15 09:35 sndstat>/proc/asound/oss/sndstat
lrwxrwxrwx1 root root15 2010-02-15 09:35 stderr -> /proc/self/fd/2

- *Socket files*

A socket file is used to pass information between applications for communication purpose. These files are indicated with "s" at the starting of the line when we use ls –l command.

Example
$ ls–l

srwxrwxrwx 1 root root 0 2010-02-15 10:07 /var/run/cups/cups.sock
srwxrwxrwx1 root root 0 2010-02-15 09:35 /var/run/samba/winbindd_privileged/pipe

5.2.4 FILE SECURITY

There are many security features are already built in the Linux operating system. But an important potential vulnerability is granting file permission. Users are not assigned the correct permissions to files and directories.

Basic file permission

Each file and directory has three types of users. They are

d. Owner
e. Group
f. All users

The three levels of file security are

- Read – Read the content of the file or directory
- Write – Write or modify a file or directory
- Execute – Execute the file or directory

To view the permission of the file or directory ls –l command is used. For example

$ ls – l

-r-xr-xr-x 1 root root20986522 2015-03-07 11.15 ex1.txt
-rwxrwxrwx 1 rootroot165 2015-02-18 10:16 abc.sh

In the above output the first ten characters shows the file permission. The first character indicates the special permission flag that can vary based on the type of the file.

The first three ‘rwx’ indicates the permission for the owner. The second three ‘rwx’ indicates the permission given to the group. The last three ‘rwx’ indicates the permission for all users.

Binary references for file permission are

Value	Permission	Description
0	- - -	No permission
1	- - x	Execute permission
2	- w -	Write permission
3	- w x	Write and Execute
4	r - -	Read permission
5	r - x	Read and Execute
6	r w -	Read and Write
7	r w x	Read, Write and Execute

Modifying file permission

The command used to modify the file permission is **chmod.** This command is used to change the file permission for all the three types of user. To add permission ‘+’ is used and to remove permission ‘-‘ is used.

Types of user	symbol
Owner	u
Group	g
All user	o or a

The table above shows the abbreviation of the different types of users used in chmod command. Let us consider the following examples.

Example1
$ls – l

-r-xr-xr-x file1

In the above example all the three types of users have read and execute permission only. To change the file permission, then type

$chmodu +w file1
$chmod g –x file1

After executing the above command the permission for file1 will be changed like the following
$ ls – l
-rwxr--r-x file1

Example2
$chmod 640 file1

The above command gives read and write permission to owner, read permission to group and no permission for all user.

5.2.5 MOUNTING FILE SYSTEM

All the files in Linux operating system are arranged in a tree like structure rooted with /. The mount command attaches a file system located on some drives to the file hierarchy. All the files are to be mounted before the actual use.

The syntax for mounting the file is

$mount device-name destination-directory

Where

Mount – Command used for mounting
Device-name – Name of the new device to be mounted
Destination-directory – The Destination directory to which the new device is going to be Mounted

Example
$mount –t cdrom /dev

5.2.6 UNMOUNTING

All the files are automatically unmounted after its use. Unmounting is nothing but detaching the specified file system from the file hierarchy. A file system cannot be unmounted when it is busy. The general form is

$ unmounts device-name

Example

$ unmounts dev/fd1

Summary

- Linux is a free and open source operating system. It was developed by Linus torvaldsa student of computer science in the University of Helsinki.
- Important features of Linux operating system.-Multi-User, Multitasking ,Portable ,Open Source ,Hierarchical File System , Security and Communication.
- Linux architecture has Hardware layer ,Kernel ,Shell ,Application Software parts.
    - The Free Software Foundation (FSF) to support the free software movement, which promotes the universal freedom to study, distribute, create, and modify the computer software.
- GNU is a Unix-like operating system. It is a collection of many programs like applications, libraries, developer tools, even games.
- The linux desktop environment are -GNOME , KDE
- The linux file system has EXT2, and virtual file system.
- The different types of files in Linux are Regular files , Directory files and Special files
- The different special files are

    - Block file(b)
    - Character device file(c)
    - Named pipe file or just a pipe file(p)
    - Symbolic link file(l)
    - Socket file(s)
- There are many security features are already built in the Linux operating system. But an important potential vulnerability is granting file permission.
- Each file and directory has three types of users. They are Owner,Group,All users.
- The three levels of file security are
    - Read – Read the content of the file or directory
    - Write – Write or modify a file or directory
    - Execute – Execute the file or directory
- The mount command attaches a file system located on some drives to the file hierarchy. All the files are to be mounted before the actual use.
- All the files are automatically unmounted after its use. Unmounting is nothing but detaching the specified file system from the file hierarchy.

REVIEW QUESTIONS

PART - A

1. Who developed Linux?
2. Why Linux is known as free software?
3. Expand FSF/GNU.
4. What is meant by GPL?
5. List any three flavors of Linux.
6. Define directory.
7. Define inode number.
8. List the different types of files used in Linux.
9. Which command is used to modify the file permission?

PART –B

- List the features of Linux.
- Differentiate Linux and Unix.
- Write briefly about GNOME desktop.
- Write briefly about KDE desktop.
- Define mounting and unmounting

PART – C

- With neat sketch explain the architecture of Linux.
- Explain EXT2 file system with neat diagram
- Explain virtual file system.
- Explain the different types of files supported in Linux.
- Explain file security in Linux.

References

- http://tldp.org
- www.computerhope.com
- www.Linux.com