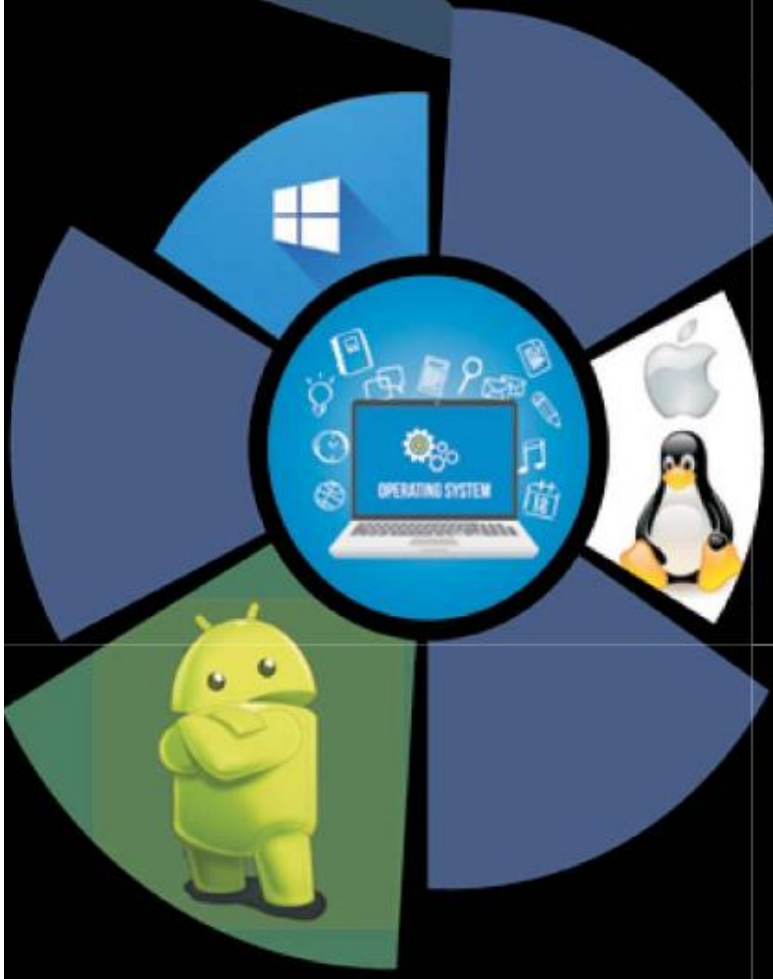
Operating System

EDITION : 2019

Sub Code : 22516



MSBTE I SCHEME PATTERN
T. Y. DIPLOMA SEM V
COMP ENGG./IT PROGRAM GROUP
(CO/CM/IF/CW)

- CHAPTERWISE SOLVED MSBTE QUESTIONS SUMMER 2015 to WINTER 2018
- 2 MARKS QUESTIONS WITH ANSWERS

INCLUDES-I SCHEME PATTERN

SAMPLE PAPERS

As per Revised Syllabus of

MSBTE - I SCHEME

Operating System

T. Y. Diploma (Semester - V)
Computer Engineering / IT Program Group (CO/CM/IF/CW)

Iresh A. Dhotre

M.E. (Information Technology)
Ex-Faculty, Sinhgad College of Engineering
Pune

Poonam A. Vengurlekar

Pursuing M.E. (IT), B.Tech. (IT)
Lecturer, Thakur Polytechnic
Kandivali East, Mumbai

Hitesh K. Mhatre

M.E. (Computer Engineering)
I/C Head of Computer Engineering Department
Pravin Patil College of Diploma Engineering
and Technology
Thane

TECHNICAL PUBLICATIONS
An Up-Thrust for Knowledge

Website : www.technicalpublications.org
https://www.facebook.com/technicalpublications

Operating System

T. Y. Diploma (Semester - V)
Computer Engineering / IT Program Group (CO/CM/IF/CW)

First Edition : June 2019

Published by :

TECHNICAL PUBLICATIONS

Amit Residency, Office No.1, 412, Shaniwar Peth, Pune - 411030, M.S. INDIA
Ph.: +91-020-24495496/97, Telefax : +91-020-24495497
Email : sales@technicalpublications.org Website : www.technicalpublications.org

Printer :

Yogiraj Printers & Binders
Sr.No. 10\1A,
Ghule Industrial Estate, Nanded Village Road,
Tal-Haveli, Dist-Pune - 411041.

Price : ₹ 120 /-

ISBN 978-93-89180-02-2

MSBTE I

PREFACE

The importance of **Operating System** is well known in various engineering fields. Overwhelming respondse to our books on various subjects inspired us to write this book. The book is structured to cover the key aspects of the subject **Operating System.**

The book uses plain, lucid language to explain fundamentals of this subject. The book provides logical method of explaining various complicated concepts and stepwise methods to explain the important topics. Each chapter is well supported with necessary illustrations, practical examples and solved problems. All chapters in this book are arranged in a proper sequence that permits each topic to build upon earlier studies. All care has been taken to make students comfortable in understanding the basic concepts of this subject.

Board questions have been added at the end of each section to help the students in picking important points from that section.

The book not only covers the entire scope of the subject but explains the philosophy of the subject. This makes the understanding of this subject more clear and makes it more interesting. The book will be very useful not only to the students but also to the subject teachers. The students have to omit nothing and possibly have to cover nothing more.

We wish to express our profound thanks to all those who helped in making this book a reality. Much needed moral support and encouragement is provided on numerous occasions by our whole family. We wish to thank the **Publisher** and the entire team of **Technical Publications** who have taken immense pain to get this book in time with quality printing.

Any suggestion for the improvement of the book will be acknowledged and well appreciated.

Authors

D. A. Dhotre

Poonam A. Vengurlekar

Hitesh K. Mhatre

Dedicated to God.

SYLLABUS

Operating System (22516)

Teaching Scheme			Credit (L + T + P)	Examination Scheme											
				Theory						Practical					
L	T	P		Paper Hrs.	ESE		PA		Total		ESE		PA		Total	
					Max	Min	Max	Min	Max	Min	Max	Min	Max	Min	Max	Min
3	-	2	5	3	70	28	30*	00	100	40	25@	10	25	10	50	20

Unit	Unit Outcomes (UOs) (in cognitive domain)	Topics and Sub - topics
Unit - I **Overview of Operating System**	1a. Explain the functioning of given component of OS. 1b. Explain characteristics of the given type of operating system. 1c. Identify type of operating system suitable for the given type of application. 1d. Execute command on command line for the given task.	1.1 Operating System - Concept, Components of operating system, operations of OS: Program Management, Resource management, Security and protection. Views of OS: User view, System View 1.2 Different Types of Operating systems- Batch operating system, Multi Programmed, Time Shared OS, Multiprocessor Systems, Distributed Systems, Real time systems. Mobile OS (Android, iOS). 1.3 Command line based OS - DOS, UNIX GUI based OS - WINDOWS, LINUX.
Unit - II **Services and Components of Operating System**	2a. Start, stop, and restart the given service in Linux. 2b. Explain use of the given System call of specified OS. 2c. Explain process the OS follows in managing the given resource. 2d. Explain use of the given operating system tool.	2.1 Different Services of Operating System. 2.2 System Calls- Concept, types of system calls 2.3 OS Components:- Process Management, Main Memory Management, File Management, I/O System management, Secondary storage management. 2.4 Use of operating system tool user management, security policy device management, performance monitor, task scheduler

Unit - III **Process Management**	3a. Explain functions carried out in the given process state. 3b. Describe the function of the given component of process stack in PCB. 3c. Explain characteristics of the given multithreading model. 3d. Describe method of executing the given process command with example.	3.1 Process-: process states, Process Control Block (PCB). 3.2 Process Scheduling- Scheduling Queues, Schedulers, Context switch. 3.3 Inter-process communication (IPC): Introduction , shared memory system and message passing system. 3.4 Threads - Benefits, users and kernel threads, Multithreading Models- Many to One, One to One, Many to Many. 3.5 Execute process commands- like ps, wait, sleep, exit, kill
Unit - IV **CPU Scheduling and Algorithms**	4a. Justify the need and objective of given job scheduling criteria with relevant example. 4b. Explain with example the procedure of allocating CPU to the given process using the specified OS. 4c. Calculate turnaround time and average waiting time of the given scheduling algorithm. 4d. Explain functioning of the given necessary condition leading to deadlock.	4.1 Scheduling types - scheduling Objectives, CPU and I/O burst cycles, Pre-emptive, Non-Pre-emptive Scheduling, Scheduling criteria. 4.2 Types of Scheduling algorithms- First come first served (FCFS), Shotlest Job First (SJF), Shortest Remaining Time(SRTN), Round Robin (RR) Priority scheduling, multilevel queue scheduling. 4.3 Deadlock- System Models, Necessary Conditions leading to Deadlocks, Deadlock Handling- Preventions, avoidance.
Unit - V **Memory Management**	Sa. Describe the working of specified memory management function. 5b. Explain characteristic of the given memory management techniques. 5c. Write algorithm for the given page replacement technique. 5d. Calculate Page fault for the given page reference string.	5.1 Basic Memory Management- Partitioning, Fixed and Variable, Free Space management Techniques - Bitmap, Linked List. 5.2 Virtual Memory- Introduction to Paging, Segmentation, Fragmentation and Page fault. 5.3 Page Replacement Algorithms: FIFO, LRU , Optimal.

Unit - VI **File Management**	6a. Explain structure of the given file system with example. 6b. Describe mechanism of the given file access method. 6c. Explain procedure to create and access directories and assign the given files access permissions. 6d. Explain features of the given Raid level structure of hard disk.	6.1 File - Concepts, Attributes, Operations, types and File System Structure. 6.2 Access Methods - Sequential, Direct, Swapping, File Allocation Methods - Contiguous, Linked, Indexed. 6.3 Directory structure-Single level , two levels, tree-structured directory, Disk Organization and disk Structure - Physical structure, Logical structure, Raid structure of disk, raid level 0 to 6.

TABLE OF CONTENTS

Unit - I

Unit - II

Unit - III

Unit - IV

Unit - V

Unit - VI

UNIT - I

1 OVERVIEW OF OPERATING SYSTEM

1.1 Operating System Concept

Define operating system

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

What are the functions of OS ?

- **Operating system performs three functions :**
  1. **Convenience** : An OS makes a computer more convenient to use.
  2. **Efficiency** : An OS allows the computer system resources to be used in an efficient manner.
  3. **Ability to evolve** : An OS should be constructed in such a way as to permit the effective development, testing and introduction of new system functions without at the same time interfering with service.

Board Question

1. *List any four functions of operating system.*

MSBTE : Summer-16, Marks 4

1.2 Components of Computer System

- Computer system consists of hardware device and software that are combined to provide a tool to user for solving problems.
- Fig. 1.2.1 shows modern computer system.
- Modern computer consists of one or two CPU with main memory and various I/O devices. Common bus is used for communication between these devices. Each device has its own device controller.
- CPU and device controller uses memory cycle for execution purposes. But memory cycle is only available to one device at a time.
- Bootstrap program is loaded when user start the computer. It initialies all the device connected to the computer system and then loads required device drivers.
- After this, operating system loads in the computer system. In UNIX OS, an 'init' is the first process which execute by OS.
- Interrupt is software and hardware. It is used to send signal to CPU. Software interrupt is sometime called system call.
- When interrupt is trigger, the CPU stops executing the instruction and control is transfer to the fixed location. Starting address is stored at fixed location where the service routine executes.
- Interrupts do not alter the control flow of the process executing on the processor.

Storage structure

- Processor access the data from main memory before executing any instruction. Main memory is also called Random Access Memory (RAM).
- DRAM is used in main memory. Fig. 1.2.2 shows hierarchy of storage device.
- At the top of the hierarchy, we have storage on the CPU registers. For accessing the CPU, it is fastest form of storage.
- Cache memory capacity is less than 1 MB.

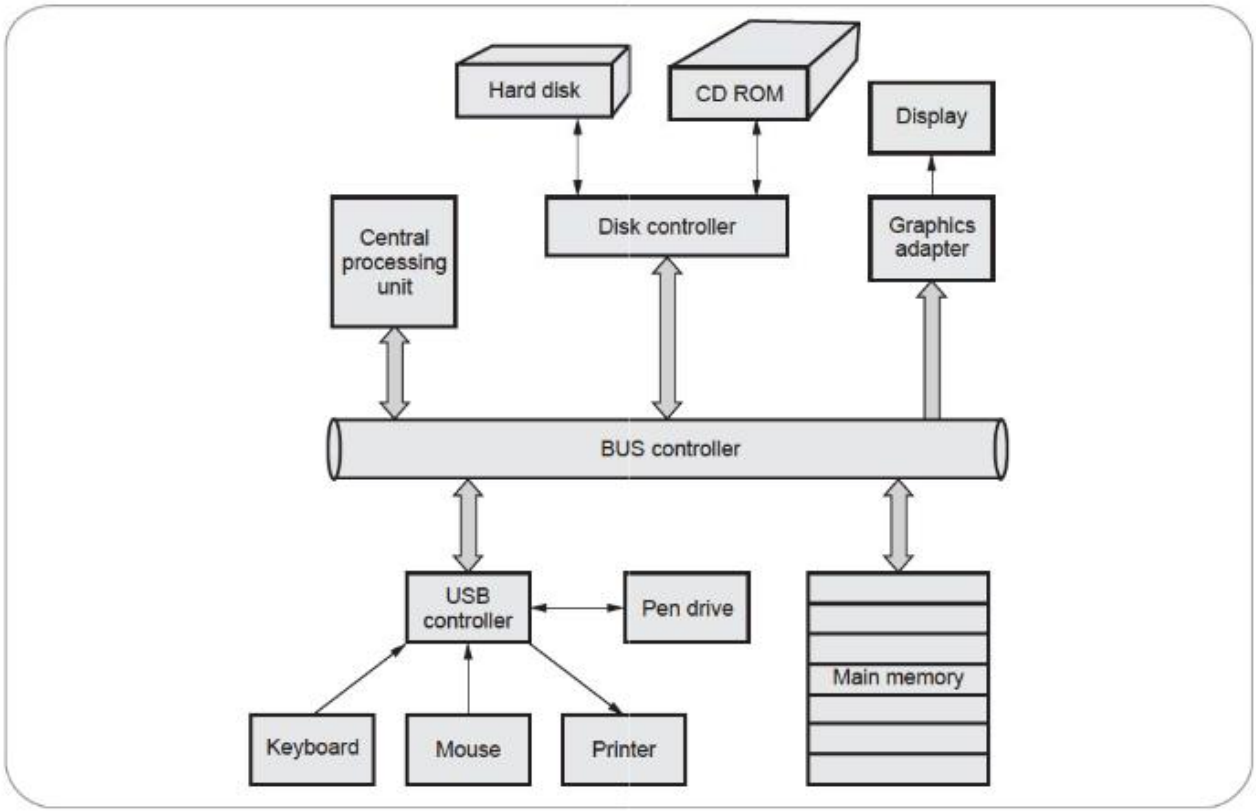


Fig. 1.2.1 Modern computer system

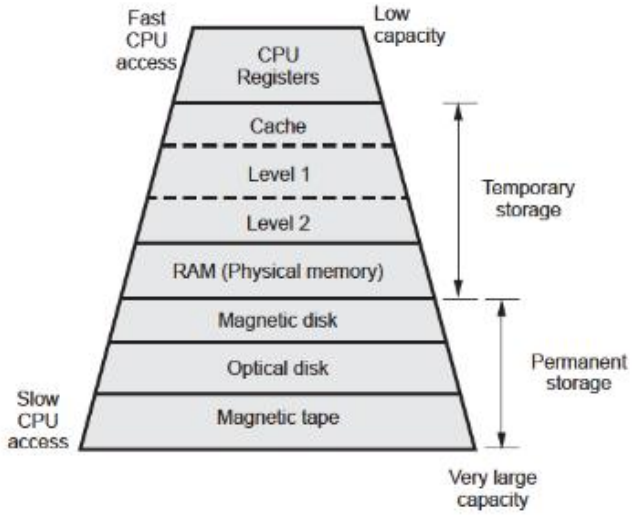


Fig. 1.2.2 Hierarchy of storage device

- User program and data are stored in the main memory. Main memory is volatile, so it can not stored permanently.
- Storage system is classified as temporary storage or permanent storage.
- Top level storage devices are low capacity with faster CPU access and bottom level storage devices having very large capacity with slow CPU access speed.

I/O structure

- Every device uses a device controller to connect it to the computer's address and data bus. Devices can be classified as a block oriented or character oriented, depending on the number of bytes transferred on an individual operation.
- Storage devices are used to store data while the computer is off.
- All the I/O devices are connected to each other by using common bus. CPU and main memory is also connected with this bus.
- Various types of controller is used in the computer system. Small Computer System Interface (SCSI) controller can handle upto seven devices. Each device controller have its own buffer.

- Device controller manage the data transfer between peripheral device and its controller. Device driver is handled by device controller.

I/O operation steps

1. Device driver loads the registers within the device controller.
2. Device controller takes action according to the data loaded into the register.
3. Data is transfer from device to its local buffer with the help of device controller.
4. After completion of data transfer, the device controller sends an interrupt signal to device driver about data transfer completion operation.
5. Then control goes to operating system.

- The device driver is the operating system entity that controls CPU-I/O parallelism. The software that communicates with device controller is called device driver.
- A device can be started by the device driver, and the application program can continue operation in parallel with the device operation.

Board Question

1. List and draw a neat labelled diagram of four components of a computer system.

MSBTE : Summer-18, Marks 4

1.3 Operation of OS

- For single-user programmer operating systems, programmer has the complete control over the system. They operate the system from the console. When new operating systems developed with some additional features, the system control transfers from programmer to the operating system.
- Old operating systems were called resident monitors. The operating system began to perform many of the functions, like input-output operation with the resident monitor.
- Before the operating system, programmer is responsible for the controls of input-output device operations. As the requirements of programmers from computer systems go on increasing and development in the field of communication helps to the operating system to fulfill the needs of programmers.
- Sharing of resource among different programmers is possible without increasing cost. It improves the system utilization but problems increase. If single system was used without share, an error occurs, that could cause problems for only the one program which was running on that machine.
- In sharing, other programs also affected by single program. For example, batch operating system faces the problem of infinite loop. This loop could prevent the correct operation of many jobs. In multiprogramming system, one erroreous program affects the other program or data of that program.
- For proper operation and error free result, protection of error is required. Without protection, only single process will execute one at a time otherwise the output of each program is separated. While designing the operating system, this type of care must be taken into consideration.
- Computer hardware detects the errors. Operating system handled this type of errors. Execution of illegal instruction or access of memory that is not in the user's address space, this type of operation found by the hardware and will trap to the operating system.
- Trap uses an interrupt vector and it transfer the control to the operating system. Whenever error occurs, OS terminates the program abnormally. Hardware protection is used to handle this type of situation.

1.3.1 Dual Mode Operation

- For proper operation and correct output, operating system must be protected. The users program and data must be protected from any malfunctioning program. Shared resource also needs some kind of protection.
- Dual mode uses user mode and monitor mode for working of OS. The monitor mode also called system mode, supervisor mode or privileged mode.
- For indicating current mode of the system, mode bit is used in the computer hardware. The mode bit is 0 for monitor and 1 for user. With the mode bit, user are able to distinguish between a task that is executed in user mode or monitor mode.

- At the booting time, the hardware starts in the monitor mode, then operating system is loaded. The hardware switches from user mode to monitor mode when interrupts occur. In monitor mode, OS gains control of the system.
- The dual mode operation provides the protection to the operating system from unauthorised users. The privileged instructions are executed only in the monitor mode. The computer hardware is not allowed for executing the privillege instructions in other mode, i.e. user mode. If anybody tries to execute the instructions in user mode, it is considered as illegal instruction and also traps it to the operating system.
- Software may trigger an interrupt by executing a special operation called a system call. System call is one type of request which is invoked by the user or system. Using privileged instructions, user will interact with the operating system. This type of request is invoked by user to execute the privileged instructions. As said earlier, this request is called system call or monitor call. When a system call is executed, it is treated by the hardware as a software interrupt.

1.3.2 Resource Management

- A computer is a set of resources. These resource provides various functions to the user. Functions like data movement, storing of data and program, operation on data are control by an operating system.
- Fig. 1.3.1 shows OS as a resource manager.

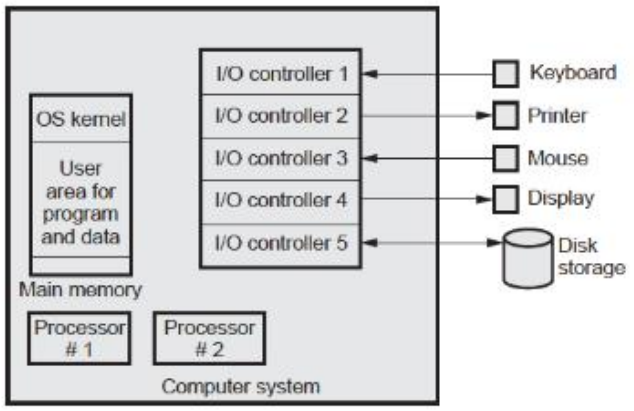


Fig. 1.3.1 OS as a resource manager

- The operating system is responsible for managing the all resources. A portion of the OS is in main memory. This portion of the OS is called kernel.
- User program and data is also stored in remaining parts of the memory. Allocation of main memory is controlled by operating system with the help of memory management hardware.
- I/O device is controlled by OS and it decides when an I/O device can be used by program in execution. Processor is one type of resource and OS control the execution of user program on the processor.
- Modern OS allows multiple programs to run at the same time. If multiple users are using computer then there is need of managing and protecting the memory, I/O devices and other resources.
- Resource management includes sharing resources in different ways. Time and space are the two concept for resource sharing.

1. **Time** : Time slot is allocated to each program first one gets to use the resource then another and so on.
2. **Space** : Consider the example of main memory. Main memory is normally divided up among several running programs, so each one can be resident at the same time.

1.3.3 Protection and Security

- **Protection** : Any mechanism for controlling access of processes or users to resources defined by the OS.
- **Security** : Defense of the system against internal and external attacks.
- Protection can improve reliability by detecting errors at the interfaces between component subsystems. A system can have adequate protection but still be prone to failure and allow inappropriate access.
- For protection and security, systems generally first distinguish among users, to determine who can do what. Most of the operating system maintain the list of user names and associated user identifiers. User identities (user IDs, security IDs) include name and associated number, one per user.
- User ID then associated with all files processes of that user to determine access control. Group

identifier (group ID) allows set of users to be defined and controls managed, then also associated with each process, file.

- Privilege escalation allows user to change to effective ID with more rights. The user may need access to a device that is restricted. Various operating systems provide different methods to allow privilege escalation.
- **UNIX operating system :** The setuid attribute on a program causes that program to run with the user ID of the owner of the file, rather than the current user's ID.

1.4 Views of OS

1.4.1 User View

- The user's view of the computer changes according to the interface being used. Normal PC consists of monitor, keyboard, CPU and mouse. This system is for single user. Now a days, new generation started using Laptop but it consists same things.
- The goal is to maximize the user work. For that purpose, system is designed for easy use and better performance. Resource utilization is also considered while designing the system.
- In some cases, user uses terminal which is connected to a mainframe or a minicomputer. Other users are accessing the same computer through other terminals. These users share resources and may exchange information.
- The OS is designed to get maximum resource utilization to assure that all available CPU time, memory, and I/O are used efficiently.
- If the user is sitting on a workstation connected to other workstations through networks, then the operating system needs to focus on both individual usage of resources and sharing though the network. This happens because the workstation exclusively uses its own resources but it also needs to share files etc. with other workstations across the network.
- If the user is using a handheld computer (mobile device), then the operating system handles the usability of the device including a few remote operations. The battery level of the device is also taken into account.

1.4.2 System View

- The system views the operating system as a resource allocator. There are many resources such as CPU time, memory space, file storage space, I/O devices etc. that are required by processes for execution.
- It is the responsibility of the operating system to allocate these resources to the processes so that the computer system can run as smoothly as possible.
- An operating system is a control program. A control program manages the execution of user programs to prevent errors and improper use of the computer. It is especially concerned with the operation and control of I/O devices.
- Operating systems can also be viewed as a way to make using hardware easier.
- An operating system can also be considered as a program running at all times in the background of a computer system and handling all the application programs.

1.5 Different Types of Operating System

1.5.1 Batch Operating System

- Batch system process a collection of jobs, called a batch. Batch is a sequence of user jobs.
- Job is a predefined sequence of commands, programs and data that are combined into a single unit.
- Each job in the batch is independent of other jobs in the batch. A user can define a job control specification by constructing a file with a sequence of commands.
- Jobs with similar needs were batched together to speed up processing. Card readers and tape drives are the input device in batch systems. Output devices are tape drives, card punches and line printers.
- Primary function of the batch system is to service the jobs in a batch one after another without requiring the operator's intervention. There is no need for human/user interaction with the job when it runs, since all the information required to complete job is kept in files.

- Some computer systems only did one thing at a time. They had a list of instructions to carry out and these would be carried out one after the other. This is called a **serial system.** The mechanics of development and preparation of programs in such environments are quite slow and numerous manual operations involved in the process.
- **Batch monitor** is used to implement batch processing system. Batch monitor is also called **kernel**. Kernel resides in one part of the computer main memory.
- The memory allocator managed the main memory space. Batch monitor controls the sequence of events. Main memory store the batch monitor and users program and data (jobs). Fig. 1.5.1 shows memory layout for a batch system.

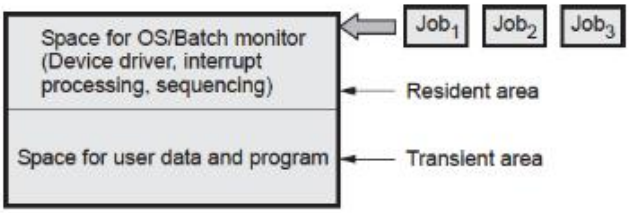


Fig. 1.5.1 Memory layout for batch system

- Computer operator gives a command to start the processing of a batch, the kernel sets up the processing of the first job. Job was selected from the job queue and loaded into main memory. When a job completed execution, its memory was released and the output for the job was copied.
- When a job is completed, it returns control to the monitor, which immediately reads in the next job.
- Fig. 1.5.2 shows concept of batch system.
- Scheduling is also simple in batch system. Jobs are processed in the order of submission i.e. first come first served fashion.
- When a job completes execution, its memory is released and the output for the job gets copied into an output **spool** for later printing.

1.5.1.1 Spooling

- Spooling an acronym for **simultaneous peripheral operation on line.** Spooling uses the disk as a large buffer for outputting data to printers and other devices. It can also be used for input, but is generally used for output.
- Its main use is to prevent two users from alternating printing lines to the line printer on the

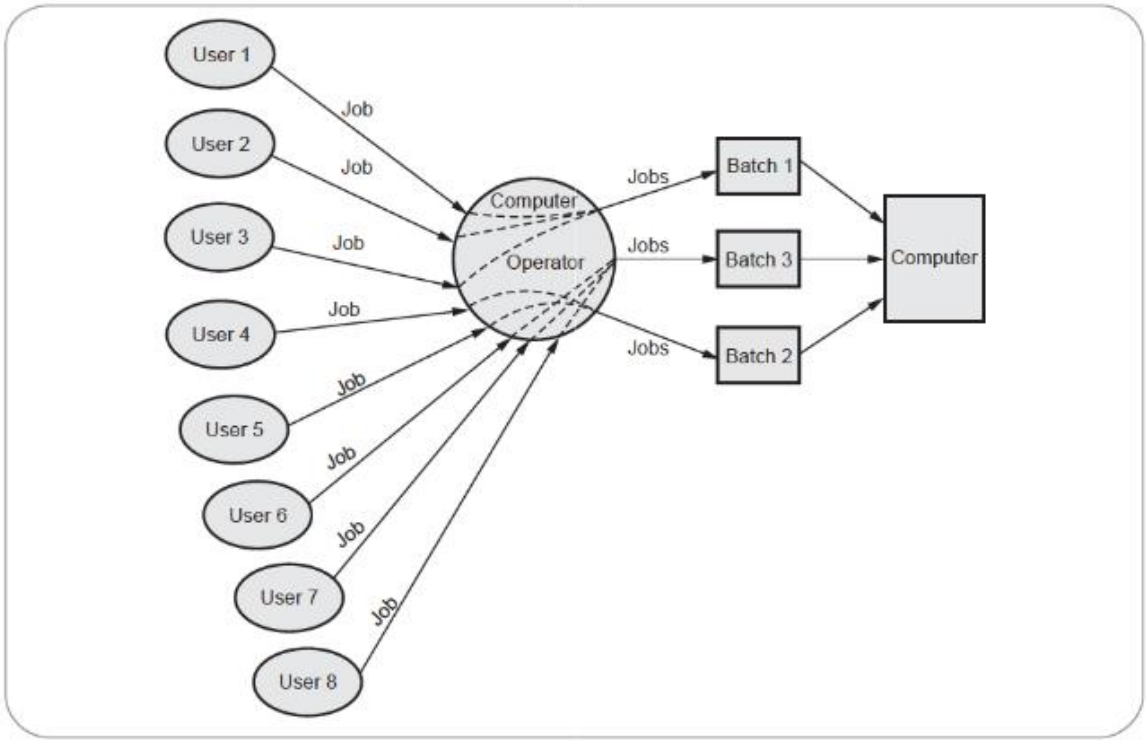


Fig. 1.5.2 Concept of batch system

same page, getting their output completely mixed together. It also helps in reducing idle time and overlapped I/O and CPU.

- Batch system often provides simple forms of file management. Access to file is serial. Batch systems do not require any time critical device management.
- Batch systems are inconvenient for users because users cannot interact with their jobs to fix problems. There may also be long turnaround times. Example of this system is generating monthly bank statement.
- An optimization used to minimize the discrepancy between CPU and I/O speeds is spooling. It overlaps of one job with computation of other job.
- The spooler for instance could be reading the input of one job while printing the output of different job.
- Spooling refers to putting jobs in a buffer, a special area in memory or on a disk where a device can access them when it is ready.
- Spooling is useful because device access data at different rates. The buffer provides a waiting station where data can rest while the slower device catches up.
- Computer can perform I/O in parallel with computation, it becomes possible to have the computer read a deck of cards to a tape, drum or disk and to write out to a tape printer while it was computing. This process is called **spooling.**
- The most common spooling application is print spooling.
- Spooling batch system were the first and are the simplest of the multiprogramming systems.

Advantages of spooling

1. The spooling operation uses a disk as a very large buffer.
2. Spooling is however capable of overlapping I/O operation for one job with processor operations for another job.

Advantages of batch system

1. Move much of the work of the operator to the computer.
2. Increased performance since it was possible for job to start as soon as the previous job finished.

Disadvantages of batch system

1. Turn around time can be large from user standpoint.
2. Program debugging is difficult.
3. There was possibility of entering jobs in infinite loop.
4. A job could corrupt the monitor, thus affecting pending jobs.
5. Due to lack of protection scheme, one batch job can affect pending jobs.

1.5.2 Multiprogrammed OS

- CPU remains idle in batch system. At any time either CPU or I/O device was idle in batch system. To keep CPU busy, more than one program/job must be loaded for execution. It increases the CPU utilizations. So multiprogramming increases the CPU utilization.
- Resource management is the main aim of multiprogramming operating system. File system, command processor, I/O control system and transient area are the essential components of a single user operating system. Multiprogramming operating system divides the transient area to store the multiple programs and provides resource management to the operating system.
- The concurrent execution of programs improves the utilization of system resources. A program in execution is called a "Process", "Job" or a "Task".
- When two or more programs are in the memory at the same time, sharing the processor is referred to the multiprogramming operating system.
- Fig. 1.5.3 shows the memory layout for a multiprogramming operating system.
- Operating system keeps number of programs into the memory. It selects one program from the memory and executes it. All the programs that enter the system are kept in the job pool.
- Job pool consists of all processes residing on disk and waiting to allocate primary memory. Job scheduling concept is used if there is no space for process in the primary memory.

Fig. 1.5.3 Multiprogramming OS memory layout

- Memory management is also required to manage the memory for process. CPU scheduling is applied for selecting process from memory.
- When computer loads more than one program in to the memory, CPU executes one program and I/O system is busy with other programs.
- Multiprogramming operating system do not provide user interaction with the program.
- Fig. 1.5.4 Shows working of multiprogramming OS.
- In multiprogramming operating system, programs are competing for resources. A function of the multiprogramming operating system is CPU scheduling, memory management and I/O management.
- Suppose there are four programs for execution. All four programs are loaded into the memory. CPU select first program for execution. Normally programs contain instruction for CPU and I/O operation.
- CPU bound instructions : c = a + b
- I/O bound instructions : printf, scanf etc.
- When any I/O instruction is encounter in the program, CPU select next program for processing. CPU select second program for execution and I/O system select first program for performing I/O operation. Multiprogramming operating system monitors the state of all active programs and system resources.

Advantages :

1. CPU utilization is high.
2. It increases the degree of multiprogramming.

Disadvantages :

1. CPU scheduling is required.
2. Memory management is also required.

1.5.3 Time Shared OS

- Time sharing is also called multitasking operating system. It is logical extension of the multiprogramming operating systems.
- User interaction with program is possible in time sharing operating system. During execution of the

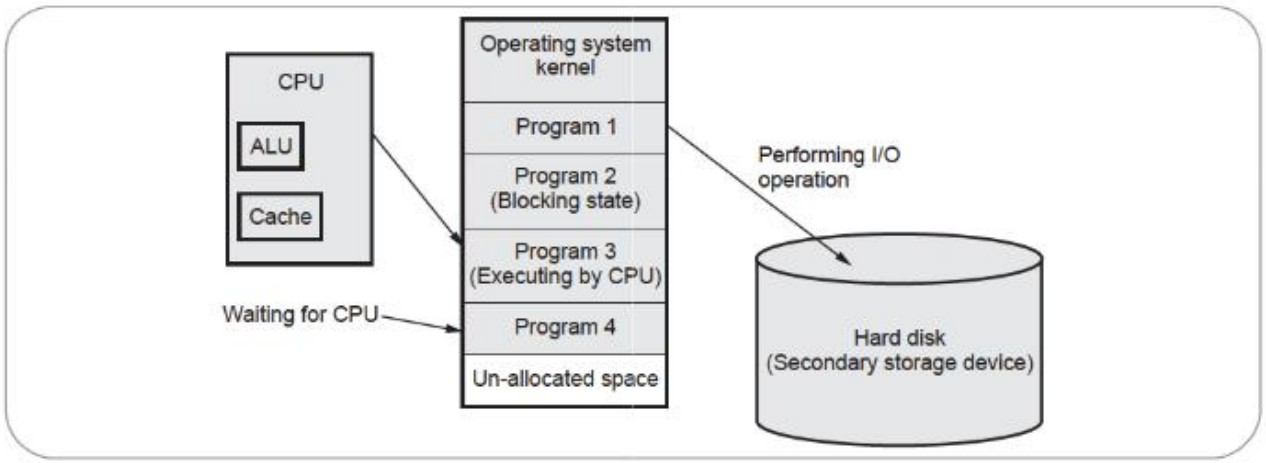


Fig. 1.5.4 Working of multiprogramming OS

program, user interacts directly with the program, supplying information to the program.

- Multi-tasking means that the computer can work with more than one program at a time. For example, user could be working with information from one database on the screen analyzing data, while the computer is sorting information from another database, while a excel sheet is performing calculations on a separate worksheet.
- Many users share the computer system simultaneously in time sharing operating system. Time sharing system uses multiprogramming and CPU scheduling. Each user has at least one separate program in memory.
- In time sharing system, each user is given a time slice for executing his/her job in round robin fashion. Job continues until the time slice ends.
- Fig. 1.5.5 shows multitasking OS.

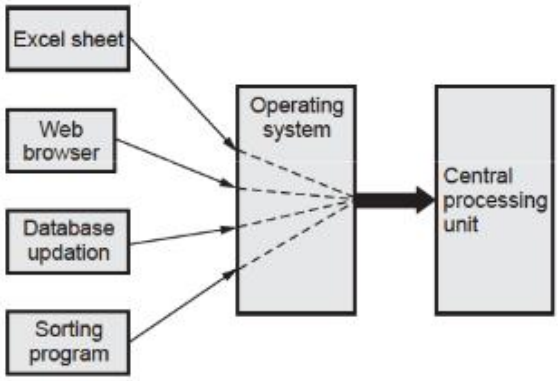


Fig. 1.5.5 Multitasking OS

- Concept of virtual machine is used in time sharing system. It creates virtual machine one per user. User interaction with system by using virtual machine. User enters the command for virtual machine and result will received back to user.
- Time sharing system is more complex than multiprogramming operating system. It also takes help of file system. File system is stored on the disk so disk management is also required.
- Major problem with time sharing system is protection and security of data.
- Time sharing system uses medium term scheduling such as round robin for the foreground. Background process uses can use a different scheduling method.
- Difference between multiprogramming and multitasking operating system is context switching. In multiprogramming system a context switching occurs only when the currently executing process stalls for some reasons. Time sharing system gives each user the impression that the entire system is dedicated to his use. Context switching simply allows several applications to be open, but only one is working at a time.
- Truly speaking, even in true multi-tasking, only one application program is ever running at anyone instant. Because the computer automatically switches from one program to the next program so quickly, all the programs seem to run simultaneously.

1.5.4 Difference between Time Sharing OS and Multiprogramming OS

Time sharing OS	Multiprogramming OS
It enables execution of multiple tasks and processes at the same processes to increases CPU performance.	Multiple programs reside in the main memory simultaneously to improve CPU utilization.
it is based on the concept of time sharing	It is based on the concept of context switching
The idea is to allow multiple processes to run simultaneously via time sharing	The idea is to reduce the CPU idle time for as long as possible.
It takes less time to execute the task allocation	It takes more time to execute the process.
Principle objective is minimize response time	Principle objective is maximize processor use

Board Questions

1. *Define multiprogramming system with diagram.* **MSBTE : Winter-15, Marks 4**
2. *Explain time sharing operating system.* **MSBTE : Summer-15, Marks 4**
3. *Explain batch operati* **MSBTE : Summer-15, Marks 4**
4. *Describe multiprogramming and multitasking.* **MSBTE : Winter-17, Marks 4**
5. *Explain time sharing operating system and state its advantages and disadvantages.* **MSBTE : Winter-17, Marks 4**

6. *Explain multiprogrammed O.S. with suitable diagram.* **MSBTE : Summer-17, Marks 4**
7. *Differentiate between multitasking and multiprogramming.* **MSBTE : Winter-18, Marks 4**
8. *Explain batch monitoring functions.* **MSBTE : Winter-18, Marks 4**
9. *Explain time sharing OS in detail.* **MSBTE : Summer-18, Marks 4**
10. *List advantages and disadvantages of batch monitoring functions. (Four points)* **MSBTE : Summer-18, Marks 4**

1.6 Multiprocessor OS

- Multiprocessor system means more than one processor in close communication. All the processor share common bus, clock, memory and peripheral devices.
- Multiprocessor system is also called parallel system or tightly coupled systems. Multiprocessor operating systems are just regular operating systems. They handle system calls, do memory management, provide a file system, and manage I/O devices.
- Multiprocessor systems is also known as *parallel systems* or *multicore systems*.
- **Features of Multiprocessor Systems :**
  1. The processor should support efficient context switching operation.
  2. It supports large physical address space and larger virtual address space.
  3. If one processor fails, then other processors should retrieve the interrupted process state so that execution of the process can continue.
  4. The inter-process communication mechanism should be provided and implemented in hardware as it becomes efficient and easy.
- **Multiprocessors can be used in different ways :**
  1. Uniprossesors (single-instruction, single-data or SISD)
  2. Within a single system to execute multiple, independent sequences of instructions in multiple contexts (multiple-instruction, multiple-data or MIMD);
  3. A single sequence of instructions in multiple contexts (single-instruction, multiple-data or SIMD, often used in vector processing);
  4. Multiple sequences of instructions in a single context (multiple-instruction, single-data or MISD, used for redundancy in fail-safe systems and sometimes applied to describe pipelined processors or hyper threading).

1.6.1 Advantages and Disadvantages of Multiprocessor Systems

- **Advantages of Multiprocessor Systems :**
1. **Throughput** : Throughput increases because number of processor is increases.
2. **Cost** : Multiprocessor system is cheaper than the multiple single processor system.
3. **Reliability** : If one processor fails, it has no effect on whole system operation.
4. **Response time** : Response time is less because of increased number of processor

- **Disadvantages of Multiprocessor Systems**
1. Multiprocessor systems are more complex in both hardware and software.
2. Additional CPU cycles are required to manage the cooperation, so per-CPU efficiency goes down.

- Multiprocessor systems are of two types: symmetric multiprocessing and asymmetric multiprocessing.

1.6.2 Symmetric Multiprocessing

- The simplest multiprocessors are based on a single bus. In symmetric multiprocessing, number of homogeneous processor are running independently without affecting other programs.
- Systems that treat all CPUs equally are called symmetric multiprocessing (SMP) systems.
- Each processor uses different data and program but sharing some common resources like memory, I/O device etc. Fig. 1.6.1 shows symmetric multiprogramming system.
- In SMP, all processor are at the same level. They work in peers. There is no master-slave relationship in between the processor. Each processor shares a single copy of operating system.

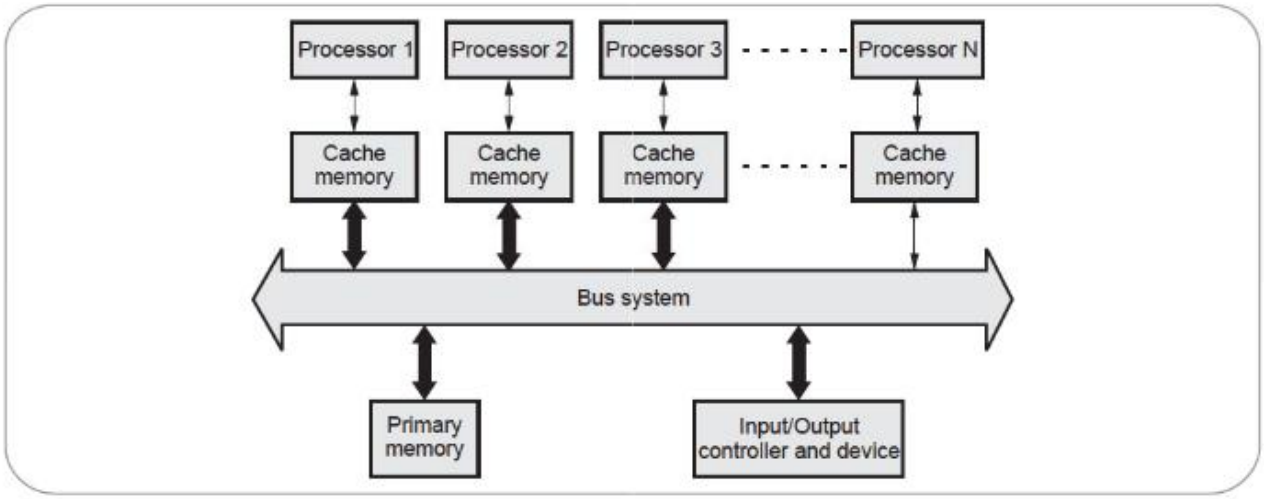


Fig. 1.6.1 Symmetric multiprogramming system

- Symmetric multiprocessing is easier to implement in operating systems. Examples of symmetric multiprocessing operating systems are Windows NT 4.0, Novell Netware 2.1 etc.

1.6.3 Asymmetric Multiprocessor

- If all CPUs are not equal, system resources may be divided in a number of ways, including asymmetric multiprocessing (ASMP), non-uniform memory access (NUMA) multiprocessing, and clustered multiprocessing.
- All CPUs are not equal. There is one master processor and remaining is the slave processor. Each processor has its own memory address space.
- A master processor controls the whole operations. It gives the instruction to the slave processor or slave processor uses some predefined set of tasks.
- Asymmetric multiprocessing has one master CPU and the remainder CPUs are slaves. The Master distributes takes among the slaves and I/O is usually done by the master only.
- Each processor has own task which assign by master processor. Asymmetric multiprocessing is difficult to implement.

1.6.4 Difference between Symmetric and Asymmetric Multiprocessor

Symmetric Multiprocessor	Asymmetric Multiprocessor
Symmetric multiprocessing treats all processors are equals, an I/O can be processed on any CPU.	Asymmetric multiprocessing has one master CPU and the remainder CPUs are slaves.
Symmetric multiprocessing is easier to implement in operating systems	Asymmetric multiprocessing is difficult to implement.
Single OS manages all processor cores simultaneously.	Separate OS, or separate copy of same OS manage each core
Master-slave concept is not used.	It uses concept of master-slave concept.
The processors communicate with each other through shared memory.	Shared memory is not used for communication.

1.6.5 Differentiate between Multiprocessing and Multiprogramming OS

Multiprocessing	Multiprogramming
Multiprocessing refers to processing of multiple processes at same time by multiple CPUs.	Multiprogramming keeps several programs in main memory at the same time and execute them concurrently utilizing single CPU.
It utilizes multiple CPUs	It utilizes single CPU

It permits parallel processing	Context switching takes place.
Less time taken to process the jobs.	More Time taken to process the jobs
It is used in parallel operating system	It is used in batch operating system.
More expensive system	Less expensive system

Board Questions

1. *List different types of operating systems. Explain advantages of multiprocessor system (any two).* **MSBTE : Winter-15, Marks 4**
2. *What is multiprocessor system ? Give two advantages of it.* **MSBTE : Summer-16, Marks 4**
3. *Explain multiprocessor system and its two types.* **MSBTE : Summer-17, Marks 4**

1.7 Distributed OS

- Definition : A distributed system is a collection of autonomous hosts that are connected through a computer network.
- A distributed system is a collection of independent computers that appears to its users as a single coherent system. Each host executes components and operates a distribution middleware.
- Middleware enables the components to coordinate their activities. Users perceive the system as a single, integrated computing facility.
- A distributed computer system consists of multiple software components that are on multiple computers, but run as a single system. The computers that are in a distributed system can be physically close together and connected by a local network, or they can be geographically distant and connected by a wide area network.
- A distributed system can consist of any number of possible configurations, such as mainframes, personal computers, workstations, minicomputers and so on.
- Distributed operating systems depend on networking for their operation. Distributed OS runs on and controls the resources of multiple machines. It provides resource sharing across the boundaries of a single computer system. It looks to users like a single machine OS.
- Distributing OS owns the whole network and makes it look like a virtual uni-processor or may be a virtual multiprocessor.
- Definition : A distributed operating system is one that looks to its users like an ordinary operating system but runs on multiple, independent CPU.
- Distributed systems depend on networking for their functionality. Fig. 1.7.1 shows the distributed system.
- Examples of distributed operating system are Amoeba, chrous, mach and v-system
- A **Local Area Network** (LAN) is a network that is confined to a relatively small area. It is generally limited to a geographic area such as a college, lab or building.
- **WAN** provides long distance transmission of data and voice. Computers connected to a wide-area network are often connected through public networks, such as the telephone system. They can also be connected through leased lines or satellites.
- A **MAN** typically covers an area of between 5 and 50 km diameter. Many MANs cover an area the size of a city, although in some cases MANs may be as small as a group of buildings.

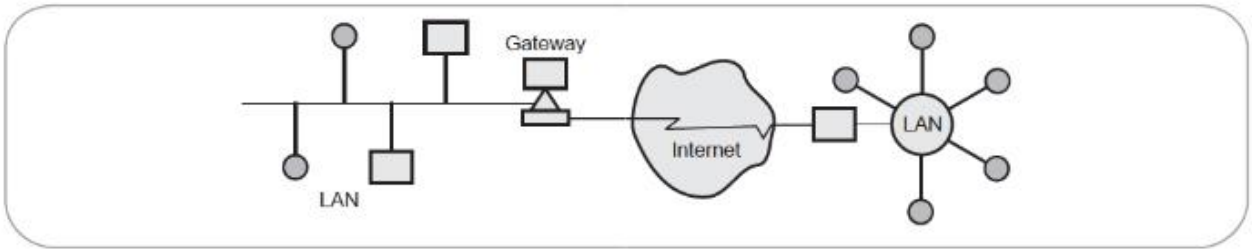


Fig. 1.7.1 Distributed system

- A MAN often acts as a high speed network to allow sharing of regional resources. MAN provides the transfer rates from 34 to 150 Mbps.

Advantages of distributed OS :

1. **Resource sharing :** Sharing of software resources such as software libraries, database and hardware resources such as hard disks, printers and CDROM can also be done in a very effective way among all the computers and the users.
2. **Higher reliability :** Reliability refers to the degree of tolerance against errors and component failures. Availability is one of the important aspects of reliability. Availability refers to the fraction of time for which a system is available for use.
3. **Better price performance ratio :** Reduction in the price of microprocessor and increasing computing power gives good price-performance ratio.
4. **Shorter responses times and higher throughput.**
5. **Incremental growth :** To extend power and functionality of a system by simply adding additional resources to the system

Difficulties in distributed OS are

1. There are no current commercially successful examples.
2. Protocol overhead can dominate computation costs.
3. Hard to build well.
4. Probably impossible to build at the scale of the internet.

1.7.1 Client-Server Computing

- The system is structured as a set of processes, called servers that offer services to the users, called clients.
- Server systems are of two types : compute servers and file servers
- **Compute-server system :** It provides an interface to which a client can send a request to perform an action. In response, the server performs some operation and sends the results to the client. Server contains database.
- **File-server system :** Client can performs various operation like create, read, update and delete file on file server. Web server is best example of this type.
- Fig. 1.7.2 shows the client server model.

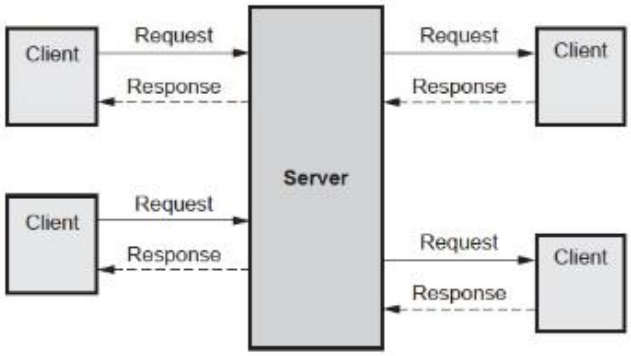


Fig. 1.7.2 Client server model

- The client-server model is usually based on a simple request/reply protocol, implemented with send/receive primitives or using Remote Procedure Calls (RPC) or Remote Method Invocation (RMI) :
  1. The client sends a request (invocation) message to the server asking for some service;
  2. The server does the work and returns a result (e.g. the data requested) or an error code if the work could not be performed.
- Server is a process; Client is a process. Clients invoke servers, servers send results to clients. Servers can be clients of other servers.
- HTTP (Web) server is a server process to its client processes (web browsers). HTTP server may be a client of a database server. Service may be provided by multiple servers, as is most often the case within a large enterprise.
- Cache is a repository of recently accessed objects (files, graphics) that is physically closer to the client than the server from which it originated. Proxy server sits in between clients and servers and can play many mitigation roles.

Advantages :

1. Simple to implement.
2. Provides good security
3. All files are stored in a central location.

Disadvantages :

1. Single point of failure
2. A specialist network operating system is needed.

1.7.2 Peer-to-Peer System

- All processes (objects) play similar role. Do not require a server process. Processes (objects) interact without particular distinction between clients and servers.
- The pattern of communication depends on the particular application. Fig. 1.7.3 shows the peer-to-peer model.

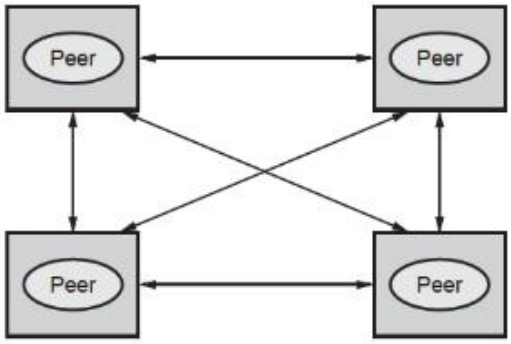


Fig. 1.7.3 Peer to peer model

- A large number of data objects are shared; any individual computer holds only a small part of the application database. Processing and communication loads for access to objects are distributed across many computers and access links. This is the most general and flexible model.
- A group of computers connected together to combine their computing and processing abilities to search the Internet or solve very complex problems
- Problems with peer-to-peer : High complexity due to
  1. Cleverly place individual objects
  2. Retrieve the objects
  3. Maintain potentially large number of replicas.

1.7.3 Distinguish between Client - Server and Peer-to-Peer Model

Client–server model	Peer-to-peer model
The client-server model firmly distinguishes the roles of the client and server. Under this model, the client requests services that are provided by the server	The peer-to-peer model doesn't have such strict roles. In fact, all nodes in the system are considered peers and thus may act as either clients or servers or both
Single point of failure	No single point of failure
Need for dedicated application and database servers	No need for dedicated application and database servers
Resources may be removed at any time	Resources are not removed from the network until they are no longer being requested.
Storage and bandwidth must be provided by the host.	Storage and bandwidth are distributed and provided by the entire network.
Provides good security	Provides Poor security

Board Questions

1. *Describe distributed operating system.*
   MSBTE : Winter-16, Marks 4
2. *Explain distributed system in detail.*
   MSBTE : Winter-17, Marks 4

1.8 Real Time Systems

- Time constraints is the key parameter is real time systems. It controls autonomous system such as robots, satellites, air traffic control and hydroelectric dams.
- When user gives an input to the system, it must process within the time limit and result is sent back. Real time system fails if it does not give result within the time limits.
- Real time systems are of two types : **Hard real time and soft real time**.
- Critical task is completed within the time limit in hard real time system. All the delay in the system is fixed and time bounded. Existing general purpose operating system does not support the hard real time system functions. Real time task cannot keep waiting for longer time without allocating kernel.

- Soft real time system is less restrictive type. Soft real time cannot guarantee that it will be able to meet deadline under all condition. Example of soft real time system is digital telephone and digital audio.
- Real time operating system uses priority scheduling algorithm to meet the response requirement of a real time application. General real time applications with their example are listed below :

1. Transportation : Air traffic control and traffic light system
2. Communication : Digital telephone
3. Process control : Petroleum and paper mill
4. Detection : Burglar system and radar system
5. Flight simulation : Auto pilot shuttle mission simulator.

- Memory management in real time system is less demanding than in other type of multiprogramming operating systems. Time critical device management is one of the main characteristics of the real time systems.

Board Questions

1. *What is real time system ? Explain its types.* **MSBTE : Winter-15, Marks 4**
2. *Define real time operating system. Explain with the help of example.* **MSBTE : Winter-16, Marks 4**
3. *Describe real time operating system in brief.* **MSBTE : Summer-16, Marks 4**
4. *Describe real time system. State any one example of its application.* **MSBTE : Winter-17, 18, Marks 4**
5. *With neat diagram, explain real time system. List its any four application.* **MSBTE : Summer-17, Marks 4**

1.9 Mobile OS

1.9.1 Andriod

- Android is an open source mobile OS developed by the Open Handset Alliance led by Google.
- Android is a software stack for mobile devices that includes an operating system, middleware and key applications.
- It is based on Linux 2.6 kernel.
- Android is an open source operating system, created by Google specifically for use on mobile devices (i.e. cell phones and tablets)
- It can be programmed in C/C++ but most app development is done in Java. It supports Bluetooth, Wi-Fi and 3G and 4G networking.

Android versions	Features
Android 1.1	1.API changes 2.Support for saving attachments for MMS
Android 1.5	1.Bluetooth A2DP and AVRCP support 2.Uploading video to You Tube and pictures
Android 1.6	1.Google free turn to turn support 2.WVGA screen resolution support
Android 2.0/2.1	1.HTML5 file support 2.Bluetooth 2.1 3.Microsoft exchange server
Android 2.2	1.Adobe flash 10.1 support 2.Wi-Fi hotspot support functionality
Android 2.3	1.Multi touch software keyboard 2.Support for extra large screen sizes and resolution
Android 3.0	1.3D desktop 2.Video chat 3.Optimized tablet support with new user interface

1.9.1.1 Android Architecture

- Fig. 1.9.1 shows Android software stack. Each layer of the stack and the corresponding elements within each layer are tightly integrated and carefully tuned to provide the optimal application development and execution environment for mobile devices.
- Android provides a set of core applications :
  1. Email client
  2. SMS program
  3. Calendar
  4. Maps
  5. Browser
  6. Contacts

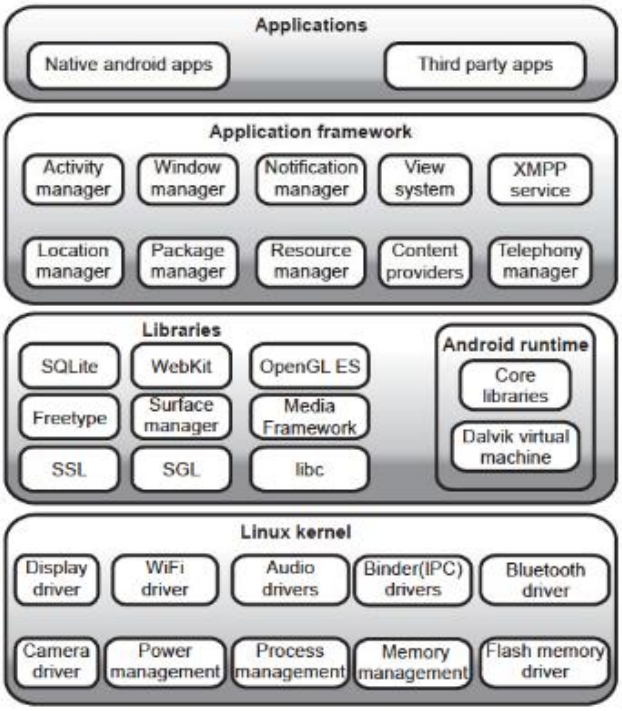


Fig. 1.9.1 Android architecture

- All applications are written using the Java language.

App framework

- Used for enabling and simplifying the reuse of components
  1. Developers have full access to the same framework APIs used by the core applications.
  2. Users are allowed to replace components.
- App Framework features are as follows :

Feature	Role
View system	Used to build an application, including lists, grids, text boxes, buttons and embedded web browser
Content provider	Enabling applications to access data from other applications or to share their own data
Resource manager	Providing access to non-code resources (localized strings, graphics and layout files)
Notification manager	Enabling all applications to display customer alerts in the status bar
Activity manager	Managing the lifecycle of applications and providing a common navigation back stack

Libraries

- Libraries include a set of C/C++ libraries used by components of the Android system. It is exposed to developers through the Android application framework

Runtime

- Android run-time system provides core set of class libraries to ensure smooth platform for developers. With these libraries developers can easily import required libraries into their applications without doing any hard coding in applications.

Dalvik virtual machine

- Dalvik is a purpose built virtual machine designed specifically for android which was developed by Dan Bornstein and his team. Strictly it was developed for mobile devices. While developing Dalvik Virtual Machine Dan Bornstein and his team realize the constraints specific to mobile environment which is not going to change in future at least, like battery life,processing power and many more. So they optimized the dalvik virtual machine. Dalvik virtual machine uses register based architecture. With this architecture dalvik virtual machine has few advantages over java virtual machine such as :
  1. Dalvik uses its own 16 bit instruction set than java 8 bit stack instructions, which reduce the dalvik instruction count and raised its interpreter speed.
  2. Dalvik use less space, which means an uncompressed .dex file is smaller in size(few bytes) than compressed java archive file(.jar file).
- An open source software stack that includes operating system. Linux operating system kernel that provides low level interface with the hardware, memory management and process control.

- Middleware : A run time to execute Android applications including virtual machine and core libraries.

Important blocks in Android :

1. **Activity manager :** Manages the activity life cycle of applications
2. **Content providers :** Manage the data sharing between applications
3. **Telephony manager :** Manages all voice calls. We use telephony manager if we want to access voice calls in our application.
4. **Location manager :** Location management, using GPS or cell tower
5. **Resource manager :** Manage the various types of resources we use in our application

Android SDK features

The Android SDK includes

1. The Android APIs.
2. The core of the SDK.
3. Development tools.
4. No licensing, distributions, or development fees or release approval processes.
5. GSM, EDGE, and 3G networks for telephony and data transfer.
6. Full multimedia hardware control.
7. APIs for using sensor hardware including accelerometer and the compass.
8. APIs for location based services.

Application framework

1. Android offers developers the ability to build rich and innovative applications.
2. Developers have full access to the **same** framework APIs used by the core applications.
3. Underlying all applications is a set of services, including Views.
4. It can be used to build an application, including lists, grids, text boxes, buttons, and even an embeddable web browser.

- Content providers enable applications to access data from other applications (such as Contacts), or to share their own data.
- A resource manager provides access to non-code resources such as localized strings, graphics and layout files.
- A notification manager enables all applications to display custom alerts in the status bar.
- An activity manager manages the lifecycle of applications and provides a common navigation backstack.

Libraries used in Android

- A set of C/C++ libraries used by various components of the Android system.
- System C library : Tuned for embedded Linux-based devices .
- Media Libraries : Based on PacketVideo's OpenCORE; the libraries support playback and recording of many popular audio and video formats, as well as static image files.
- Surface Manager : Manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications.
- LibWebCore : A modern web browser engine which powers both the Android browser and an embeddable web view.
- SGL/ 3D libraries : SGL is underlying 2D graphics engine.
- SQLite : A powerful and lightweight relational database engine available to all applications.

Android run-time

- Android includes a set of core libraries that most of the functionality available in the core libraries of the Java programming language.
- Every Android app runs in its own process with its own instance of the Dalvik virtual machine. The Dalvik VM executes files in the Dalvik Executable (.dex) format.

Slow Android apps

1. By default on Android, all work is done in a single thread, the "main application" thread. If a component of the work takes a long time, the rest of the work will be "blocked". For example, a

long time to access data across the network prevents responding to any GUI events.

2. In the Android OS, if a GUI doesn't respond to an input event in five seconds, then it is considered unresponsive and the OS will try to kill it.

Android thread design

1. Only perform GUI actions on main application thread. Spawn separate threads to perform data-intensive or slow actions. Make these threads asynchronous.
2. Main thread does not have to wait for/check on other threads. Instead, those threads run as they need to and report back to the original thread. Any changes made to the UI should go through the UI thread.

1.9.1.2 Comparison of Android OS Vs iPhone OS Features

Android OS	iPhone OS
Limited internal memory. Big headache because apart from photos and media content, the default memory is already limited	Good internal memory and user have choice of different internal memory sizes
It supports external memory. External SD card can be inserted to store photos, media, etc.	No external expandable memory. But no complaints here because the internal memory itself is huge and good enough
File Transfer/Synchronization is easy. Plug and transfer the data in flash drive mode.	File Transfer/ Synchronization is just like iPods.
It broadcasts the system-wide notifications and other application notifications in cascade windows which can be pulled down to see details. Register Listener, use callback to read the information	Push notifications and individual notifications on updates. It saves power because server pushes data. No need to poll and pull data
Hardware Design/Buttons : Menu and back buttons do not always do the same thing. The functionalities vary for different apps.	Hardware Design/Buttons : One button is used for one clear function. Menu and back always mean what they imply.

Android applications

a. Android applications get distributed in a .apk file. APK stands for "Android Package". It is simply a zip file that has a particular file structure. An APK contains :
   1. The Android Manifest file (an XML file with lots of metadata).
   2. A Resource bundle containing sounds, graphics, etc.
   3. The Dalvik classes that make up user application.

1.9.1.3 Android Benefits

1. An open and free development platform. Handset makers can use it without royalty and customize to their hearts content.
2. Component-based architecture : Lots of default components can be replaced straightforwardly.

Proponents of Android point to the following benefits :

a. Lots of services : location, sql, maps, web, etc.

b. Well managed applications; isolated from each other to protect data and provide security;

c. Operating system can quit programs as needed to ensure good performance on mobile devices.

d. Portability : To support a new device, a company has to port the virtual machine; Android apps (Dalvik) then execute on the new device with little to no modification.

1.9.2 iOS

- iOS is the operating system that runs on iPad, iPhone and iPod touch devices. The operating system manages the device hardware and provides the technologies required to implement native apps.
- The iOS Software Development Kit (SDK) contains the tools and interfaces needed to develop, install, run, and test native apps that appear on an iOS device's Home screen.
- Fig. 1.9.2 shows iOS architecture.
- The iOS Architecture is layered. At the highest level, iOS acts as an intermediary between the underlying hardware and the apps you create.

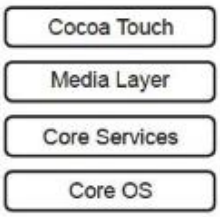


Fig. 1.9.2 iOS architecture

- Apps do not talk to the underlying hardware directly. Instead, they communicate with the hardware through a set of well-defined system interfaces. These interfaces make it easy to write apps that work consistently on devices having different hardware capabilities.
- The **Cocoa Touch layer** contains key frameworks for building iOS apps. These frameworks define the appearance of your app. They also provide the basic app infrastructure and support for key technologies such as multitasking, touch-based input, push notifications and many high-level system services.
- High-Level features of Cocoa touch layers are AirDrop, Multitasking, Auto Layout, Storyboards and Local Notifications And Apple Push Notification Service.
- Cocoa touch layer contains following frameworks for iPhone app development:
  a. UIKit framework
  b. Map kit framework
  c. Push notification service.
  d. Message UI framework
  e. Address book UI framework
  f. Game kit framework
  g. iAd framework
  h. Event kit UI framework
  i. Accounts framework
  j. Twitter framework
- The **Media layer contains** the graphics, audio and video technologies you use to implement multimedia experiences in your apps. The technologies in this layer make it easy for you to build apps that look and sound great.

1.9.2.1 Media Layer

- The role of the Media layer is to provide iOS with audio, video, animation and graphics capabilities.
- As with the other layers comprising the iOS stack, the media layer comprises a number of frameworks which may be utilized when developing iPhone apps.
- The technologies in this layer make it easy for you to build apps that look and sound great.
- Features of media layer :
  a. Graphics technologies
  b. Audio technologies
  c. Video technologies
  d. AirPlay
- Media layer contains following frameworks :

1. Core video framework : This framework provides buffering support for the Core media framework. Whilst this may be utilized by application developers it is typically not necessary to use this framework.
2. Core text framework : The iOS core text framework is a C-based API designed to ease the handling of advanced text layout and font rendering requirements.
3. Image I/O framework : The Image I/O framework, the purpose of which is to facilitate the importing and exporting of image data and image metadata, was introduced in iOS 4. The framework supports a wide range of image formats including PNG, JPEG, TIFF and GIF.
4. Assets library framework : The assets library provides a mechanism for locating and retrieving video and photo files located on the iPhone device. In addition to accessing existing images and videos, this framework also allows new photos and videos to be saved to the standard device photo album.
5. Core graphics framework : The iOS core graphics framework provides a lightweight two dimensional rendering engine. Features of this framework include PDF document creation and presentation, vector based drawing, transparent layers, path based drawing, anti-aliased

rendering, color manipulation and management, image rendering and gradients.

6. Core image framework : A new framework introduced with iOS 5 providing a set of video and image filtering and manipulation capabilities for application developers.
7. Quartz core framework : The purpose of the Quartz Core framework is to provide animation capabilities on the iPhone. It provides the foundation for the majority of the visual effects and animation used by the UIKit framework and provides an Objective-C based programming interface for creation of specialized animation within iPhone apps.
8. OpenGL ES framework : For many years the industry standard for high performance 2D and 3D graphics drawing has been OpenGL. OpenGL for Embedded Systems (ES) is a lightweight version of the full OpenGL specification designed specifically for smaller devices such as the iPhone.
9. GLKit framework : The GLKit framework is an Objective-C based API designed to ease the task of creating OpenGL ES based applications.
10. NewsstandKit framework : The Newsstand application is a new feature of iOS 5 and is intended as a central location for users to gain access to newspapers and magazines. The NewsstandKit framework allows for the development of applications that utilize this new service.
11. iOS audio support : iOS is capable of supporting audio in AAC, Apple Lossless (ALAC), A-law, IMA/ADPCM, Linear PCM, μ-law, DVI/Intel IMA ADPCM, Microsoft GSM 6.10 and AES3-2003 formats through the support provided by the following frameworks.
12. AV foundation framework : An Objective-C based framework designed to allow the playback, recording and management of audio content.

1.9.2.2 Core Services Layer

- The Core Services layer contains fundamental system services for apps.
- Key among these services are the core foundation and foundation frameworks, which define the basic types that all apps use.
- This layer also contains individual technologies to support features such as location, iCloud, social media and networking.
- Features :

1. Peer-to-Peer services
2. iCloud storage
3. Automatic reference counting
4. Block objects
5. Grand central dispatch
6. In-App purchase
7. SQLite
8. XML support
9. File-sharing support, data protection

- It consists of the following frameworks.
- Address book framework : This provides programmatic access to the iPhone Address Book contact database allowing applications to retrieve and modify contact entries.
- CFNetwork framework : The CFNetwork framework provides a C-based interface to the TCP/IP networking protocol stack and low level access to BSD sockets. This enables application code to be written that works with HTTP, FTP and domain name servers and to establish secure and encrypted connections using Secure Sockets Layer (SSL) or Transport Layer Security (TLS).
- Core Data Framework : This framework is provided to ease the creation of data modeling and storage in Model-View-Controller (MVC) based applications. Use of the Core Data framework significantly reduces the amount of code that needs to be written to perform common tasks when working with structured data within an application.
- Core foundation framework : The core foundation framework is a C-based Framework which provides basic functionality such as data types, string manipulation, raw block data management, URL manipulation, threads and run loops, date and times, basic XML manipulation and port and socket communication.

- The core media framework is the lower level foundation upon which the AV foundation layer is built.
- Core telephony framework : The iOS core telephony framework is provided to allow applications to interrogate the device for information about the current cell phone service provider and to receive notification of telephony related events.
- EventKit framework : An API designed to provide applications with access to the calendar and alarms on the device.
- Most applications will use iCloud document storage to share documents from a user's iCloud account. This is the feature that users think of when they think of iCloud storage. A user cares about whether documents are shared across devices and can see and manage those documents from a given device.
- Data protection allows applications that work with sensitive user data to take advantage of the built-in encryption available on some devices.
- When your application designates a specific file as protected, the system stores that file on-disk in an encrypted format. While the device is locked, the contents of the file are inaccessible to both your application and to any potential intruders.
- However, when the device is unlocked by the user, a decryption key is created to allow your application to access the file.

1.9.2.3 Core OS Layer

- The Core OS layer contains the low-level features that most other technologies are built upon
- Even if you do not use these technologies directly in your apps, they are most likely being used by other frameworks.
- And in situations where you need to explicitly deal with security or communicating with an external hardware accessory, you do so using the frameworks in this layer.
- This layer provides a variety of services including low level networking, access to external accessories and the usual fundamental operating system services such as memory management, file system handling and threads.
- The Core OS layer occupies the bottom position of the iOS stack and, as such, sits directly on top of the device hardware.
- The layer provides a variety of services including low level networking, access to external accessories and the usual fundamental operating system services such as memory management, file system handling and threads.
- Accelerate framework : Introduced in iOS 4.0, the Accelerate framework contains interfaces for performing DSP, linear algebra and image-processing calculations. The advantage of using this framework over writing your own versions of these interfaces is that they are optimized for all of the hardware configurations present in iOS, based devices. Therefore, you can write your code once and be assured that it runs efficiently on all devices.
- External accessory framework : It provides the ability to interrogate and communicate with external accessories connected physically to the iPhone via the 30-pin dock connector or wirelessly via Bluetooth.
- Security framework : The iOS Security framework provides all the security interfaces you would expect to find on a device that can connect to external networks including certificates, public and private keys, trust policies, key chains, encryption, digests and Hash-based Message Authentication Code (HMAC).
- The core bluetooth framework allows developers to interact specifically with Bluetooth Low-Energy ("LE") accessories. The Objective-C interfaces of this framework allow you to scan for LE accessories, connect and disconnect to ones you find, read and write attributes within a service, register for service and attribute change notifications and much more.
- System : The system level encompasses the kernel environment, drivers and low level UNIX interfaces of the operating system. The kernel itself is based on Mach and is responsible for every aspect of the operating system.
- It manages the virtual memory system, threads, file system, network and interprocess communication. The drivers at this layer also provide the interface

between the available hardware and system frameworks.

- For security purposes, access to the kernel and drivers is restricted to a limited set of system frameworks and applications.
- iOS provides a set of interfaces for accessing many low-level features of the operating system. Your application accesses these features through the LibSystem library.

1.10 Command Line based OS

- The command line is also called the Windows command line, command screen, or text interface. It is a user interface that is navigated by typing commands at prompts, instead of using the mouse.
- Unlike a GUI operating system, a command line only uses a keyboard to navigate by entering commands and does not utilize a mouse for navigating.
- Command Prompt is a command line interpreter application available in most Windows operating systems. It's used to execute entered commands.
- Most of those commands automate tasks via scripts and batch files, perform advanced administrative functions, and troubleshoot or solve certain kinds of Windows issues.

1.10.1 DOS

- Disk Operating System (DOS) is created by Microsoft : MS-DOS. MS-DOS is a disk operating system for IBM PC-compatible computers.
- As with any other operating system, its function is to oversee the operation of the system by providing support for executing programs, controlling I/O devices, handling errors, and providing the user interface.
- MS-DOS is a disk-based, single-user, single-task operating system. These qualities make it one of the easiest disk operating systems to understand.
- The main portions of MS-DOS are the *IO.SYS*, *MSDOS.SYS*, and *COMMAND.COM* files.
- IO.SYS and MSDOS.SYS are special, hidden system files
- The IO.SYS file moves the system's basic I/O functions into memory and then implements the MS-DOS default control programs, referred to as device drivers, for various hardware components.
- The COMMAND.COM command interpreter accepts commands issued through the keyboard, or other input device, and carries them out according to the commands definition
- When DOS runs an application, COMMAND.COM finds the program, loads it into memory, and then gives it control of the system. When the program is shut down, it passes control back to the command interpreter.
- Directory is just like a file folder, which contain all the logically related files. DOS files are organized in a hierarchical or an inverted tree-like structure.
- DOS enables the user to organize the files in a disk into directories and sub-directories. A directory within another directory is called a sub-directory

1.10.2 UNIX

- Unix was designed to be a time sharing system.
- Unix has a simple user interface that has the power to provide the services that user want.
- It uses a hierarchical file system that allows easy maintainance and efficient implementation.
- Unix file system is a multilevel tree.
- Each user data file is simply a sequence of bytes.
- It provides a simple, consistent interface to peripheral device.
- Unix supports multiple processes.
- Unix is a multiuser, multiprocess system.
- CPU scheduling in Unix is a simple priority algorithm.
- Swapping is used if a system is suffering from excess paging.
- Unix hides the machine architecture from the user.
- The system is written in high level language.
- Unix consists of two separable parts : The Kernel and the systems programs.
- The Kernel interacts with the machines hardware.
- Shell interacts with the user.
- Viewing the system as a set of layers, the operating system is commonly called the Kernel.

- The Kernel provides the file system, CPU scheduling, memory management through system calls.

1.10.2.1 Architecture of Unix OS

- There are three levels :

  a. User level b. Kernel level c. Hardware level.

- The file subsystem manages files, allcoating file space, controlling access to files and retrieving data for users. Processes interact with the file subsystem via a specific set of system calls.
- The file subsystem accesses file data using a buffering mechanism that regulates flow between the Kernel and secondary storage devices.
- The buffering mechanism interact with block I/O device drivers to initiate data transfer to and from the Kernel.
- Hardware control is responsible for handling interrupts and for communicating with the machine. Devices such as disks or terminals may interrupt the CPU while a process is executing.
- The Kernel may resume execution of the interrupted process after servicing the interrupt.
- The process control subsystem is responsible for process synchronization, IPC, memory management and process scheduling.
- The file subsystem and the process control subsystem interact when loading a file into memory for execution. The process subsystems reads executable files into memory before executing them.
- The memory management module controls the allocation of memory. If at any time the system does not have enough physical memory for all processes, the Kernel moves them between main memory and secondary memory so that all processes get a fair chance to execute.
- The scheduler module allocates the CPU to processes.
- The **scheduler** module allocates the CPU to processes. It schedules them to run in turn until they voluntarily relinquish the CPU while awaiting a resource or until the kernel preempts them when their recent run time exceeds a time quantum.
- The hardware control is responsible for handling interrupts and for communicating with the machine. Devices such as disks or terminals may interrupt the CPU while a process is executing.

1.11 GUI based OS

- The second approach allows the user to interface with the OS via a graphical user interface or GUI.
- Rather than having users directly enter commands via a command-line interface, a GUI allows provides a mouse-based window-and-menu system as an interface.
- Graphical user interfaces first appeared due in part to research taking place in the early 1970s at Xerox PARC research facility. The first GUI appeared on the Xerox Alto computer in 1973.
- However, graphical interfaces became more widespread with the advent of Apple Macintosh computers in the 1980s. Microsoft's first version of Windows-version 1.O-was based upon a GUI interface to the MS-DOS OS.
- Traditionally, UNIX systems have been dominated by command-line interfaces, although there are various GUI interfaces available.
- The choice of whether to use a command-line or GUI interface is mostly one of personal preference. As a very general rule, many UNIX users prefer a command-line interface as they often provide powerful shell interfaces.
- Alternatively, most Windows users are pleased to use the Windows GUI environment and almost never use the MS-DOS shell interface.

1.11.1 Windows

- Windows 7 is an operating system that Microsoft has produced for use on personal computers. It is the follow-up to the Windows Vista Operating System.
- The 64-bit version of Windows 7 is now commonly installed on larger client systems.
- Design goals are as follows :

1. Security
2. Reliability
3. Application compatibility of windows and POSIX
4. Good performance
5. Extensibility

6. Portability
7. Support.

- **Security :** Automatic analysis tools are used to test the security of windows source code. Windows XP receive security certificate (C-2 level Certificate) from US government. It uses Kerberos, access control list and packet filtering firewalls to protect the system.
- **Reliability :** Stable and most reliable windows operating system from Microsoft was windows 2000. Windows uses combination of hardware and software protection. Hardware protection for virtual memory and software protection for operating system resources.
- **Application compatibility of windows and POSIX :** IEEE 1003.1 is standard used by POSIX. Without changing source code of windows, user cannot run POSIX application on windows platform.
- **Performance :** Windows XP OS provides high performance for desktop systems. Message passing method is used for interprocess communication. Windows support preemption of low priority threads enables the system to respond quickly to external events.
- **Extensibility :** Extensibility means to support or add new technologies. Windows XP is layered architecture system. Kernel runs in protected mode. An environment subsystems operates in user mode providing different API.
- **Portability :** Windows XP support portability. Source code of windows XP is in C and C++ language. Windows 7 OS can be moved from one hardware platform to another with relatively few changes. Platform-dependent code is isolated in a Dynamic Link Library (DLL) called the "Hardware Abstraction Layer" (HAL).

1.11.1.1 Windows Architecture

- Windows architecture consists of hardware abstraction layer, kernel and executive. Fig. 1.11.1 shows windows XP architecture block diagram.

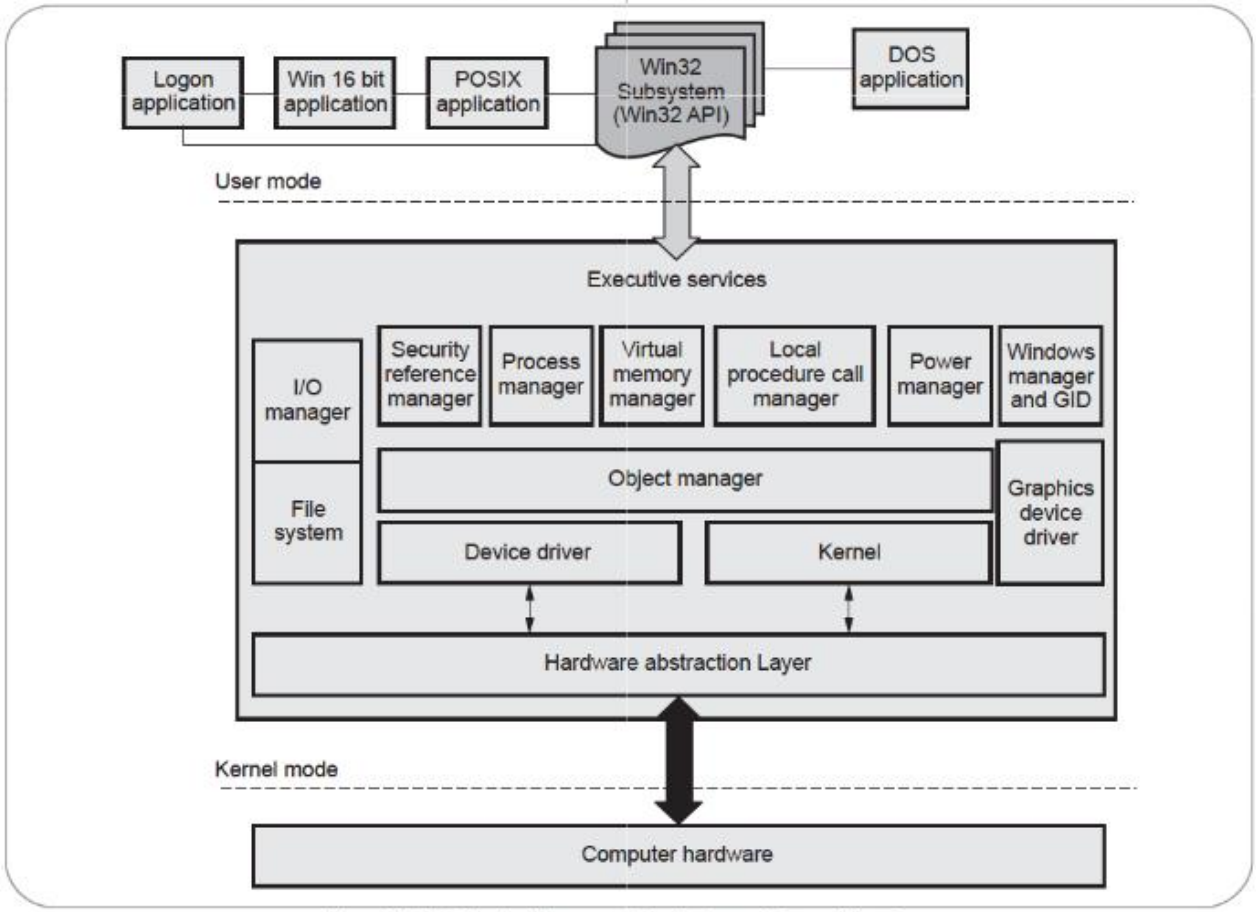


Fig. 1.11.1 Block diagram of windows XP architecture

- Windows XP architecture is based on microkernel architecture. It is modular operating system. Windows XP is also layered operating system.

Hardware Abstraction Layer (HAL)

- HAL interacts directly with hardware, abstracting hardware specifies to the rest of the system. HAL Also interacts with the microkernel. The microkernel and the HAL combine to make Windows XP portable allowing it to run in many different hardware environments.
- HAL is responsible for mapping various low level, processor specific operations into a fixed interface that is used by the XP kernel and executive.
- HAL provides support for symmetric multiprocessing. Device drivers map devices and access them directly.
- The function of HAL is to present the rest of the operating system with abstract hardware that hides the specific details of processor versions, support chipset and other configurations.
- HAL does not provide services for specific I/O devices such as mouse, keyboards and disks.
- Layered on the top of the microkernel are the kernel mode components responsible for administering the operating system subsystems. Collectively these components are known as the **executive.**
- HAL also manages clocks and timers in a portable way. After booting the system, HAL talks to the BIOS and checks the system configuration to find out which buses and I/O devices the system contains and how they have been configured.

Kernel Layer

- Kernel creates basic unit of computation and provides the foundation for multitasking support. The kernel provides objects and threads on top of the HAL and the hardware. Software that uses the kernel can be defined using objects and threads as primitives.
- Functions of kernel :

1. Thread scheduling
2. Interrupt and execution handling
3. Low level processor synchronization
4. Recovery after power failure.

- To implement objects and threads, the kernel must manage the hardware interrupts and exceptions, perform processor scheduling and handle multiprocessor synchronization.
- The kernel's thread scheduler is responsible for determining which thread is executing on each CPU in the system. When switching from one thread to another thread, the scheduler runs on the CPU and ensures that the registers and other hardware state have been saved.
- Thread state are ready, standby, running, waiting, transition and terminated. The dispatcher uses a 32 level priority scheme to determine the order of thread execution. Priorities are of two types: real time class and variable class.
- As in multiple-level queue scheduler, as long as there are threads in the highest priority queue, then only those threads will be allocated the processor. If there is no thread in that queue, then the scheduler will service the threads in the second highest priority queue.
- In the second highest priority queue, threads are not in the ready state, the scheduler will service the third highest priority queue and so on.
- Windows XP is not a real time system and cannot guarantee that threads running at high priority will receive the processor before any fixed deadline.
- Kernel is object oriented. Kernel objects are intended to be fast. They run in supervisor mode in the trusted context.
- Kernel object cannot be manipulated directly by user mode programs, only through function calls.
- Kernel objects are of two types : ***Control object and dispatcher object.***
- **Control object** implement mechanism to control the hardware and kernel resources. Control objects are data structures that the kernel layer provides as abstractions to the executive layer for managing the CPU.
- **Dispatcher objects** are used to implement threads along with their scheduling and synchronization operations.
- In windows, objects may be either named or unnamed.

Software Interrupts

- Software interrupts are of two types : ***Asynchronous and deferred procedure call.***

1. Asynchronous Procedure Call (APC)

- APC execute in the context of a specific thread. APC is used to break into the execution of a specified thread and to cause a procedure to be called in a specified processor mode.
- The system uses APCs to complete various functions that must be done in a specific threads context. Each thread maintains two APC queues : One for kernel mode and another for user mode APC.
- Software interrupts that are generated by kernel mode components are **the kernel mode APC**. After allocating processor, thread must process all pending kernel mode APC.
- User mode thread can create a user APC and request to queue this APC to another thread. A thread executes user mode APCs when the thread enters an alertable wait state.
- User mode APC can also be used to deliver notification of I/O completion in user mode to the thread that initiated the I/O.

2. Deferred Procedure Call (DPC)

- DPCs are software interrupts. They execute at dispatch interrupt request levels (IRQL). DPC are used to postpone interrupt processing to avoid delaying hardware interrupts.
- DPC is also used to implements timers and inter-process communication.
- When device driver generates a DPC, the system places the DPC in a queue associated with a specific processor. The driver routine can request a processor to which the DPC should be queued. If processor is not specified, the processor queues the DPC to itself. The routine can also specify the DPC priority.

Object Manager

- Object manager is the executive components.
- In Windows XP, each object is defined by an object type. Object types are created by executive components. The object manager maintains the windows XP internal name space.
- Object types in windows XP include devices, files, threads, processes, pipes, semaphores etc. each windows XP object is an instance of an object type.
- The object manager is responsible for creating and deleting objects and maintaining information about each object type.

Virtual Memory Manager (VMM)

- Windows XP allocates a unique 4 GB virtual address space to each process. By default, a process can access only the first 2 GB of its virtual address space. The remaining 2 GB address space is used for kernel components and this is called as system space.
- Virtual memory manager uses a two level hierarchical addressing system. Windows XP divides virtual memory into fixed size pages which it stores either in page frames in main memory or files on disk.
- Virtual memory manager assigns each process one **page directory table.** The page directory table contains page directory entries.
- Each page directory entries points to a **page table.** Page tables contain page table entries. Each page table entries points to a page frame in main memory or a location on disk.
- VMM uses copy on write pages and employs lazy allocation. Windows XP allows applications to allocate large pages. Large page is a set of contiguous pages. Operating system consider this as one page. The system manages copy on write pages using prototype page tables.

Memory Allocation

- Memory allocation is in three stages :

1. Reserve
2. Commit
3. Access

a. A process must first reserve space in its virtual address space.

b. A process may not access space it has reserved until it commits the space. Process usually commits a page only when it is read to write to the page.

c. When a process is ready to use its committed memory, it accesses the committed virtual memory.

Process Manager

- Process consists of program code, an execution context, resources and one or more threads. Process manger provides services for creating, deleting and using processes and threads.
- Process and threads is the object. Windows processes are implemented as objects. Both process and thread objects have built in synchronization capabilities.

1.12 Two Marks Questions with Answers

Q.1 *Define operating systems.*

Ans. : An operating system is a program that manages the computer hardware.

Q.2 *What are the objective of operating systems ?*

Ans. : The objective of operating systems are efficient use, user convenience, ability to evolve.

Q.3 *Explain what is batch processing.*

Ans. : Here jobs with similar requirements are batched together and run through the computer as a group. Thus a batch operating system reads a stream of separate jobs, each with its own control cards that predefine what the job does, feed the batches one after another and send the output of each job to the appropriate destination.

Q.4 *What is spooling ?*

Ans. : The use of secondary memory as buffer storage to reduce processing delays when transferring data between peripheral equipment and the processors of a computer.

Q.5 *Define the degree of multiprogramming.*

Ans. : Degree of multiprogramming is the number of processes in the memory.

Q.6 *What is meant by "hard real systems and soft real systems"?*

Ans. : Hard real systems guarantee that critical tasks complete on time. In Soft real system a critical task get priority over other tasks and remains that priority until it completes.

Q.7 *What is the main advantage of multiprogramming ?*

Ans. : Multiprogramming makes efficient use of the CPU by overlapping the demands for the CPU and its I/O devices from various users. It attempts to increase CPU utilization by always having something for the CPU to execute.

Q.8 *What are the three main purposes of an operating system ?*

Ans. :

- To provide an environment for a computer user to execute programs on computer hardware in a convenient and efficient manner.
- To allocate the separate resources of the computer as needed to solve the problem given. The allocation process should be as fair and efficient as possible.
- As a control program it serves two major functions : 1) Supervision of the execution of user programs to prevent errors and improper use of the computer and 2) Management of the operation and control of I/O devices.

Q.9 *Define real time system.*

Ans. : Real time system is one that must react to inputs and responds to them quickly. A real time system has well defined, fixed time constraints.

Q.10 *What do you mean by multiprogramming ?*

Ans. : Multiprogramming increases CPU utilization by organizing jobs so that the CPU always has one to execute.

Q.11 *Write the difference between batch system and time sharing systems.*

Ans. :

- A batch system executes jobs, whereas a time-shared system has user programs, or tasks.
- Batch systems are inconvenient for users because users cannot interact with their jobs to fix problems. User interacts with system in time sharing system.

Q.12 *What is command interpreter system ?*

Ans. : Command-Interpreter system is a system program, which is the interface between the user and the operating system. Command-Interpreter system is known as the shell.

Q.13 *List any four services provided by operating system.* **MSBTE : Winter-18, Marks 2**

Ans. : Services provided by operating systems are program execution, I/O operations, File system manipulation, Communication, Error detection and resource allocation.

Q.14 *Define UNIX operating system.* **MSBTE : Winter-18, Marks 2**

Ans. : UNIX is a multi-user, multi-tasking operating system. Multiple users may have multiple tasks running simultaneously. UNIX is a machine independent operating system

Q.15 *What is booting process ?* **MSBTE : Winter-18, Marks 2**

Ans. : When the computer is switched on, it activates the memory-resident code which resides on the CPU board. The normal facilities of the operating system are not available at this stage and the computer must 'pull itself up by its own boot-straps' so to speak. This procedure therefore is often referred to as bootstrapping.

❑❑❑

UNIT - II

2 SERVICES AND COMPONENTS OF OPERATING SYSTEM

2.1 Different Services of Operating System

- Operating system provides different types of services to different users. It provides services to program like load of data into memory, allocating disk for storage, files or directory for open or read etc.
- Services will change according to the operating system. May be the two different operating system provides same type of services with different names. OS makes programmer job easy by providing different services.
- Following are the list of services provided by operating systems :
  1. Program execution
  2. Input-output operation
  3. Error detection
  4. File and directory operation
  5. Communication
  6. Graphical User Interface

1. **Program execution :** Before executing the program, it is loaded into the memory by operating system. Once program loads into memory, its start execution. Program finishes its execution with error or without error. It is up to the user for next operation.
2. **Input - output operation :** Program is combination of input and output statement. While executing the program, it requires I/O device. OS provides the I/O devices to the program.
3. **Error detection :** Error is related to the memory, CPU, I/O device and in the user program. Memory is full, stack overflow, file not found, directory not exist, printer is not ready, attempt to access illegal memory are the example of error detection.
4. **File and directory operation :** User wants to read and writes the file and directory. User wants to search the file/directory, rename file, and modify the file etc. user also create the file or directory. All these operation is performed by user by using help of operating system.
5. **Communication :** Communication may be inter-process communication and any other type of communication. Data transfer is one type of communication. Communication is in between two process of same machine or two process of different machine. Pipe, shared memory, socket and message passing are the different methods of communication.
6. **Graphical user interface :** User interacts with operating system by using user interface. All operating system provides user interface. Command line interface and batch interface are two types of user interface used in the operating system. In command line interface, user enters the text command for performing operation. Batch interface uses files. A file contains the command for various operations. When file executes, command output is displayed on the screen.

- All above services provided by operating system is only for single user. If suppose there are multiple user in the system, then operating system provides some additional services. These services are resource allocation, accounting, protection of data and security.

- Different types of resources are managed by the operating system. Multiple users require different resource for their execution. One type of resource may be required by two different users. So resource allocation is necessary.
- Accounting means keeping information of resources and users. Which types of resources are allocated to which user and whether particular user has permission for using this type of resources.

Board Questions

1. *Explain any six services provided operating system. Draw diagram showing services.* **MSBTE : Winter-15, Marks 4**
2. *List any four operating system services and describe in one/two sentences.* **MSBTE : Summer-16 Marks 4**
3. *List and state any four services provided by an operating system.* **MSBTE : Winter-16, Marks 4**
4. *State and describe services provided by an operating system.* **MSBTE : Summer-15, 17, Marks 6**
5. *List any four services provided by OS and explain any two them..* **MSBTE : Winter-17 Marks 4**
6. *Explain following two services of operating systems.*

i) File system manipulation

ii) Resource Allocation **MSBTE : Summer-18 Marks 4**

2.2 System Calls

- System calls provide the interface between a running program and the operating system. Any single CPU computer can execute only one instruction at a time. If a process is running a user program in user mode and needs a system service, such as reading a data from a file, it has to execute a trap instruction to transfer control to the operating system.
- Operating system provides services and system call provides interface to these services. System call is written in language C and C++ as routines. System calls are performed in a series of steps.
- System call is a technique by which a program executing in user mode can request the kernel's service.
- An application programmer interface is a function definition that specifies how to obtain a given service.
- Fig. 2.2.1 shows the working of system call.

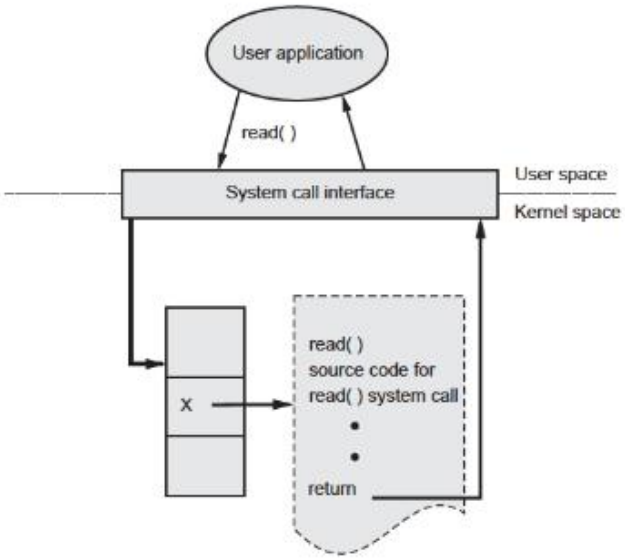


Fig. 2.2.1 Working of system call

- When application program calls the stub, trap instruction is executed and CPU switches to supervisor mode. Each system call contains its identification number.
- OS maintains the table of system call number. Operating system executes the system call using that number.
- When the function completes, it switches the processor to user mode and then returns control to the user process.
- A system call is an explicit request to the kernel mode via a software interrupt. When user mode process invokes a system call, the CPU switches to kernel mode and starts the execution of the kernel function.
- Making a system call is like making a special kind of procedure call. Only system call enters the kernel and procedure call does not enter into the kernel.
- Kernel implements many different types of system calls. The user mode process must pass a parameter called the system call number to identify the required system call. All system calls return an integer value. In the kernel, positive or 0 values denote a successful termination of the system call and negative values denote an error condition.

- An API does not necessarily correspond to a specific system calls.
  1. API could offer its services directly in user mode.
  2. Single API function could make several system calls.
  3. Several API functions could make the same system call.
- API supports a set of functions for application programmer. It includes passing parameter to each function and return values. API used by application programmer are as follows :
  1. Windows system : win32 API
  2. POSIX system : POSIX API
  3. Virtual machine : Java API
- User function uses trap instruction for using kernel services. Application programmer uses ordinary procedure call. Operating system provides a library of user functions with name corresponding to each actual system call. Each of these stub function contains a trap to the operating system function.
- Trap is also called system call interface.
- Three general methods are used to pass parameters to the operating system.
  a. Pass parameters in registers
  b. Registers pass starting addresses of blocks of parameters
  c. Parameters can be placed, or pushed, onto the stack by the program, and popped off the stack by the OS
- Fig. 2.2.2 shows passing of parameters as a table.

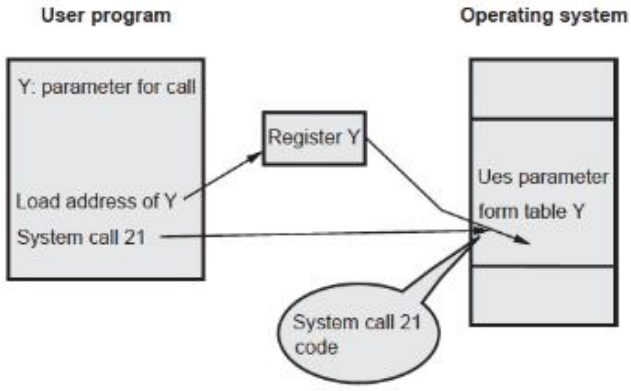


Fig. 2.2.2 Passing of parameters

2.2.1 Classification of System Call

- System call is divided into following types :
  1. File management
  2. Process management
  3. Interprocess communication
  4. I/O device management
  5. Information processing and maintenance

1. File management

- File management system calls are *create file, delete file, open file, close file, read file, write file, get and set file attribute.*
- User can create a file using create( ) system call. File name with attributes are required for creating and deleting a file through system call. After creating a file, user can performs various operations on the file.
- Read, write, reposition are the operation performed on the file. File is closed after finished using. Same type of operation is performed on directory.
- Every file has file attributes. File attributes includes name of file, type of file, accounting information etc. To perform any operation on the file, set file attribute and get file attribute executed to check the attributes.

2. Process management

- System calls for process management are create, terminate, load, execute, abort, set and get process attributes. Other system call for process management is wait for time, wait event, allocate and free memory.
- In some situation user want to terminate the currently running process abnormally then system call used. Other reasons for abnormal process termination are error message generated, memory dump, error trap.
- Operating system provides debugging facility to determine the problem of dump. Dump is written to secondary storage disk.
- Debugger is a one type of system program. It provides facility for finding and correcting bug.

3. Interprocess communication

- Pipe, socket, message passing and shared memory are used for interprocess communication. Send message, receive message, create and delete connection, attach remote device are the system calls used in interprocess communication.
- Shared memory : A process uses memory for communication. One process will create a memory portion which other processes can access. A shared segment can be attached multiple times by the same process. Shared memory is the fastest form of IPC because data does not need to be copied between processes.
- A *socket* is a bidirectional communication device that can be used to communicate with another process on the same machine or with a process running on other machine.
- Message passing : Two processes communicate with each other by passing messages. Message passing is direct and indirect communication. Indirect communication uses mailbox for sending receiving message from other process.

4. I/O device management

- System calls for device management are request ( ) device, release ( ) device, get and set device attributes, read, write etc.
- Process needs several resources in its life time. When process request for resources, request is granted if it is free otherwise request is rejected. Once request is granted, control is transfer to the process.

5. Information processing and maintenance

- System calls for this category is set time and date, get time and date, get system data, get system data, get and set process, file and device.
- Most of the operating system provides system call for set and get the time and date.

This type of system call is used for transferring information from user to operating system and vice versa.

Board Questions

1. *What is system call ? Explain open( ) system call and close( ) system call.* **MSBTE : Winter-15, Summer-16, Marks 4**
2. *Explain how parameter passing is done while implementing system calls.* **MSBTE : Summer-16, Marks 4**
3. *Describe the purpose of system calls ? State two system calls with its functions.* **MSBTE : Winter-16, Marks 4**
4. *State any four types of system calls provided by an operating system.* **MSBTE : Summer-17, Marks 4**
5. *What is system call ? With the help of diagram explain open( ) system call.* **MSBTE : Summer-17, Marks 4**
6. *What is system call? Enlist any four system calls related with process management.* **MSBTE : Summer-15, Winter-17, Marks 4**
7. *List types of system call and explain the system call - "Information Maintenance".* **MSBTE : Summer-18, Marks 4**
8. *Explain any four file related system calls.* **MSBTE : Winter-18, Marks 4**

2.3 OS Components

2.3.1 Process Management

- Process refers to a program in execution. The process abstraction is a fundamental operating system mechanism for management of concurrent program execution. The operating system responds by creating a process.
- A process needs certain resources, such as CPU time, memory, files and I/O devices. These resources are either given to the process when it is created or allocated to it while it is running.
- When the process terminates, the operating system will reclaim any reusable resources.
- The term process refers to an executing set of machine instructions. Program by itself is not a process. A program is a passive entity.
- The operating system is responsible for the following activities of the process management.

1. Creating and destroying the user and system processes.
2. Allocating hardware resources among the processes.
3. Controlling the progress of processes.
4. Providing mechanisms for process communications.
5. Also provides mechanisms for deadlock handling.

2.3.2 Main Memory Management

- The memory management modules of an operating system are concerned with the management of the primary (main memory) memory. Memory management is concerned with following functions :
  1. Keeping track of the status of each location of main memory. i.e. each memory location is either free or allocated.
  2. Determining allocation policy for memory.
  3. Allocation technique i.e. the specific location must be selected and allocation information updated.
  4. Deallocation technique and policy. After deallocation, status information must be updated.
- Memory management is primarily concerned with allocation of physical memory of finite capacity to requesting processes. The overall resource utilization and other performance criteria of a computer system are affected by performance of the memory management module. Many memory management schemes are available and the effectiveness of the different algorithms depends on the particular situation.

2.3.3 File Management

- Logically related data items on the secondary storage are usually organized into named collections called files. In short, file is a logical collection of information. Computer uses physical media for storing the different information.
- A file may contain a report, an executable program or a set of commands to the operating system. A file consists of a sequence of bits, bytes, lines or records whose meanings are defined by their creators. For storing the files, physical media (secondary storage device) is used.
- Physical media are of different types. These are magnetic disk, magnetic tape and optical disk. All the media has its own characteristics and physical organization. Each medium is controlled by a device.
- The operating system is responsible for the following in connection with file management.
  1. Creating and deleting of files.
  2. Mapping files onto secondary storage.
  3. Creating and deleting directories.
  4. Backing up files on stable storage media.
  5. Supporting primitives for manipulating files and directories.
  6. Transmission of file elements between main and secondary storage.
- The file management subsystem can be implemented as one or more layers of the operating system.

2.3.4 I/O System Management

- The module that keeps track of the status of devices is called the I/O traffic controller. Each I/O device has a device handler that resides in a separate process associated with that device.
- The I/O subsystem consists of
  1. A memory management component that includes buffering, caching and spooling.
  2. A general device driver interface.
  3. Drivers for specific hardware devices.

2.3.5 Secondary Storage Management

- A storage device is a mechanism by which the computer may store information in such a way that this information may be retrieved at a later time. Secondary storage device is used for storing all the data and programs. These programs and data access by computer system must be kept in main memory. Size of main memory is small to accommodate all data and programs. It also lost the data when power is lost. For this reason secondary storage device is used. Therefore the proper management of disk storage is of central importance to a computer system.

- The operating system is responsible for the following activities in connection with the disk management.

  1. Free space management 2. Storage allocation

  3. Disk scheduling
- The entire speed and performance of a computer may hinge on the speed of the disk subsystem.

Board Questions

1. *Describe any four activities of process management and memory management.*
   MSBTE : Winter-15, Marks 4
2. *List and explain components of operating system.*
   MSBTE : Winter-16, Marks 6
3. *What is process management? State any four functions of process management.*
   MSBTE : Winter-18, Marks 4
4. *What is process management ? State four functions to be performed by OS for process management.* **MSBTE : Summer-16, Marks 4**

2.4 Use of Operating System Tools

1. Task Scheduler

- The Task Scheduler is a tool included with Windows that allows predefined actions to be automatically executed whenever a certain set of conditions is met.
- For example, you can schedule a task to run a backup script every night, or send you an e-mail whenever a certain system event occurs
- Windows can perform a task at a scheduled time and scheduled date. Not only one time does it work, but windows can do a particular task scheduled every day at a particular time or at particular interval.
- In windows 7, windows vista or windows 8, one may find the task scheduler in an updated version. Thus it makes the windows user equipped with he best scheduler and that also without the support of any other software.

2. Performance monitor

- It is the option to actually monitor how the CPU is performing. One can go there from the control panel or another easy way to reach there is by pressing sticky keys on the key boards.
- One may find there in the small window a button called Resource Monitor. By clicking there, a new larger window will open where there will be four continuously running meters.
- The meters are of the CPU, Memory, Network and Disk. Thus all the records are placed in front one's eye and thus making the performance monitoring easy.
- Performance Monitor acts as both a real time and log-based performance monitoring tool for operating systems.
- Like Task Manager, Performance Monitor measures performance by making system calls to retrieve system counters, but Performance Monitor makes these calls via a performance library that also supports logging of the counters.
- Unlike Task Manager, Performance Monitor provides an interface to monitor any selection of a huge set of system counters on a graph in real time, rather than just the limited set Task Manager uses.
- Counters include things like percentage of processor time, thread count, page fault rate, memory size, and elapsed time for processes. Similarly, there are counters that provide state for threads, the processor, the system, network interfaces, memory, physical disks, and many others.

3. Security policy

- OS security protects systems and data from threats, viruses, worms, malware, ransom-ware, backdoor intrusions, and more.
- Security policies cover all preventative measures and techniques to ensure the safeguarding of an OS, the network it connects to, and the data which can be stolen, edited or deleted.

4. Device Management

- Device management generally performs the following :

  1. Installing device and component-level drivers and related software

2. Configuring a device so it performs as expected using the bundled operating system, business/workflow software and/or with other hardware devices.
3. Implementing security measures and processes

2.5 Two Marks Questions with Answers

Q.1 *What are the five major categories of system calls ?*

Ans. : Five main categories of system calls are : File management, IPC, process management, I/O devices management, information management.

Q.2 *List general methods used to pass parameters in system call.*

Ans. : Three general methods are used to pass parameters between a running program and the operating system.

a. Pass parameters in registers.
b. Store the parameters in a table in memory, and the table address is passed as a parameter in a register.
c. Push (store) the parameters onto the stack by the program, and pop off the stack by operating system.

Q.3 *Why APIs need to be used rather than system calls ?*

Ans. : An application programmer designing a program using an API can expect his program to compile and run on any system that supports the same API. Actual system calls can often be more detailed and difficult to work with the API available to an application programmer.

Q.4 *List any four services provided by operating system.* **MSBTE : Winter-18**

Ans. : Services provided by operating systems are program execution, I/O operations, File system manipulation, Communication, Error detection and resource allocation.

Q.5 *What is system call?* **MSBTE : Winter-18**

Ans. : System calls provide the interface between a process and the operating system.

❑❑❑

Notes

UNIT - III

3 PROCESS MANAGEMENT

3.1 Process

- Process term is used in MULTICS system in the 1960. Process is an asynchronous activity.
- Process is an active entity that requires a set of resources, including a processor, program counter, registers to perform its function. Multiple processes may be associated with one program.
- Process means a program in execution. Process execution must progress in sequential order. It also called task. Task is a single instance of an executable program. Fig. 3.1.1 shows memory layout for a process.

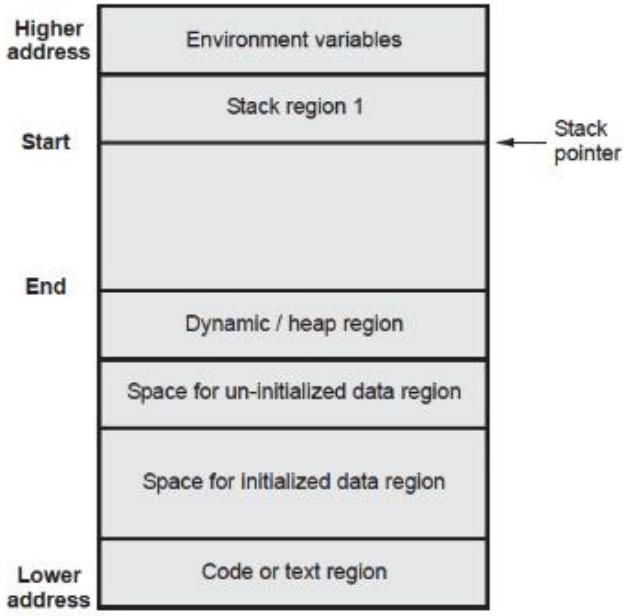


Fig. 3.1.1 Memory layout for a process

- Each process has its own address space. Address space is divided into two regions :

  1. Text region 2. Data region 3. Stack region

- **Text region :** It stores the code that that the processor executes.
- **Data region :** It stores variables and dynamically allocated memory that the process uses during execution.
- **Stack region :** It stores instructions and local variables for active procedure calls.
- Process is a dynamic entity that executes a program on a particular set of data using resources allocated by the operating system.
- A process can run to completion only when all requested hardware and software resources have been allocated to the process.
- Program is a passive entity. Program becomes process when executable file loaded into memory. A process may be independent of other processes in the system.
- After booting operating system, it creates foreground processes and background processes.
- Foreground processes interact with user and background processes is used by system. Background processes are related e-mail, web pages, news and printing.
- A process is used as a fundamental unit for resource allocation in operating system. A process normally has its own private memory area in which it runs.
- Text region contains executable instructions. It is be placed below the heap or stack.
- An initialized data region contains the static and global variables that are initialized by the user.

- Un-initialized data region contains all static and global variables that are initialized by kernel to zero.
- Heap is used for dynamic memory allocation and stack contains program counter.

3.1.1 Process States

- Each process has an execution state which indicates what process is currently doing. The process descriptor is the basic data structure used to represent the specific state for each process.
- Fig. 3.1.2 shows a process state diagram.
- A state diagram is composed of a set of states and transitions between states.
- The process states are new, ready, running, waiting and terminated.
  1. **New :** Initially process will be in new state. OS creates new process by using fork( ) system call.
  2. **Ready :** The process is competing for the CPU. Process reaches to the head of the list (queue).
  3. **Running :** The process that is currently being executed. OS allocates all the hardware and software resources to the process for execution.
  4. **Waiting :** A process is waiting until some event occurs such as the completion of an input-output operation.
  5. **Terminated :** A process is completes its operations and releases it all resources.

3.1.2 Process Control Block

- Operating system keeps an internal data structure to describe each process it manages.
- When OS creates process, it creates this process descriptor. In some operating system, it calls Process Control Block (PCB).
- Fig. 3.1.3 shows process control block.
- Process control block will change according to the operating system. PCB is also called **task control block.**

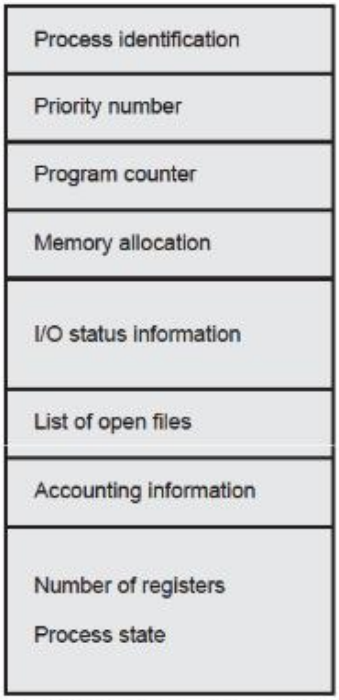


Fig. 3.1.3 Process control block

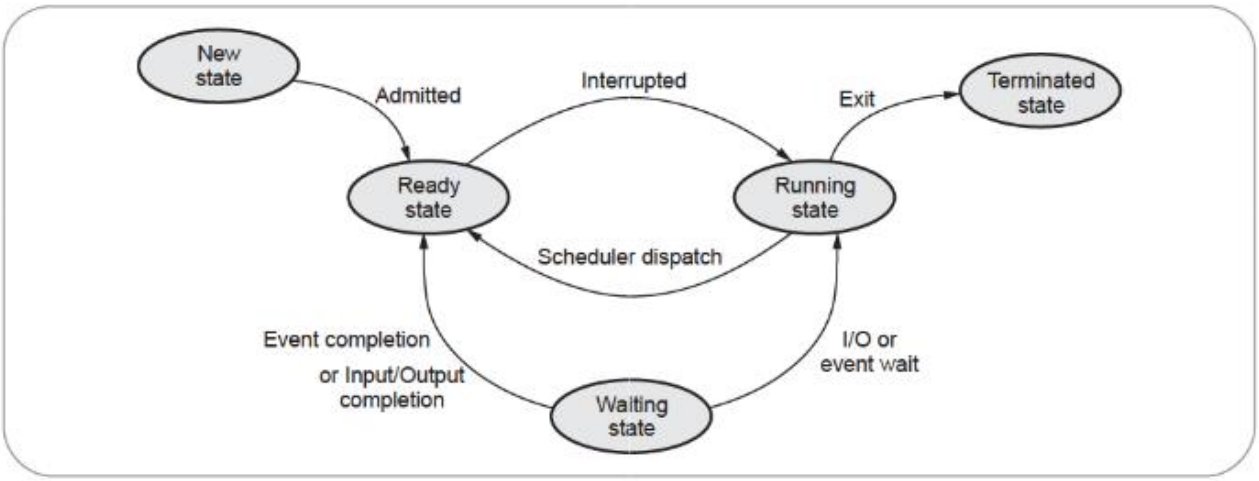


Fig. 3.1.2 Process state diagram

- The PCB is identified by an integer Process ID (PID). When a process is running, its hardware state is inside the CPU.
- When the OS stops running a process, it saves the register's values in the PCB.
- When a process is created by operating system, it allocates a PCB for it. OS initializes PCB and puts PCB on the correct queue. Following information is stored in process control block.
  1. **Process identification** : Each process is uniquely identified by the user's identification and a pointer connecting it to its descriptor.
  2. **Priority number** : Operating system allocates the priority number to each process. According to the priority number it allocates the resources.
  3. **Program counter** : The PC indicates the address of the next instruction to be executed for this current process.
  4. **Memory allocation** : It contains the value of the base registers, limit registers and the page tables depending on the memory system used by the operating system.
  5. **I/O status information** : It maintains information about the open files, list of I/O devices allocated to the process etc.
  6. **List of open files** : Process uses number of files for operation. Operating system keeps track of all opened file by this process.
  7. **Process state** : Process may be in any one of the state : new, ready, running, and waiting, terminate.
- When process changes the state, the operating system must update information in the process's process control block. Process control block maintain other information which is not included in PCB block diagram. This information includes CPU scheduling, file management, input-output management information.
- Fig. 3.1.4 shows process table with PCB.
- When process is created, hardware registers and flags values are set by loader or linker.
- Operating system maintains pointers to each process's PCB in a per user process table or system wide process table. This information is used to access PCB quickly.

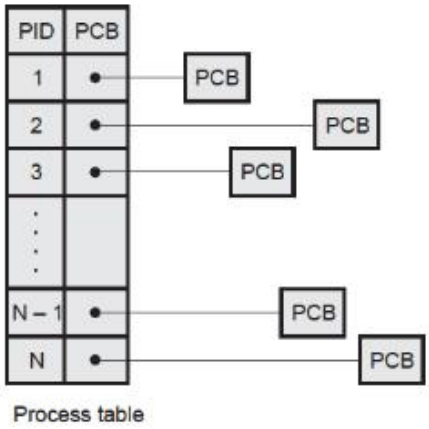


Fig. 3.1.4 Process table with PCB

3.1.3 Difference between Process and Program

Sr. No.	Process	Program
1.	Process is active entity.	Program is passive entity.
2.	Process is a sequence of instruction executions.	Program contains the instructions.
3.	Process exists in a limited span of time.	A program exists at single place and continues to exist.
4.	Process is a dynamic entity.	Program is a static entity.

Board Questions

1. *Define process. Explain process state in detail with the help of state diagram.* **MSBTE : Summer-15, Marks 4**
2. *Draw and explain process control block in detail.* **MSBTE : Summer-15, Winter-18, Marks 4**
3. *Draw and explain process state diagram.* **MSBTE : Winter-15, Marks 4**
4. *What is process ? Explain the different process states with diagram.* **MSBTE : Winter-15, Marks 6**
5. *With neat diagram describe use of Process Control Block (PCB).* **MSBTE : Summer-16,17, Winter-16,17, Marks 6**

6. *Draw the process state diagram and describe each state in one/two sentences.*
MSBTE : Summer-16, Winter-18, Marks 4
7. *Define process. Describe process creation and termination.* **MSBTE : Winter-16, Marks 8**
8. *State and explain different process state.*
MSBTE : Winter-17, Marks 4
9. *List and draw a neat labelled diagram of process state.* **MSBTE : Summer-18, Marks 4**
10. *Explain working of CPU switch from process to process with diagram.*
MSBTE : Summer-18, Marks 6

3.2 Process Scheduling

- CPU utilization is maximizing by using multiprogramming concept. Processor is not idle, it is executing a process. Processor scheduler selects one process for execution from the ready queue.

Scheduling Queue

- Scheduling queue is queue of processes or input-output devices. When the user process enters into the system, they put into the job queue. Job queue consists of all processes of the system.
- Operating system maintains different types of queues for different purposes. The queue are ready queue, device queue etc. Fig. 3.2.1 shows ready queue and device queue.
- **Ready queue :** The processes which are ready and waiting to execute are kept in the ready queue. Ready queue is stored in main memory. Linked list is used for representing ready queue. Pointer field of PCB is used for this.

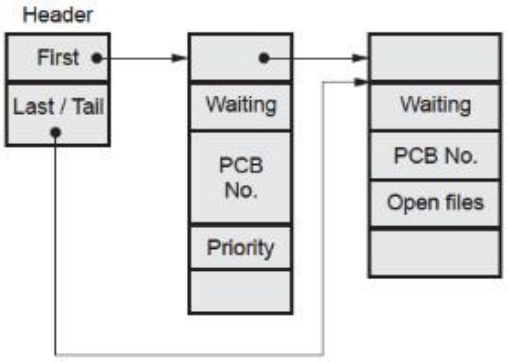


Fig. 3.2.1 Ready and device queue

- **Device queue :** Device queue is a queue for which multiple processes are waiting for a particular I/O device. Each device has maintained its own device queue.
- Process scheduling is represented by queueing diagram. Newly created process is kept into the ready queue and waits for processor. When CPU select the process for executing, following things happens :
  1. Process may require I/O device to perform operation. So it is put into the I/O queue.
  2. Process may create child process and wait for child process termination.
  3. Because of interrupt, process preempted from CPU and put into the ready queue.

3.2.1 Schedulers

- Schedulers are used to handles process scheduling. It is one type of system software and selects the jobs from the system and decide which process to run. Schedulers are of three types -
  1. Long term scheduler
  2. Short term scheduler 3. Medium term scheduler
- Fig. 3.2.2 shows queueing diagram for process scheduling. (See Fig. 3.2.2 on next page)

Long term scheduler

- Long term scheduler is also called job scheduler. It determines which process are admitted to the system for processing. Processes are selected from the queue and loads into the main memory for execution.
- Long term scheduling controls the degree of multiprogramming in multitasking systems. It provides a balanced mix of jobs, such as I/O bound and CPU bound.
- Long term scheduling is used in real time operating system. Time sharing operating system has no long term scheduler.

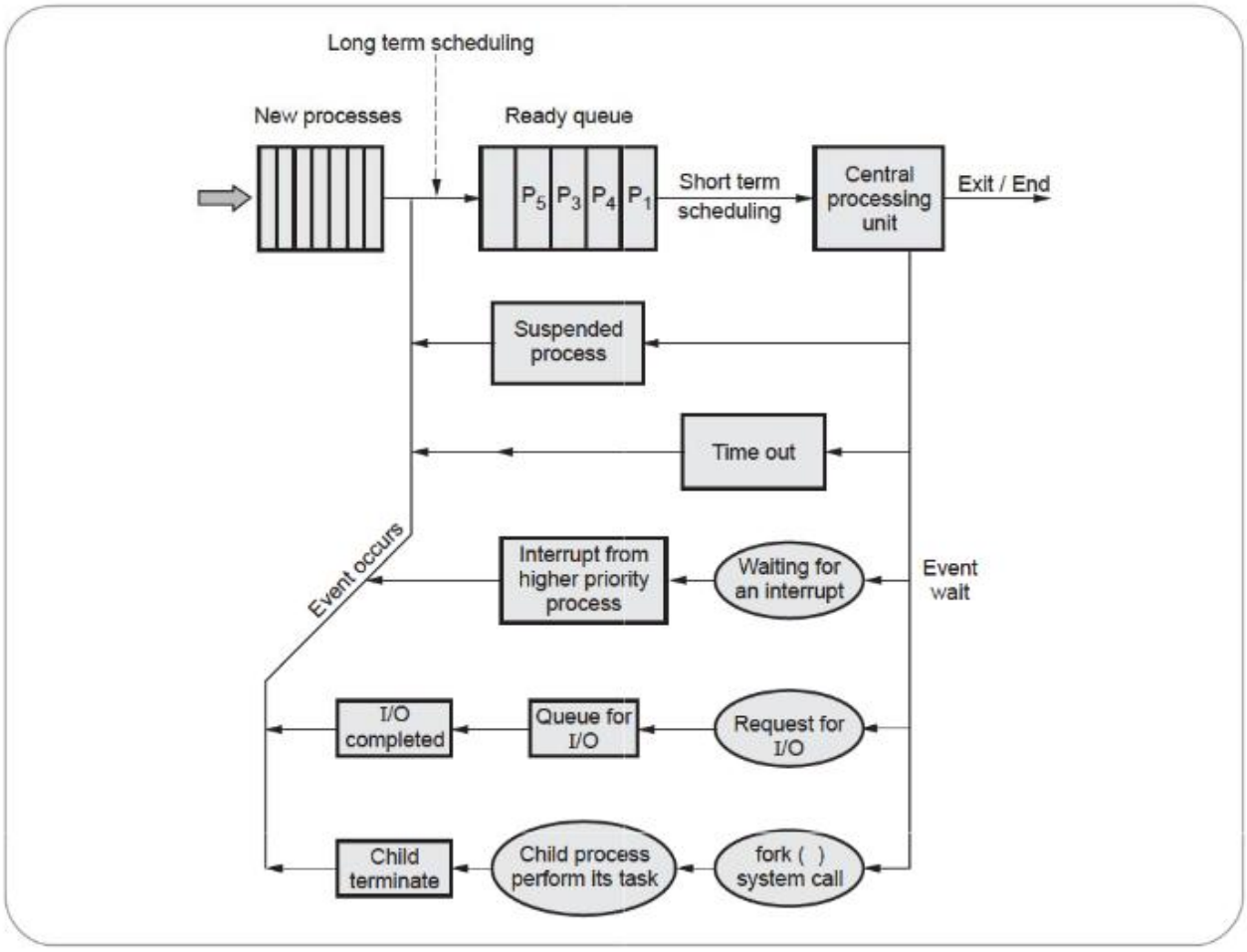


Fig. 3.2.2 Process scheduling queueing diagram

Medium term scheduler

- Medium term scheduler is part of swapping function. Sometimes it removes the process from memory. It also reduces the degree of multiprogramming.
- If process makes an I/O request and it is in memory then operating system takes this process into suspended states. Once the process becomes suspended, it cannot make any progress towards completion.
- In this situation, the process is removed from memory and makes free space for other process.
- The suspended process is stored in the secondary storage device i.e. hard disk. This process is called swapping.

Short term scheduler

- Short term scheduler is also called CPU scheduler. It selects the process from queue which are ready to execute and allocate the CPU for execution.
- Short term scheduler is faster that long term scheduler. This scheduler makes scheduling decisions much more frequently than the long-term or mid-term schedulers.
- A scheduling decision will at a minimum have to be made after every time slice, and these are very short.
- It is also known as dispatcher.

3.2.2 Difference between Long Term, Short Term and Medium Term Scheduler

Sr. No.	Long term	Short term	Medium term
1.	It is job scheduler.	It is CPU scheduler.	It is swapping.
2.	Speed is less than short term scheduler.	Speed is very fast.	Speed is in between both.
3.	It controls the degree of multi-programming.	Less control over degree of multi-programming.	Reduce the degree of multi-programming.
4.	Absent or minimal in time sharing system.	Minimal in time sharing system.	Time sharing system use medium term scheduler.
5.	It select processes from pool and load them into memory for execution.	It select from among the processes that are ready to execute.	Process can be reintroduced into memory and its execution can be continued.
6.	Process state is (New to Ready).	Process state is (Ready to Running)	-
7.	Select a good process, mix of I/O bound and CPU bound.	Select a new process for a CPU quite frequently.	-

3.2.3 Context Switch

- A context switch is the switching of the CPU from one process or thread to another. A context is the contents of a CPU's registers and program counter at any point in time.
- Switching the CPU to another process requires performing a state save of the current process and a state restore of a different process. This task is known as **a context switch**.
- A context switch can mean a register context switch, a task context switch, a thread context switch or a process context switch.
- A register is a small amount of very fast memory inside of a CPU that is used to speed the execution of computer programs by providing quick access to commonly used values.
- A program counter is a specialized register that indicates the position of the CPU in its instruction sequence and which holds either the address of the instruction being executed or the address of the next instruction to be executed, depending on the specific system.
- Context switching can be described in more detail as the Kernel performing the following activities with regard to processes (including threads) on the CPU :
  1. Suspending the progression of one process and storing the CPU's state (i.e., the context) for that process somewhere in memory.
  2. Retrieving the context of the next process from memory and restoring it in the CPU's registers and
  3. Returning to the location indicated by the program counter in order to resume the process.
- Context switches can occur only in Kernel mode (system mode). Kernel mode is a privileged mode of the CPU in which only the Kernel runs and which provides access to all memory locations and all other system resources.
- Other programs, including applications, initially operate in user mode, but they can run portions of the Kernel code via system calls. Software context switching can be used on all CPUs and can be used to save and reload only the state that needs to be changed.
- To use the hardware context switch you need to tell the CPU where to save the existing CPU state and where to load the new CPU state from. The CPU state is always stored in a special data structure called a TSS (Task State Segment).
- Context switch times are highly dependent on hardware support. Context switching represents a substantial cost to the system in terms of CPU time and it can be the most costly operation on an operating system.
- There are three situations where a context switch needs to occur. They are multitasking, interrupt handling, user and Kernel mode switching.

Board Questions

1. *Explain context switch with suitable example.*
   MSBTE : Summer-15, Marks 4
2. *State and describe types of scheduler.*
   MSBTE : Summer-15, Marks 4
3. *Explain Interprocess communication models with diagram.* **MSBTE : Summer-15, Marks 8**
4. *State and describe types of schedules. Describe how each of them schedule the job.*
   MSBTE : Winter-15, Marks 4
5. *Explain the concept of context switching.*
   MSBTE : Winter-15, Marks 4
6. *Differentiate between long term schedular and short term schedular on basis of*
   i) Selection of job
   ii) Frequency of execution
   iii) Speed iv) Accessing which part of system.
   MSBTE : Summer-16, Marks 4
7. *Describe how context switch is executed by operating system.* **MSBTE : Summer-16, Marks 4**
8. *Differentiate between short term, medium term and long term scheduling.*
   MSBTE : Winter-16, Marks 4
9. *Describe the following :*
   i) Schedulers ii) Context switch
   MSBTE : Summer-17, Marks 4
10. *Differentiate between short term and long term scheduler.* **MSBTE : Winter-17, Marks 4**
11. *Differentiate between long term scheduling and medium term scheduling.*
    MSBTE : Summer-18. Marks 4
12. *How context switching is done ?*
    MSBTE : Winter-18, Marks 4

3.3 Inter-Process Communication

- Exchange of data between two or more separate, independent processes/threads is possible using IPC. Operating systems provide facilities/resources for Inter-Process Communications (IPC), such as message queues, semaphores, and shared memory.
- A complex programming environment often uses multiple cooperating processes to perform related operations. These processes must communicate with each other and share resources and information. The Kernel must provide mechanisms that make this possible. These mechanisms are collectively referred to as **interprocess communication.**
- Distributed computing systems make use of these facilities/resources to provide Application Programming Interface (API) which allows IPC to be programmed at a higher level of abstraction. (e.g., send and receive).
- Five types of inter-process communication are as follows :
  1. Shared memory permits processes to communicate by simply reading and writing to a specified memory location.
  2. Mapped memory is similar to shared memory, except that it is associated with a file in the file system.
  3. Pipes permit sequential communication from one process to a related process.
  4. FIFOs are similar to pipes, except that unrelated processes can communicate because the pipe is given a name in the file system.
  5. Sockets support communication between unrelated processes even on different computers.

Name	Description	Scope	Use
File	• Data is written to and read from a typical UNIX file. • Any number of processes can interoperate.	Local	Sharing large data sets.
Pipe	• Data is transferred between two processes using dedicated file descriptors. • Communication occurs only between a parent and child process.	Local	Simple data sharing, such as producer and consumer.

Named pipe	• Data is exchanged between processes via dedicated file descriptors. • Communication can occur between any two peer processes on the same host.	Local	Producer and consumer, or command-and-control, as demonstrated with MySQL server and its command-line query utility.
Signal	• An interrupt alerts the application to a specific condition.	Local	Cannot transfer data in a signal, so mostly useful for process management.
Shared memory	• Information is shared by reading and writing from a common segment of memory.	Local	Cooperative work of any kind, especially if security is required.
Socket	• After special setup, data is transferred using common input/output operations.	Local or remote	Network services such as FTP, ssh and the Apache web server.

Table 3.3.1 Interprocess communication in UNIX

- **Purposes of IPC**

1. **Data transfer :** One process may wish to send data to another process.
2. **Sharing data :** Multiple processes may wish to operate on shared data, such that if a process modifies the data, that change will be immediately visible to other processes sharing it.

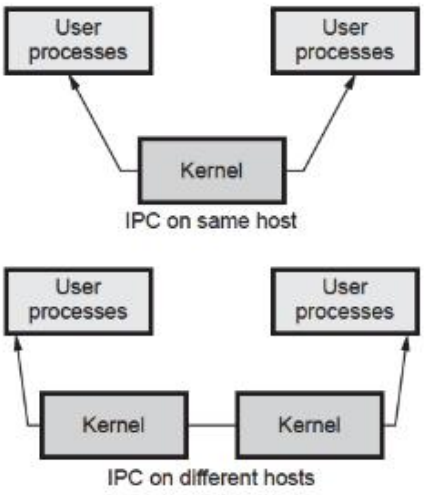


Fig. 3.3.1 IPC on different hosts

3. **Event modification :** A process may wish to notify another process or set of processes that some event has occurred.
4. **Resource sharing :** The Kernel provides default semantics for resource allocation; they are not suitable for all application.
5. **Process control :** A process such as a debugger may wish to assume complete control over the execution of another process.

- IPC has two forms : IPC on same host and IPC on different hosts

IPC is used for 2 functions :

1. **Synchronization :** Used to coordinate access to resources among processes and also to coordinate the execution of these processes. They are record locking, semaphores, mutexes and condition variables.
2. **Message passing :** Used when processes wish to exchange information. Message passing takes several forms such as : Pipes, FIFOs, message queues and shared memory.

3.3.1 Shared Memory

- A region of memory that is shared by co-operating processes is established. Processes can then exchange information by reading and writing data to the shared region.
- Shared memory allows maximum speed and convenience of communication, as it can be done at memory speeds when within a computer. Shared memory is faster than message passing, as message-passing systems are typically implemented using system calls and thus require the more time-consuming task of Kernel intervention.
- In contrast, in shared-memory systems, system calls are required only to establish shared-memory regions. Once shared memory is established, all accesses are treated as routine memory accesses, and no assistance from the Kernel is required.
- Fig. 3.3.2 shows client/server with shared memory.

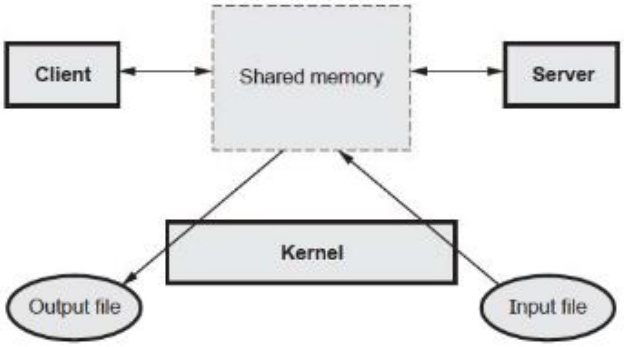


Fig. 3.3.2 Client / server with shared memory

Advantages

1. Good for sharing large amount of data. 2. Very fast.

Limitations

1. No synchronization provided - applications must create their own.
2. Alternative to shared memory is *mmap* system call, which maps file into the address space of the caller.

3.3.2 Message Passing System

- Message passing system requires the synchronization and communication between the two processes. Message passing used as a method of communication in microkernels. Message passing systems come in many forms. Messages sent by a process can be either fixed or variable size. The actual function of message passing is normally provided in the form of a pair of primitives.
  a) Send (destination_name, message)
  b) Receive (source_name, message).
- Send primitive is used for sending a message to destination. Process sends information in the form of a message to another process designated by a destination. A process receives information by executing the receive primitive, which indicates the source of the sending process and the message.
- Design characteristics of message system for IPC.
  1. Synchronization between the process
  2. Addressing
  3. Format of the message
  4. Queueing discipline.

Issues in IPC by Message Passing

- Message is a block of information.
- A message is a meaningful formatted block of information sent by the sender process to the receiver process.
- The message block consists of a fixed length header followed by a variable size collection of typed data objects.
- The header block of a message may have the following elements :
  1. **Address :** A set of characters that uniquely identify both the sender and receiver.
  2. **Sequence number :** It is the message identifier to identify duplicate and lost messages in case of system failures.
  3. **Structural information :** It has two parts. The type part that specifies whether the data to be sent to the receiver is included within the message or the message only contains a pointer to the data. The second part specifies length of the variable-size message.
- Fig. 3.3.3 shows the typical message format.

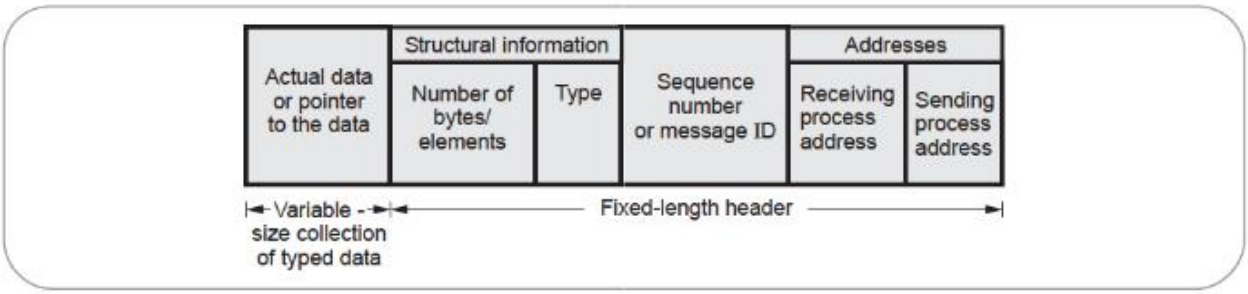


Fig. 3.3.3 Message structure

- Some important issues to be considered for the design of an IPC protocol based message passing system :
  i. The sender's identity
  ii. The receiver's identity
  iii. Number of receivers
  iv. Guaranteed acceptance of sent messages by the receiver
  v. Acknowledgment by the sender
  vi. Handling system crashes or link failures
  vii. Handling of buffers
  viii. Order of delivery of messages

3.3.3 Features of Message Passing

1. **Simplicity :** Message passing system should be simple and easy to use. It should be possible to communicate with old and new applications.
2. **Uniform semantics :** Message passing is used for two types of IPC.
   a. **Local communication :** Communicating processes are on the same node.
   b. **Remote communication :** Communicating processes are on the different nodes.
3. **Efficiency :** IPC become so expansive if message passing system is not effective. Users try avoiding to IPC for their applications. Message passing system will become more efficient if we try to avoid more message exchanges during communication process. For examples :
   a. Avoiding the costs of establishing and terminating connection.
   b. Minimizing the costs of maintaining the connections.
   c. Piggybacking of acknowledgement.
4. **Reliability :** Distributed systems are prone to different catastrophic events such as node crashes or physical link failures. Loss of message because of communication link fails. To handle the loss messages, we required acknowledgement and retransmission policy. Duplicate message is one of the major problems. This happens because of timeouts or events of failures.
5. **Correctness :** Correctness is a feature related to IPC protocols for group communication. Issues related to correctness are as follows :
   i. **Atomicity :** Every message sent to a group of receivers will be delivered to either all of them or none of them.
   ii. **Ordered delivery :** Messages arrive to all receivers in an order acceptable to the application.
   iii. **Survivability :** Messages will be correctly delivered despite partial failures of processes, machines, or communication links.
6. **Security :** Message passing system must provide a secure end to end communication.
7. **Portability :** Message passing system should itself be portable.

Board Questions

1. *Explain interprocess communication model with diagram.* **MSBTE : Summer-15, Winter-15, Marks 4**
2. *What is inter process communication ? Describe any one technique of it.* **MSBTE : Summer-16, Marks 4**
3. *Draw and explain inter-process communication model.* **MSBTE : Winter-16, 17, 18, Marks 6**
4. *With neat diagram, explain message passing system. Also describe the following :*
   i) Naming ii) Synchronization iii) Buffering. **MSBTE : Summer-17, Marks 8**
5. *Explain the working of inter-process communication considering.*
   1) Shared memory 2) Message passing **MSBTE : Summer-18, Marks 6**
6. *Define synchronization explain :*
   i) Blocking ii) Non blocking in message passing **MSBTE : Summer-18, Marks 4**

3.4 Threads

- Thread is a dispatchable unit of work. It consists of thread ID, program counter, stack and register set. Thread is also called a Light Weight Process (LWP). Because they take fewer resources then a process. A thread is easy to create and destroy.
- It shares many attributes of a process. Threads are scheduled on a processor. Each thread can execute a set of instructions independent of other threads and processes. Fig. 3.4.1 shows a thread.

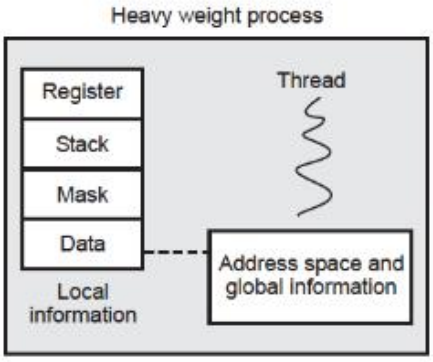


Fig. 3.4.1 Thread

- Every program has at least one thread. Programs without multithreading executes sequentially. That is, after executing one instruction the next instruction in sequence is executed.
- Thread is a basic processing unit to which an operating system allocates processor time and more than one thread can be executing code inside a process.
- When user double click on an icon by using mouse, the operating system creates a process and that process has one thread that runs the icon's code.
- Each thread belongs to exactly one process and no thread can exist outside a process. Each thread represents a separate flow of control.
- Normally, an operating system will have a separate thread for each different activity. The OS will have a separate thread for each process and that thread will perform OS activities on behalf of the process. In this condition, each user process is backed by a kernel thread. When process issues a system call to read a file, the process's thread will take over, find out which disk accesses to generate, and issue the low level instructions required to start the transfer. It then suspends until the disk finishes reading in the data.
- When process starts up a remote TCP connection, its thread handles the low-level details of sending out network packets.
- In Remote Procedure Call (RPC), servers are multithreaded. Server receives the messages using a separate thread.
- Different operating system uses threads in different ways.
  1. Free memory is managed by kernel thread in LINUX OS.
  2. Solaries uses a kernel thread for interrupt handling.

3.4.1 Thread Benefits

1. Context switching time is minimized.
2. Thread support for efficient communication.
3. Resource sharing is possible using threads.
4. A thread provides concurrency within a process.
5. It is more economical to create and context switch threads.

3.4.2 Difference between Thread and Process

Sr. No.	Thread	Process
1.	Thread is also called lightweight process.	Process is also called heavyweight process.
2.	Operating system is not required for thread switching.	Operating system interface is required for process switching.
3.	One thread can read, write or even completely clean another threads stack.	Each process operates independently of the other process.
4.	All threads can share same set of open files and child processes.	In multiple processing, each process executes the same code but has its own memory and file resources.

5.	If one thread is blocked and waiting then second thread in the same task can run.	If one server process is blocked then other server process cannot execute until the first process is unblocked.
6.	Uses fewer resources.	Uses more resources.

3.4.3 Thread Lifecycle

- Fig. 3.4.2 shows a thread lifecycle. A thread has one of the following states.
  1. **New :** Thread is just created.
  2. **Ready :** Thread's start( ) method invoked and now it is executing. OS put thread into Ready queue.
  3. **Running :** Highest priority ready thread enters the running state. Thread is assigned a processor and now is running.
  4. **Blocked :** This is the state when a thread is waiting for a lock to access an object.
  5. **Waiting :** Here thread is waiting indefinitely for another thread to perform an action.
  6. **Sleeping :** Thread sleep for a specified time of period. When sleeping time expires, it enters to ready state. CPU is not used by sleeping thread.
  7. **Dead :** When thread completes task or operation.

3.4.4 User Level Thread

- User level thread uses user space for thread scheduling. These threads are **transparent** to the operating system. User level threads are created by runtime libraries that cannot execute privileged instructions.
- User-level threads have low overhead but it can achieve high performance in computation. User-level threads are managed entirely by the run-time system.
- User-level threads are small and faster. A thread is simply represented by a PC, registers, stack and small thread control block. Fig. 3.4.3 shows user level thread.

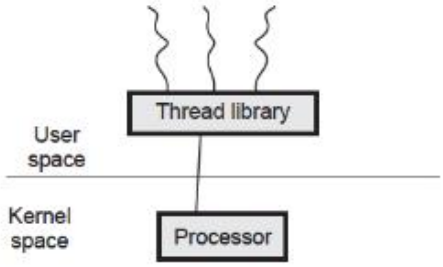


Fig. 3.4.3 User level thread

- The code for creation and destroying thread, message passing and data transfer, thread scheduling is included into thread library. Kernel is unaware of user level thread.

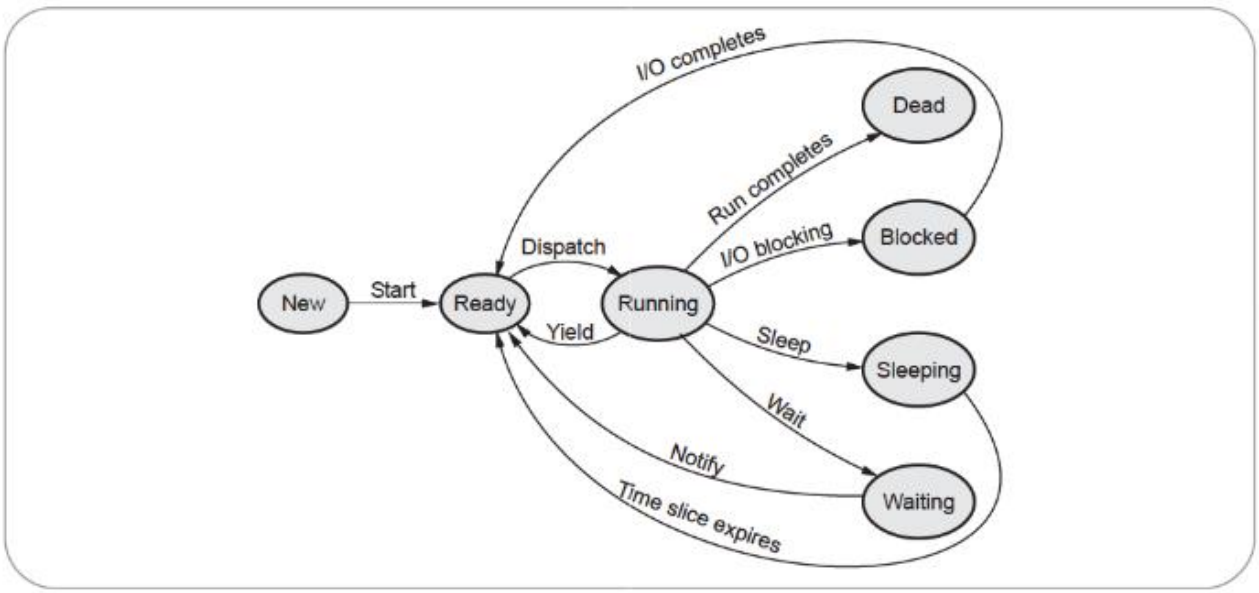


Fig. 3.4.2 Thread lifecycle

- User level threads do not invoke the Kernel for scheduling decision.
- User level thread are also called **many to one mapping** thread because the operating system maps all threads in a multithreaded process to a single execution context. The operating system considers as each multithreaded processes as a single execution unit.
- Example : POSIX Pthreads and Mach C-threads.

Advantages :

1. Kernel mode privilege does not require for thread switching.
2. These threads are fast to create and manage.
3. User level thread works even if the OS does not support threads.
4. User level threads are more portable.
5. Threading library controls flow of thread.

Disadvantages :

1. If thread blocks, the Kernel may block the all threads.
2. Not suitable for multiprocessor system.
3. User level threads also do not support system wide scheduling priority.

3.4.5 Kernel Level Thread

- In Kernel level thread, thread management is done by Kernel. Operating systems support the Kernel level thread.
- Since Kernel managing threads, Kernel can schedule another thread if a given thread blocks rather than blocking the entire processes. Fig. 3.4.4 shows Kernel level thread.

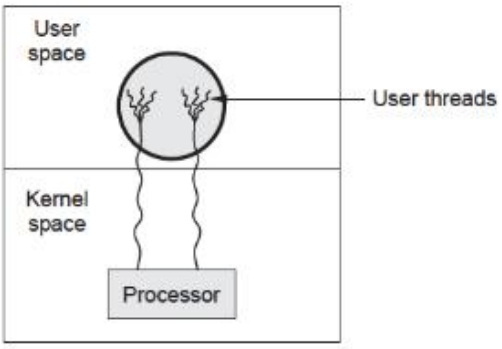


Fig. 3.4.4 Kernel level thread

- Kernel level thread support one to one thread mapping. This mapping requires each user thread with kernel thread. Operating system performs this mapping.
- Threads are constructed and controlled by system calls. The system knows the state of each thread.
- Thread management code is not included in the application code. It is only API to the Kernel thread. Windows operating system uses this facility.
- Any application can be programmed to be multithreaded. All of the threads within an application are supported within a single process.
- Kernel performs scheduling on a thread basis. The Kernel support for scheduling and management, thread creation only in Kernel space.
- Kernel level threads are slower than user level threads.
- Example : Windows 95/98/NT, Sun Solaris and Digital UNIX.

Advantages :

1. Each thread can be treated separately.
2. A thread blocking in the Kernel does not block all other threads in the same process.
3. Kernel routines itself as multithreaded.

Disadvantages :

1. Slower than the user level thread.
2. There will be overhead and increased in Kernel complexity.

3.4.6 Difference between User Level and Kernel Level Thread

Sr. No.	User level threads	Kernel level threads
1.	User level threads are faster to create and manage.	Kernel level threads are slower to create and manage.
2.	Implemented by a thread library at the user level.	Operating system support directly to Kernel threads.
3.	User level thread can run on any operating system.	Kernel level threads are specific to the operating system.

4.	Support provided at the user level called user level thread.	Support may be provided by Kernel is called Kernel level threads.
5.	Multithread application cannot take advantage of multiprocessing.	Kernel routines themselves can be multithreaded.
6.	Example : POSIX Pthreads and Mach C-threads	Example : Windows 95/98/NT, Sun Solaris and Digital UNIX
7.	User level threads are also called many to one mapping thread.	Kernel level thread support one to one thread mapping.

3.4.7 Multithreading Models

- Operating system uses user level thread and kernel level thread. User level threads are managed without kernel support. Operating system support and manage the kernel level threads.
- Modern operating system support kernel level threads. Kernel performs multiple simultaneous tasks in operating system. In most of the application, user threads are mapped with kernel level threads.
- Different methods of mapping is used in operating system are as follows :
  1. One to one
  2. Many to one
  3. Many to many

1. One to one

- In this model, one user level thread maps with one kernel level thread. If there are three user threads then system creates three separate kernel thread. This model provides more concurrency because of separate thread. Fig. 3.4.5 shows one to one mapping.
- In this method, the operating system allocates data structure that represents kernel threads. Here multiple threads are run in parallel on multiprocessor system.
- Number of threads in the system increases, the amount of memory required is also increases.
- Windows 95/XP and Linux operating system uses this one to one thread mapping.

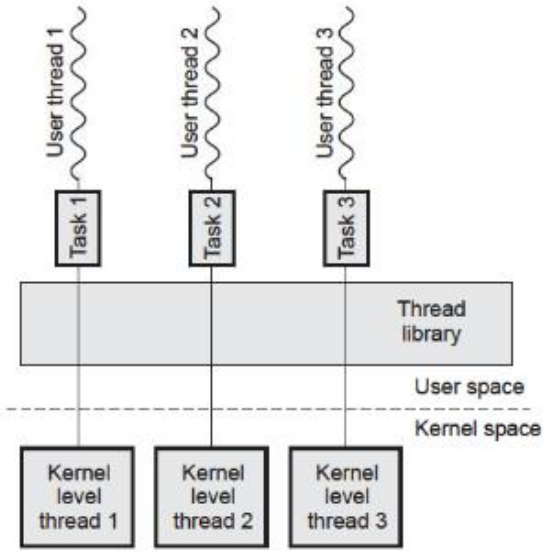


Fig. 3.4.5 One to one multithreading model

- Only overhead in this method is creation of kernel level thread for user level thread. Because of this overhead, system performance is slowdown.

2. Many to one

- Many to one mapping means many user thread maps with one kernel thread. Fig. 3.4.6 shows many to one mapping.

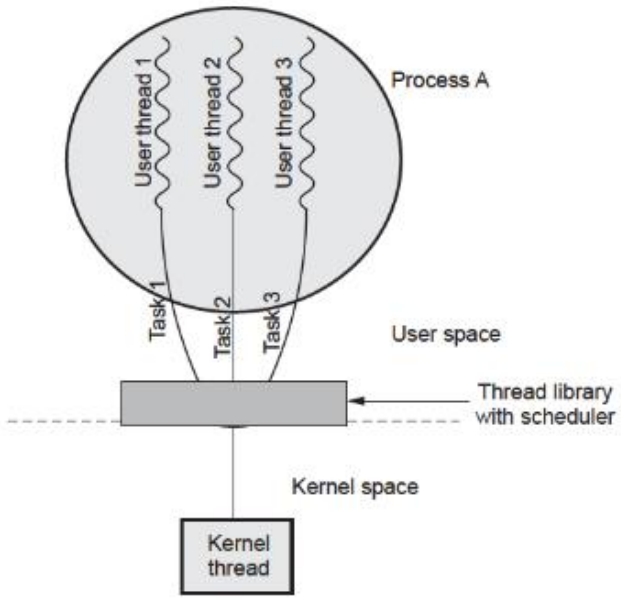


Fig. 3.4.6 Many to one multithreading model

- Operating system blocks the entire multi-threaded process when a single thread blocks because the entire multi-threaded process is a single thread of

control. So when operating system receive any a blocking I/O request, it blocks the entire process.

- Thread library handle thread management in user space. The many-to-one model does not allow individual processes to be split across multiple CPUs.
- This type of relationship provides an effective context-switching environment, easily implementable even on simple kernels with no thread support.
- Green threads for Solaris and GNU portable threads implement the many-to-one model in the past, but few systems continue to do so today.
- **Advantage :** System performance is improved by customizing the thread library scheduling algorithm.

3. Many to many

- In the many to many mapping, many user-level threads maps with equal number of kernel threads. Fig. 3.4.7 shows many to many thread mapping.
- Many to many mapping is also called M-to-N thread mapping. Thread pooling is used to implement this method.
- Users can create required number of thread and there is no limitation for user. Again here blocking Kernel system calls do not block the entire process.
- This model support for splitting processes across multiple processors.
- **Limitations :** Operating system design becomes complicated.

Ex. 3.4.1 *Provide two programming examples in which multithreading does not provide better performance than a single-threaded solution.*

Sol. : a. A web server that services each request in a separate thread.

b. A parallelized application such as matrix multiplication where different parts of the matrix may be worked on in parallel.

c. An interactive GUI program such as a debugger where a thread is used to monitor user input, another thread represents the running application, and a third thread monitors performance.

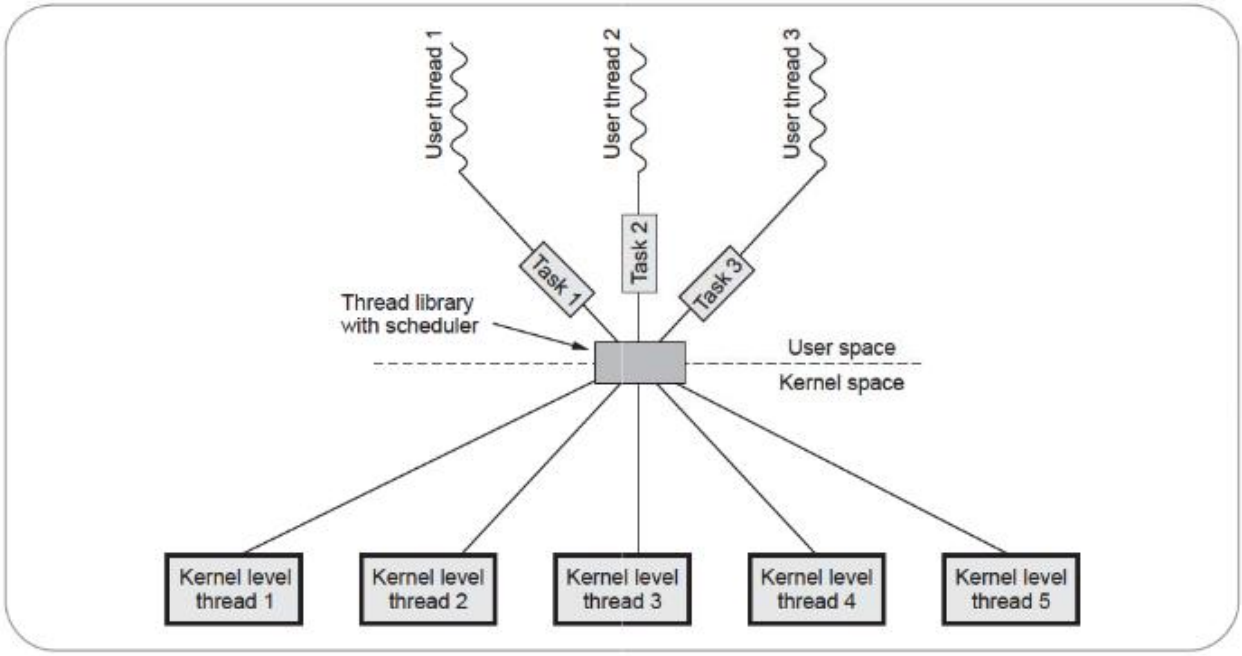


Fig. 3.4.7 Many to many multithreading model

Board Questions

1. *Describe many to one and one to one multithreading model with diagram.*
   MSBTE : Summer-15, Marks 6
2. *What is thread ? Explain users and Kernel threads.*
   MSBTE : Winter-15, Marks 4
3. *Describe many to one and one to many multithreading model with diagram. Explain advantages of each.*
   MSBTE : Winter-15, Marks 8
4. *Define thread. State any three benefits of thread.*
   MSBTE : Winter-16, Marks 4
5. *Write benefits of using threads.*
   MSBTE : Summer-17, Marks 4
6. *With neat diagram, explain many to one and many to many multithreading model with its advantages and disadvantages.*
   MSBTE : Summer-17, Marks 6
7. *List and explain various type of multi-threading models with diagram.*
   MSBTE : Winter-16,17,18, Marks 8

3.5 Execute Process Commands

3.5.1 ps Command

- The ps command is used to provide information about the currently running processes, including their process identification numbers (PIDs).
- This command stands for 'Process Status '.
- A process, also referred to as a task, is an executing instance of a program. Every process is assigned a unique PID by the system.
- The basic syntax of ps command :

  ps [options]
- When ps is used without any options, it sends to standard output, which is the display monitor by default, four items of information for at least two processes currently on the system: the shell and ps
- You can also check the process status of a single process, use the syntax : ps PID

3.5.2 wait

- The kernel keeps a small amount of information for every terminating process, so that the information is available when the parent of the terminating process calls wait or waitpid.
- When a process terminates, either normally or abnormally, the kernel notifies the parent by sending the SIGCHLD signal to the parent.
- Because the termination of a child is an asynchronous event it can happen at any time while the parent is running this signal is the asynchronous notification from the kernel to the parent.
- The parent can choose to ignore this signal or it can provide a function that is called when the signal occurs : a signal handler. The default action for this signal is to be ignored. For now, we need to be aware that a process that calls wait or waitpid can :
  1. Block, if all of its children are still running.
  2. Return immediately with the termination status of a child, if a child has terminated and is waiting for its termination status to be fetched
  3. Return immediately with an error, if it doesn't have any child processes.
- If the process is calling wait because it received the SIGCHLD signal, we expect wait to return immediately. But if we call it at any random point in time, it can block.

```
#include <sys/wait.h>
pid_t wait(int *statloc);
pid_t waitpid(pid_t pid, int *statloc, int options);
```

- A process can synchronize its execution with the termination of a child process by executing the wait system call.
- The syntax for the system call is

  pid = wait (state_addr);

  where pid is the process ID of the zombie child and stat addr is the address in user space of an integer that will contain the exit status code of the child.

3.5.3 Sleep

- When a process must temporarily suspend its execution, it does so of its own free will. An interrupt handler cannot go to sleep because if it could, the interrupted process would be put to sleep by default.
- Processes go to sleep because they are awaiting the occurrence of some event, such as waiting for I/O completion from a peripheral device, waiting for a

process to exit, waiting for system resources to become available and so on.

- Processes are said to sleep on an event, meaning that they are in the sleep state until the event occurs, at which time they wakeup and enter the state "ready to run".
- When a process wakeup, it follows the state transition from the "sleep" state to the "ready to run" state, where it is eligible for later scheduling, it does not execute immediately.
- Sleeping processes do not consume CPU resources.
- Process executing in Kernel mode must sometimes lock a data structure in case it goes to sleep at a later stage. The Kernel implements locks in the following manner ;

```
while (condition is true)
Sleep (event : the condition becomes false) ;
Set condition true;
```

- It unlocks the lock and awakness all processes a sleep on the lock in the following manner :

```
Set condition false ;
Wakeup (event : the condition is false) ;
```

3.5.4 exit

- Processes on a UNIX system terminate by executing the exit system call. The syntax for the call is

```
exit (status);
```

where the value of status is returned to the parent process for its examination.

- The exit () functions terminate the calling process.
- The command causes the shell or program to terminate. If performed within an interactive command shell, the user is logged out of their current session and/or user's current console or terminal connection is disconnected.
- Typically an optional exit code can be specified, which is typically a simple integer value that is then returned to the parent process.
- exit () command closes all of a process's file descriptors, de-allocates its code, data and stack and then terminates the process.

3.5.5 Kill

- The **kill** function sends a signal to a process or a group of processes. The ***raise*** function allows a process to send a signal to itself.
- The function prototype for kill and raise API :

```
#include <signal.h>
int kill(pid_t pid, int signo);
int raise(int signo);
Both return: 0 if OK, 1 on error
```

- There are four different conditions for the pid argument to kill.
  1. **pid > 0** : The signal is sent to the process whose process ID is pid.
  2. **pid = = 0** : The signal is sent to all processes whose process group ID equals the process group ID of the sender and for which the sender has permission to send the signal.
  3. **pid < 0** : The signal is sent to all processes whose process group IDequals the absolute value of pid and for which the sender has permission to send the signal.
  4. **pid = = 1** : The signal is sent to all processes on the system for which the sender has permission to send the signal. As before, the set of processes excludes certain system processes.

3.6 Two Marks Questions with Answers

Q.1 *Define context switching.*

Ans. : Switching the CPU to another process requires saving the state of the old process and loading the saved state for the new process. This task is known as context switch.

Q.2 *Define a thread. State the major advantage of threads.*

Ans. : A thread is a flow of execution through the process's code with its own program counter, system registers and stack.

Advantages :

1. Minimize context switching time.
2. Efficient communication.

Q.3 *What is PCB ? Specify the information maintained in it.*

Ans. : Each process is represented in the operating system by a process control block. PCB

contains information like process state, program counter, CPU register, accounting information etc.

Q.4 *Differentiate a thread from a process.*

Ans. : Thread is called light weight process and process is heavy weight process. Thread switching does not need to call OS and cause an interrupt to the Kernel. Process switching needs interface with OS.

Q.5 *What are the benefits of multithreads ?*

Ans. : Benefits of multithreading is responsiveness, resource sharing, economy and utilization of multiprocessor architecture.

Q.6 *Why a thread is called as light weight process ?*

Ans. : Thread is light weight taking lesser resources than a process. It is called light weight process to emphasize the fact that a thread is like a process but is more efficient and uses fewer resources and they also share the address space.

Q.7 *What are the reasons for terminating execution of child process ?*

Ans. : Parent may terminate execution of children processes via abort system call for a variety of reasons, such as :

1. Child has exceeded allocated resources.
2. Task assigned to child is no longer required.
3. Parent is exiting and the operating system does not allow a child to continue if its parent terminates.

Q.8 *What is ready queue ?*

Ans. : The processes that are residing in main memory and are ready and waiting to execute are kept on a list called the ready queue.

Q.9 ***What do you mean by short term scheduler ?***

Ans. : Short term scheduler, also known as a dispatcher executes most frequently, and makes the finest-grained decision of which process should execute next. This scheduler is invoked whenever an event occurs.

Q.10 *Differentiate single threaded and multi-threaded processes.*

Ans. : Single-threading is the processing of one command at a time. When one thread is paused, the system waits until this thread is resumed. In Multithreaded processes, threads can be distributed over a series of processors to scale. When one thread is paused due to some reason, other threads run as normal.

Q.11 *Define process.* **MSBTE : Winter-18**

Ans. : A process is simply a program in execution. Process is an active entity which requires set of resources.

Q.12 *List different multithreading model.* **MSBTE : Winter-18**

Ans. : Multithreading model are many to one, one to one and many to many.

❑❑❑

UNIT - IV

4 CPU SCHEDULING AND ALGORITHMS

4.1 Scheduling Types

- In a multiprogramming environment, usually more programs to be executed than could possibly be run time at one time. In CPU scheduling it switches from one process to another process. CPU resource management is commonly known as scheduling.
- Objective of the multiprogramming is to increases the CPU utilization. CPU scheduling is one kind of fundamental operating system functions.
- User program contains combination of CPU burst cycle and I/O burst cycles. Process starts with CPU burst cycle then I/O burst cycle again CPU burst cycle again I/O burst cycle. In this way the process completes its executions. Process will terminates after final CPU burst cycle completions.

  Program execution sequence : *CPU burst cycle → I/O burst cycle → CPU burst cycle → I/O burst cycle → CPU burst cycle → Program terminates.*
- A CPU bound process tends to use the processor time that the system allocates for it. An I/O bound process tends to use the processor only briefly before generating an I/O request. The CPU bound processes spend most of their time using the processor.

CPU Scheduler

- CPU scheduler is also called as short scheduler. System programmers use parameters like program size, resources required for program and any other special devices are needed to determine scheduling policies.
- Scheduling mechanism is the part of the process manager that handles the removal of the running process from the CPU and the selection of another process on the basis of a particular strategy.
- Scheduler is responsible for multiplexing processes on the CPU.

4.1.1 Preemptive and Non-preemptive Scheduling

- The scheduling policy determines when it is time for process to be removed from the CPU and which ready process should be allocated the CPU next. The scheduling mechanism is composed of several different parts, depending on exactly how it is implemented in any particular operating system.
- CPU scheduling is divided into two types : Preemptive scheduling and non-preemptive scheduling.
- **Preemptive scheduling :** A scheduling method that interrupts the processing of a process and transfers the CPU to another process is called a preemptive CPU scheduling. The process switches from running state to the ready state and waiting state to the ready state.
- **Non-preemptive scheduling :** Non-preemptive operation usually proceeds towards completion uninterrupted. Once the system has assigned a processor to a process, the system cannot remove that processor from the process. The process switches from running state to the waiting state and termination of process.
- Preemptive scheduling increases the cost and it has higher overheads. This scheduling method is useful only in the high priority processes require rapid response.
- Non-preemptive scheduling is simple to implement. It requires some hardware platform. This method is

attractive because of its simplicity. Windows 3.1 and Apple Macintosh operating system uses this scheduling method.

4.1.2 Difference between Preemptive and Non-preemptive Scheduling

Sr. No.	Preemptive scheduling	Non-preemptive scheduling
1.	Preemptive scheduling allows a process to be interrupted in the midst of its execution, taking the CPU away and allocating it to another process.	Non-preemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.
2.	Preemptive scheduling incurs a cost associated with access shared data.	Non-preemptive scheduling does not increase the cost.
3.	It also affects the design of the operating system Kernel.	It does not affects the design of OS Kernel.
4.	Preemptive scheduling is more complex.	Simple, but very inefficient.
5.	Example : Round robin method.	Example : First come first serve method.

4.1.3 CPU Scheduling Criteria

- Depending on the system, CPU scheduling criteria will change.

1. **Throughput :** CPU scheduling should attempt to service the maximum number of processes per unit time. The higher is the number, the more work is done by the system.
2. **Waiting time :** The average period of time a process spends waiting. Process is normally in the ready queue in waiting time.
3. **Turnaround time :** Turnaround time start from process submission to completion of process.

   Turnaround time = Burst time + Waiting time
4. **Response time :** It is the time from the submission of a request until the first response is produced.
5. **CPU utilization :** It is average function of time during which the processor is busy.
6. **Fairness :** Avoid the process from the starvation. All the processes must be given equal opportunity to execute.
7. **Priority :** If the operating system assigns priorities to processes, the scheduling mechanism should favor the higher priority processes.
8. **Predictability :** A given process always should run in about the same amount of time under similar system loads.

- Depending upon the nature of operations the scheduling policy may differ. The CPU utilization and throughput are the system centered parameters. Fairness is affect by user and system.

4.1.4 Dispatcher

- Using dispatcher, CPU selects the process from the short term scheduler. Functions of the dispatcher is as follows:
  1. Switching context 2. Switching to user mode
  3. Jump to proper location in the user program for restarting that program.
- Dispatcher is a small program that switches the processor from one process to another.
- **Dispatch Latency :** It is time taken by dispatcher to stop running one process and start other process running is called as dispatch latency.

Board Questions

1. *Define preemptive and non-preemptive scheduling with suitable example.* **MSBTE : Winter-15, Marks 4**
2. *Define the following terms :*
   i) Preemptive scheduling
   ii) Nonpreemptive scheduling.
   MSBTE : Summer-16,15, Winter-17,18, Marks 4
3. *State and explain criteria in CPU scheduling.*
   MSBTE : Summer-16, Winter-17, Marks 4
4. *Describe CPU and I/O burst cycle with the help of diagram.* **MSBTE : Winter-16,18, Marks 4**
5. *State and explain criteria used in differentiating CPU scheduling.* **MSBTE : Winter-16, Marks 4**
6. *Explain the pre-emptive and non-pre-emptive type of scheduling. State when pre-emptive and non-pre-emptive type scheduling is used.*
   MSBTE : Winter-16, Marks 8

7. *State and explain four scheduling criteria.*
MSBTE : Summer-17, Marks 4

8. *List four process scheduling criteria and explain the term Turnaround in detail.*
MSBTE : Summer-18, Marks 4

4.2 Types of Scheduling Algorithms

- CPU scheduling algorithms are as follows :
  1. First In First Out (FIFO) scheduling
  2. Shortest job first scheduling
  3. Priority scheduling
  4. Round robin scheduling

4.2.1 First Come First Serve Scheduling

- This method of scheduling means that the first process to request the processor gets it until it finished execution. With this algorithm, processes are assigned the CPU in the order they request it.
- Normally there is a single queue of ready processes. When the first process enters the system from the outside, it is started immediately and allowed to run as long as it wants it. At the same time, other process enters into the system; they are put onto the end of the queue.
- New process enters the tail of the queue and the schedule selects the process from the head of the queue.
- When new process enters into the system, its process control block is linked to the end of the ready queue and it is removed from the front of the queue.
- When a process waits or blocks, it is removed from the queue and it queues up again in FCFS queue when it gets ready.
- FCFS is non-preemptive CPU scheduling algorithm. It is also called First In First Out method (FIFO).
- FCFS is simple to implement because it uses a FIFO queue. This algorithm is fine for most of the batch operating systems.
- FCFS is not useful in scheduling interactive processes because it cannot guarantee short response time. This algorithm performs much better for long processes than short ones.
- Turnaround time is unpredictable with the FCFS algorithm. Average waiting time of the FCFS is often quite long.
- Real life analogy is **buying tickets**.

Convey effect

- To reduce I/O device utilization, all I/O bound processes will be waiting excessively long for processor bound ones. This is called as **convey effect**.
- If one process monopolizes the system, the extent of its overall effect on system performance depends on the scheduling policy and whether the process is processor bound or I/O bound.

Advantages

1. Simple to implement
2. Fair

Disadvantages

1. Waiting time depends on arrival order
2. Convoy effect : short process stuck waiting for long process
3. Also called head of the line blocking

4.2.2 Shortest Job First Scheduling

- Shortest job first scheduling algorithm is also known as Shortest Job Next (SJN) scheduling algorithm. It handles the process based on length of their CPU cycle time. It reduces average waiting time over FIFO algorithm.
- SJF is a non-preemptive CPU scheduling algorithm.
- It does not work in interactive system because users do not estimate in advance the CPU time required to run their processes.
- SJF scheduling algorithm is used frequently in long term scheduling.
- When a process request a CPU, it must inform the system for how long it wants to use the CPU.
- When CPU become available, the system allocates into processes with the least expected execution time.
- To break ties, it follows the FCFS algorithm.
- Preemptive version of SJF is called as Shortest Remaining Time Next (SRTN)

- Preemptive SJF algorithm will preempt the currently exceutig process, whereas a non preemptive SJF algorithm will allow the currently running process to finish its CPU burst.
- SIF selects processes for service in a manner ensuring the next one will complete and leave the system as soon as possible.
- The SJF scheduling algorithm is optimal only when all of the processes are available at the same time and the CPU estimates are available.
- SJF scheduling algorithm may be **preemptive or non-preemptive.**

4.2.3 Priority Scheduling

- Priority CPU scheduling algorithm is preemptive and non-preemptive algorithm. It is one of the most common scheduling algorithm in batch system. Priority can be assigned by a system admin using characteristics of the process.
- In this scheduling algorithm, CPU select higher priority process first. If the priority of two process is same then FCFS scheduling algorithm is applied for solving the problem.
- The priority of a process determines how quickly its request for a CPU will be granted if other processes make competing requests.
- Each process is assigned a priority number for the purpose of CPU scheduling.
- The priority number is normally a non negative integer number.
- Non preemptive priority scheduling algorithm will simply put the new process at the head of the ready queue.
- Preemptive priority scheduling algorithm will preempt the CPU if the priority of the newly arrived process is higher than the priority of the currently running process.

Problem with Priority Scheduling

- Waiting time is more for lower priority process even if there required CPU burst time is less. Priority scheduling algorithm faces the starvation problem. Starvation problem can be solved by using **aging techniques.**
- In aging, the priority of the process will increase which is waiting for long time in the ready queue.

4.2.4 Round Robin Scheduling

- Round robin is a preemptive scheduling algorithm. It is used in interactive system.
- Here process are given a limited amount of time of processor time called a time slice or time quantum. If a process does not complete before its quantum expires, the system preempts it and gives the processor to the next waiting process. The system then places the preempted process at the back of the ready queue.
- Processes are placed in the ready queue using a FIFO scheme. With the RR algorithm, the principle design issue is the length of the time quantum to be used. For short time slice, processes will move through the system relatively quickly. It increases the processing overheads.

4.2.5 Comparison between FCFS and RR

Sr. No.	FCFS	Round robin
1.	FCFS decision made is non-preemptive.	RR decision made is preemptive.
2.	It has minimum overhead.	It has low overhead.
3.	Response time may be high.	Provides good response time for short processes.
4.	It is troublesome for time sharing system.	It is mainly designed for time sharing system.
5.	The workload is simply processed in the order of arrival.	It is similar like FCFS but uses time quantum.
6.	No starvation in FCFS.	No starvation in RR.

4.2.6 Comparison of CPU Scheduling Algorithm

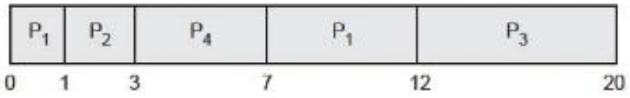
Algorithm	Policy type	Used in	Advantages	Disadvantages
FCFS	Non-preemptive	Batch	1. Easy to implement. 2. Minimum overhead.	1. Unpredictable turn around time. 2. Average waiting is more.
RR	Preemptive	Interactive	1. Provides fair CPU allocation. 2. Provides reasonable response times to interactive users.	1. Requires selection of good time slice.
Priority	Non-preemptive	Batch	1. Ensures fast completion of important jobs.	1. Indefinite postponement of some jobs. 2. Faces starvation problem.
SJF	Non-preemptive	Batch	1. Minimizes average waiting time. 2. SJF algorithm is optimal.	1. Indefinite postponement of some jobs. 2. Cannot be implemented at the level of short term scheduling. 3. Difficulty is knowing the length of the next CPU request.
Multilevel queues	Preemptive or Non-preemptive	Batch/Interactive	1. Flexible. 2. Gives fair treatment to CPUbound jobs.	1. Overhead incurred by monitoring of queues.

4.2.7 Shortest Remaining Time Next

- Preemptive SJF is called *shortest remaining time first*. In this algorithm, the scheduler selects the process with the smallest estimated run time to completion.
- This scheduler gives minimum wait times in theory but in certain situation due to preemption overhead, shortest process first might performance better.
- An advantageous of this method is that, the short processes are handled very quickly. The system requires very little overhead since it only makes a decision when a process completes or a new process is added. When a new process is admitted the SRTN algorithm only needs to compare the currently executing process with the new process, ignoring all other processes currently waiting to execute.
- Consider the four processes with their arrival and burst time.

Process	Burst Time	Arrival time
P1	6	0
P2	2	1
P3	8	2
P4	4	3

- The execution of the process is shown by the Gantt chart :



- Now we calculate waiting time and turnaround time for all processes.

Process	Waiting Time	Turnaround time
P1	(0 - 0) + (7 –1) = 6	6 + 6 = 12
P2	1 – 1 = 0	2 + 0 = 2
P3	12 – 2 = 10	8 + 10 = 18
P4	3 – 3 = 0	4 + 0 = 4

Average waiting time = $\frac{6+0+10+0}{4} = \frac{16}{4} = 4$

Average waiting time = $\frac{12+2+18+4}{4} = \frac{36}{4} = 9$

Advantages :

1. Very short processes get very good service.
2. The penalty ratios are small; this algorithm works extremely well in most cases
3. This algorithm provably gives the highest throughput of all scheduling algorithms if the estimates are exactly correct.

4.2.8 Multilevel Queue Scheduling

- Multilevel queue is not separate scheduling algorithm but it works in conjunction with several CPU scheduling algorithms. The processes can be grouped according to common characteristics.
- One type of multilevel queue is based on the priority based system with different queues for each priority level.
- System consists of two types of processes or jobs : Interactive and batch. Interactive jobs are shorter and batch jobs are longer. Fig. 4.2.1 shows multilevel queue scheduling.
- To keep minimum response time for user, interactive jobs are scheduled before the batch jobs. So ready queue is partitioned into two separate queue: interactive (foreground) and batch (background). Each queue uses their own scheduling algorithm.

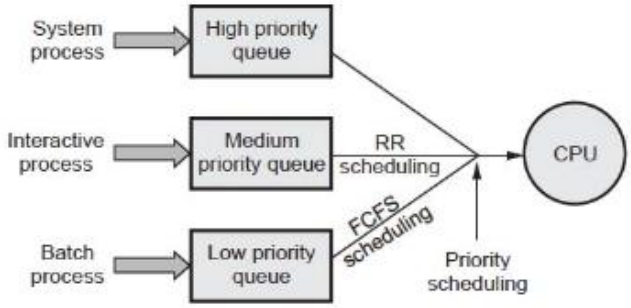


Fig. 4.2.1 Multilevel queue scheduling

- Once the jobs assigns to the queue, it will remain in that queue. Jobs never change the queue. Here we assume that FCFS is applied to one queue and round robin method is applied to another queue.
- The batch jobs are put in one queue called the background queue while the interactive jobs are put in a foreground queue. Batch jobs are scheduled by first come first serve method and interactive jobs by round robin method.
- The scheduler now has to decide which queue to run. The methods are as follows :
  1. Higher priority queues can be processed until they are empty before the lower priority queues are executed.
  2. Each queue can be given a certain amount of the CPU. Maybe, the interactive queue could be assigned 75 % of the CPU, with the batch queue being given 25 %.
- It should also be noted that there can be many other queues. Many systems will have a *system queue*. This queue will contain processes that are important to keep the system running. For this reason these processes normally have the highest priority.
- It should be also need to specify which queue a process will be put to when it arrives to the system and/or when it starts a new CPU burst.

Advantages :

1. It can be preemptive or non-preemptive.
2. Short CPU bound processes are executed first.
3. Simple to implement.

Disadvantage :

1. Queues require monitoring, which is a costly activity.

4.2.9 Multilevel Feedback Queue Scheduling

- Multilevel Feedback Queue (MLFQ) allows moving the processes between the queues. MLFQ has a number of distinct queues; each assigned a different priority level. At any given time, a process that is ready to run is on a single queue. MLFQ uses priorities to decide which process should run at a given time. A process with higher priority is selected for execution.
- Consider processes with different CPU burst characteristics. If a process uses too much of the CPU it will be moved to a lower priority queue. This will leave I/O bound and interactive processes in the higher priority queue.
- Fig. 4.2.2 shows multilevel feedback queue scheduling.

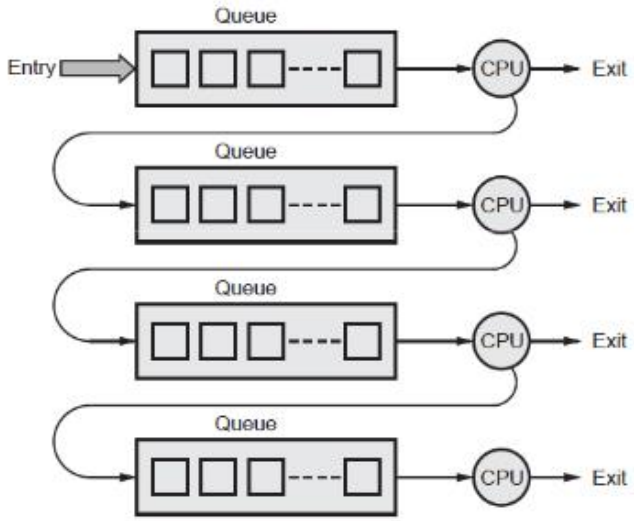


Fig. 4.2.2 Multilevel feedback queue scheduling

- MLFQ also uses concept of aging to prevent starvation. The processes with lower priority will move towards the higher priority queue in aging concept. The key to MLFQ scheduling lies in how the scheduler sets priorities. Rather than giving a fixed priority to each process, MLFQ varies the priority of a process based on its observed behavior.
- Assume we have three queues (Q0, Q1 and Q2). Q0 is the lowest priority queue and Q2 is the highest priority queue. The process enters at the highest priority (Q2). After a single time-slice of 10 ms, the scheduler reduces the process's priority by one, and thus the job is on Q1. After running at Q1 for a time slice, the job is finally lowered to the lowest priority in the system (Q0), where it remains.
- Following parameters are defined by scheduler for implementing MLFQ :
  1. Number of queue required.
  2. Scheduling algorithm for each queue.
  3. Method is required for finding to increase and/or decrease priority of processes.

Ex. 4.2.1 *Consider the following four jobs.*

Job	Burst Time
J1	8
J2	5
J3	5
J4	13

Find average waiting time for
i) FCFS
ii) SJF

MSBTE : Winter-18, Marks 4

Sol. : i) FCFS

ii) SJF

Waiting Time :

Process	FCFS	SJF
J1	0	10
J2	8	0
J3	13	5
J4	18	18

Average Waiting Time for FCFS = $\frac{0+8+13+18}{4} = 9.75$

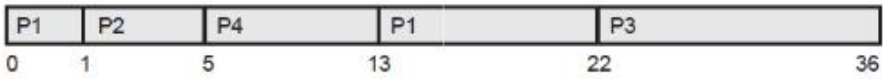
Average Waiting Time for SJF = $\frac{10+0+5+18}{4} = 8.25$

Ex. 4.2.2 : *The Jobs are scheduled for execution as follows - solve the problem by using preemptive SJF (Shortest Job First).*

Find average waiting time using Gantt chart. **MSBTE : Summer-18, Marks 4**

Process	Arriaval Time	Burst Time
P1	0	10
P2	1	04
P3	2	14
P4	3	08

Sol. : Preemptive SJF



Waiting Time :

Process	Waiting Time
P1	0 – 0 + 13 – 1 = 12
P2	1 – 1 = 0
P3	22 – 2 = 20
P4	5 – 3 = 2

$$\text{Average Waiting Time} = \frac{12+0+20+2}{4} = 8$$

Ex. 4.2.3 : *The job are scheduled for execution as follows solve the problem using :*
i) SIS ii) FCFS
also find average waiting time using Gantt chart. **MSBTE : Winter-17, Marks 4**

Process	Arrival	Burst time
P1	0	8
P2	1	4
P3	2	9
P4	3	5

Sol. : i) SJF : Gantt chart

ii) FCFS : Gantt chart

Waiting Time :

Process	SJF	FCFS
P_1	0 – 0 = 0	0 – 0 = 0
P_2	8 – 1 = 7	8 – 1 = 7
P_3	17 – 2 = 15	12 – 2 = 10
P_4	12 – 3 = 9	21 – 3 = 18
Average waiting time	$\frac{0+7+15+9}{4} = 7.75$	$\frac{0+7+10+18}{4} = 8.75$

Ex. 4.2.4 : *Consider the following set of processes, with the length of the CPU burst given in milliseconds.*

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

Find out average waiting time by using

(i) Nonpreemptive priority *(ii) Round-Robin (RR) (quantum = 1)* **MSBTE : Summer-17, Marks 4**

Sol. : i) Nonpreemptive Priority :

Fig. 4.2.3 Gantt chart

ii) Round Robin :

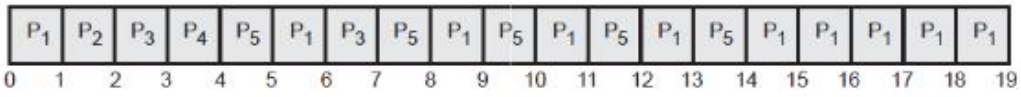


Fig. 4.2.4 Gantt chart

Waiting Time :

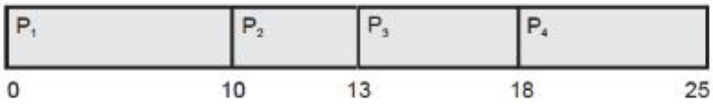
Process	Non-preemptive Priority	Round Robin
P_1	6	9
P_2	0	1
P_3	16	5
P_4	18	3
P_5	1	9
Average waiting time	$\frac{6+0+16+18+1}{5} = 8.2$	$\frac{9+1+5+3+9}{5} = 5.4$

Ex. 4.2.5 : *Solve the following problem using SJF and Round Robin (RR) scheduling algorithm. Find average waiting time for each algorithm.* **MSBTE : Summer-16 Marks 8**

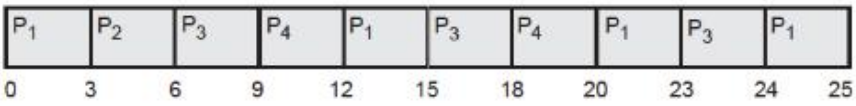
Process	Burst time
P_1	10
P_2	3
P_3	7
P_4	5

Sol. : Gantt chart

i) SJF



ii) Round Robin (Time slice = 3)



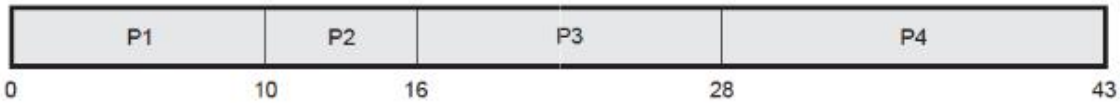
Waiting Time :

Process	SJF	Round Robin
P_1	0	0 + 12 – 3 + 20 – 15 + 24 – 23 = 15
P_2	10	3
P_3	13	6 + 15 – 9 + 23 – 18 = 17
P_4	18	9 + 18 – 12 = 15
Average waiting time	$\frac{0+10+13+18}{4} = 10.25$	$\frac{15+3+17+15}{4} = 12.5$

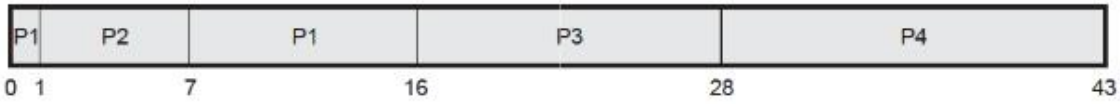
Ex. 4.2.6 *Explain the FCFS, preemptive and non preemptive versions of shortest-job-first and roudn robin (time slice = 2) schedulign algorithms with Gantt chart for the four processes given. Compare their average turn around and waiting time.*

Process	Arrival time	Burst time
P1	0	10
P2	1	6
P3	2	12
P4	3	15

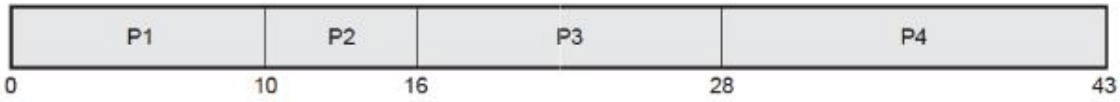
Ans. : a) Gantt chart for FCFS :



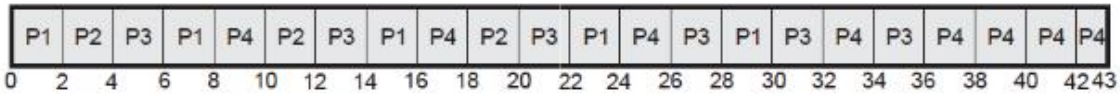
b) Gantt chart for preemptive SJF :



c) Gantt chart for non-preemptive SJF :



d) Gantt chart for round robin :



Waiting time

Process	FCFS	Preemptive SJF	Nonpreemptive SJF	RR
P1	0	6	0	20
P2	9	0	9	13
P3	14	14	14	22
P4	25	25	25	25

Average waiting time

$$\text{FCFS} = \frac{0+9+14+25}{4} = \frac{48}{4} = 12$$

$$\text{Preemptive SJF} = \frac{6+0+14+15}{4} = \frac{35}{4} = 8.75$$

$$\text{Nonpreemptive SJF} = \frac{0+9+14+15}{4} = \frac{48}{4} = 12$$

$$\text{RR} = \frac{20+13+22+25}{4} = \frac{80}{4} = 20$$

Turnaround time

Process	FCFS	Preemptive SJF	Nonpreemptive SJF	RR
P1	10	16	10	30
P2	15	6	15	19
P3	26	26	26	34
P4	40	40	40	40

Average turnaround time

$$\text{FCFS} = \frac{10+15+26+40}{4} = \frac{91}{4} = 22.75$$

$$\text{Preemptive SJF} = \frac{16+6+26+40}{4} = \frac{88}{4} = 22$$

$$\text{Nonpreemptive SJF} = \frac{10+15+26+40}{4} = \frac{91}{4} = 22.75$$

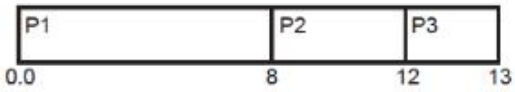
$$\text{RR} = \frac{30+19+34+40}{4} = \frac{123}{4} = 30.75$$

Ex. 4.2.7 *What is the average turnaround time for the following process using*
a) FCFS b) SJF non-prrmptive c) Preemptive SJF.

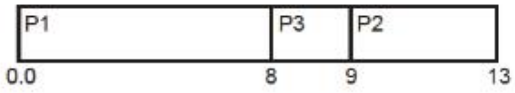
Process	Arrival Time	Burst Time
P1	0.0	8
P2	0.4	4
P3	1.0	1

Ans. : Gantt chart :

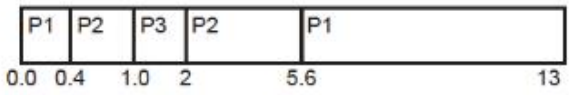
a) FCFS



b) SJF non-preemptive



c) Preemptive SJF



Process	Waiting Time		
	FCFS	SJF Non-preemptive	Preemptive SJF
P1	0	0	5.2
P2	7.6	8.6	1
P3	11	7	0

Process	Turnaround Time		
	FCFS	SJF Non-preemptive	Preemptive SJF
P1	8	8	13.2
P2	11.6	12.6	5
P3	12	8	1
Average turnaround time	**7.2**	**9.53**	**6.4**

Board Questions

1. *Expalin SJF algorithm with example. Also calculate average waiting time.* **MSBTE : Summer-15, Marks 8**
2. *What is FCFS algorithm ? Describe with example.* **MSBTE : Summer-16, Marks 4**
3. *Explain how priority scheduling algorithm works with suitable example, also list advantages and disadvantages.* **MSBTE : Summer-16, Marks 8**
4. *Explain priority scheduling algorithm with example. List its advantages and disadvantages.* **MSBTE : Winter-16, Marks 8**
5. *With neat diagram, explain multilevel queue scheduling.* **MSBTE : Summer-17, Marks 4**
6. *Explain Round Robin algorithm with suitable example.* **MSBTE : Winter-17,18, Mrks 4**
7. *Explain multilevel queue scheduling.* **MSBTE : Winter-18, Marks 4**

4.3 Deadlock

4.3.1 System Model

- A lack of process synchronization can result in two extreme conditions are deadlock or starvation. Deadlock is the problem of multiprogrammed system.
- Deadlock can be defined as the permanent blocking of a set of processes that either complete for system resources. Deadlock is more serious than starvation.
- Deadlock can occur on sharable resources such as files, printers, database, disks, tape drives, memory, CPU cycles etc.
- A process is in deadlock state if it was waiting for a particular event that will not occur. In a system deadlock, one or more processes are deadlocked.

Deadlock Example

- Computer system is collection of limited/finite number of resources. These resources are distributed among a number of competing processes.

Resources are of two types :

1. Reusable resources
2. Consumable resources.

- **Reusable resource** is used only by one process at a time. Process can release resource after use. Processors, I/O channel, I/O device, files, database, primary and secondary memory, semaphore are example of the reusable resource.
- Consumable resource is one that can be created and destroyed. There is no limit on the number of consumable resources of a particular type.
- An interrupt, messages, signals and messages in I/O buffers are examples of consumable resources.
- Process utilize the resources in the following sequence :
  1. Request for resource
  2. Use of resource
  3. Resource release.
- Process uses system call for requesting the resource. After allocating resource to the process, it use or operate the resource.
- The process releases the resource after use.
- If the resource is not available when it is requested, the requesting process is forced to wait.
- Fig. 4.3.1 shows the deadlock with 4 process.

Process 1	Process 2	Process 3	Process 4
Printf(" ")	int a;	Procedure ( )	sort ( )
- - - - -	- - - - -	- - - - -	- - - - -
- - - - -	- - - - -	- - - - -	- - - - -
request (resource 1); /* Holding res 1 */	request (resource 2); /* Holding res 2 */	request (resource 3); /* Holding res 3 */	request (resource 4) /* Holding res 4*/
- - - - -	- - - - -	- - - - -	- - - - -
- - - - -	- - - - -	- - - - -	- - - - -
request (resource 2);	request (resource 3);	request (resource 4);	request(resource 1);

Fig. 4.3.1 Deadlock with 4 process

- Process 1 is holding resource 1 and requesting resource 2; process 2 is holding resource 2 and requesting resource 3; process 3 is holding resource 3 and requesting resource 4. Process 1 is holding resource 4 also.
- Here deadlock occurs because none of the processes can proceed because all are waiting for a resource held by another blocked process.
- To break this situation, one of the process release the resource.
- Operating system must be handle the deadlock situation. OS detect the deadlock and solve the deadlock problem.

4.3.2 Resource Allocation Graphs

- Resource allocation graph is introduced by Holt. It is a directed graph that depicts a state of the system of resources and processes.
- Process and resource are represented by node in directed graph. Graph consists of a set of vertices (V) and set of edges (E).
- A process node is graphically represented by circle and shown in Fig. 4.3.2

Fig. 4.3.2 Process

- A resource node is graphically represented by a rectangle and represents one type of resources. Fig. 4.3.3 shows resources node.

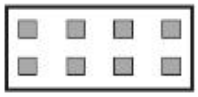
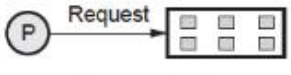


Fig. 4.3.3 Resource with 8 instances

- The number of bullet symbols in a resource node indicates how many units of that resource class exist in the system.
- An arrow from the process to resource indicates the process is requesting the resource. An arrow from resource to process shows an instance of the resource has been allocated to the process.
- Claim edge $P \rightarrow R$ indicated that process P may request resource R in the future ; represented by a dashed line. Claim edge converts to request edge when a process requests a resource. Fig. 4.3.4 shows request and claim edge.



(a) Request edge

(b) Claim edge

Fig. 4.3.4 Edges

- Request edge converted to an assignment edge when the resource is allocated to the process. When a resource is released by a process, assignment edge reconverts to a claim edge.
- Fig. 4.3.5 shows a resource allocation graph. System consists of process and resources.

  Process : P_1, P_2

  Resource : R_1, R_2

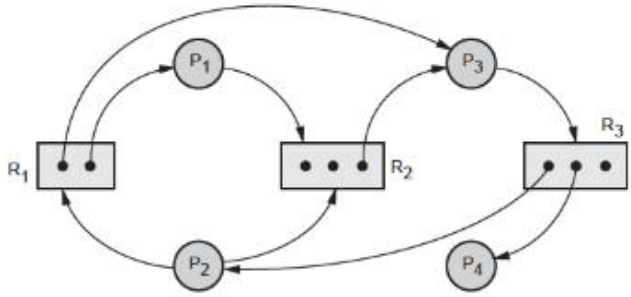


Fig. 4.3.5 Resource allocation graph

Resource	Number of instance
R_1	2
R_2	3
R_3	3

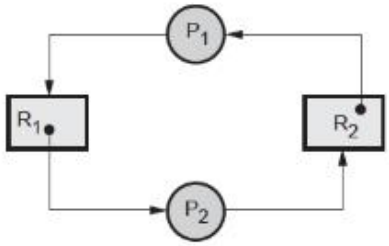


Fig. 4.3.6 Circular wait with deadlock

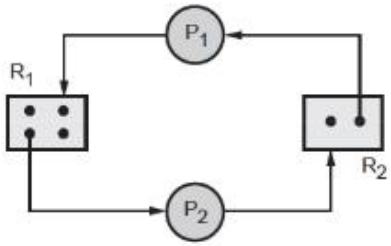


Fig. 4.3.7 Circular wait but no deadlock

Fig. 4.3.8 shows the resource allocation graph. System consists of four processes (P_1, P_2, P_3, P_4) and four resources (R_1, R_2, R_3, R_4).

Resource	Number of instance
R_1	2
R_2	2
R_3	3
R_4	6
R_5	3

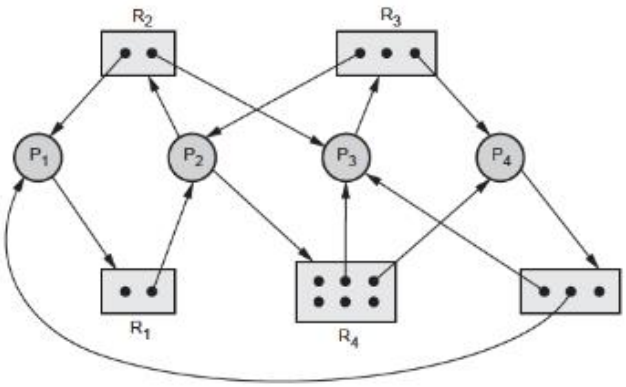


Fig. 4.3.8 Resource allocation graph

- Resource managers and other operating system processes can be involved in a deadlock. Deadlock is a global condition rather than a local one.
- An individual program cannot generally detect a deadlock. It is blocked and unable to use the processor to do any work. Deadlock detection must be handled by the operating system.

4.3.3 Necessary Condition for Deadlock

- Following four conditions are necessary for deadlock to exist.
  1. Mutual exclusion 2. Hold and wait
  3. No preemption 4. Circular wait

1. **Mutual exclusion :** A resource may be acquired exclusively by only one process at a time.
2. **Hold and wait :** Processes currently holding resources that were granted earlier can request new resources.
3. **No preemption :** Once a process has obtained a resource, the system cannot remove it from the process control until the process has finished using the resource.
4. **Circular wait :** A circular chain of hold and wait condition exists in the system.

- All four of these conditions must be present for a resource deadlock to occur. If one of them is absent, no resource deadlock is possible.

4.3.4 Deadlock Handing- Deadlock Prevention

- To prevent a deadlock, the OS must eliminate one of the four necessary conditions.
  1. Mutual exclusion 2. Hold and wait
  3. No preemption 4. Circular wait

1. **Mutual exclusion :** It is necessary in any computer system because some resources (memory, CPU) must be exclusively allocated to one user at a time. No other process can use a resource while it is allocated to a process.
2. **Hold and wait :** If a process holding certain resources is denied a further request, it must release its original resources and if required request them again.
3. **No preemption :** It could be bypassed by allowing the operating system to deallocate resources from process.
4. **Circular wait :** Circular wait can be bypassed if the operating system prevents the formation of a circle.

- A deadlock is possible only if all four of these conditions simultaneously hold in the system.
- Prevention strategies ensure that at least one of the conditions is always false.

4.3.5 Deadlock Avoidance

- Deadlock avoidance depends on additional information about the long term resource needs of each process. The system must be able to decide whether granting a resource is safe or not and only make the allocation when it is safe.
- Fig. 4.3.9 shows safe and unsafe state.

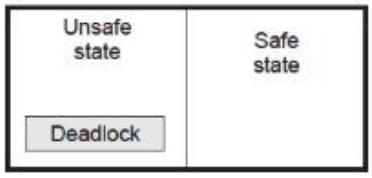


Fig. 4.3.9 Safe and unsafe state

- When a process is created, it must declare its maximum claim, i.e. the maximum number of unit resource. The resource manager can grant the request if the resources are available.

- For example, if process 1 requests a resource held by process 2 then make sure that process 2 is not waiting for resource head by first process 1.

4.3.5.1 Banker's Algorithm

- Banker's algorithm is the deadlock avoidance algorithm. Banker's is named because the algorithm is modeled after a banker who makes loads from a pool of capital and receives payments that are returned to that pool.
- Algorithm is check to see if granting the request leads to an unsafe state. If it does, the request is denied. If granting the request leads to a safe state, it is carried out.
- The Dijkstra proposed an algorithm to regulate resource allocation to avoid deadlocks. The banker's algorithm is the best known of the avoidance method.
- By using avoidance method, the system is always kept in a safe state.
- It is easy to check if a deadlock is created by granting a request, deadlock analysis method is used for this.
- Deadlock avoidance uses the worst-case analysis method to check for future deadlocks.
- **Safe state :** There is at least one sequence of resource allocations to processes that does not result in a deadlock.
- System is in a safe state only if there exist a safe sequence. A safe state is not a deadlocked state.
- Deadlocked state is an unsafe state. It does mean the system is in a deadlock.
- As long as the state is safe, the resource manager can be guaranteed to avoid a deadlock.
- Initially the system is in a safe state. When process requests a resource and that resource is available then the system must decide whether the resources can be allocated immediately or process must wait.
- If system remains in safe state after allocating resource then only OS allocates resources to process.
- Banker algorithm uses following data structures.

1. Allocation : Allocation is a table in which row represents process and column represents resources (R).

alloc [i, j] = Number of unit of resource R_j held by process P_i.

2. Max : Max be the maximum number of resources that process requires during its execution.

- **Need (claim) :** It is current claim of a process, where a process's claim is equal to its maximum need minus its current allocation.

  Need = Max – Allocation

- **Available :** There will be number of resources still available for allocation. This is equivalent to the total number of resources minus the sum of the allocation to all processes in the system.

  Available = Number of resources – Sum of the allocation to all process

  $$= \text{Number of resources} - \sum_{i=1}^{n} \text{Allocation}(P_i)$$

- Each process cannot request more that the total number of resources in the system. Each process must also guarantee that once allocated a resource, the process will return that resource to the system within a finite time.

Weakness of Banker's algorithm

1. It requires that there be a fixed number of resources to allocate.
2. The algorithm requires that users state their maximum needs (request) in advance.
3. Number of users must remain fixed.
4. The algorithm requires that the bankers grant all requests within a finite time.
5. Algorithm requires that process returns all resource within a finite time.

Examples on Banker's algorithm

1. System consists of five processes (P_1, P_2, P_3, P_4, P_5) and three resources (R_1, R_2, R_3). Resource type R_1 has 10 instances, resource type R_2 has 5 instances and R_3 has 7 instances. The following snapshot of the system has been taken :

Process	Allocation			Max			Available		
	R_1	R_2	R_3	R_1	R_2	R_3	R_1	R_2	R_3
P_1	0	1	0	7	5	3	3	3	2
P_2	2	0	0	3	2	2			
P_3	3	0	2	9	0	2			
P_4	2	1	1	2	2	2			
P_5	0	0	2	4	3	3			

Content of the matrix need is calculated as
Need = Max – Allocation.

Process	Need		
	R_1	R_2	R_3
P_1	7	4	3
P_2	1	2	2
P_3	6	0	0
P_4	0	1	1
P_5	4	3	1

Currently the system is in safe state.

Safe sequence : Safe sequence is calculated as follows :

1) Need of each process is compared with available. If $need_i \leq available_i$, then the resources are allocated to that process and process will release the resource.
2) If need is greater than available, next process need is taken for comparison.
3) In the above example, need of process P_1 is (7, 4, 3) and available is (3, 3, 2).

   need $\geq$ available $\rightarrow$ False

   So system will move for next process.
4) Need for process P_2 is (1, 2, 2) and available (3, 3, 2), so

   need $\leq$ available (work)

   (1, 2, 2) $\leq$ (3, 3, 2) = True

   then Finish [i] = True

   Request of P_2 is granted and processes P_2 is release the resource to the system.

   Work : = Work + Allocation

   Work : = (3, 3, 2) + (2, 0, 0)

   : = (5, 3, 2)

   This procedure is continued for all processes.
5) Next process P_3 need (6, 0, 0) is compared with new available (5, 3, 2).

   Need > Available = False

   (6 0 0) > (5 3 2)
6) Process P_4 need (0, 1, 1) is compared with available (5, 3, 2).

   Need < Available

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

   Available = Available + Allocation

   = (5 3 2) + (2 1 1)

   = (7 4 3) (New available)
7) Then process P_5 need (4 3 1) is compared with available (7 4 3)

   Need < Available

   (4 3 1) < (7 4 3) = True

   Available = Available + Allocation

   = (7 4 3) + (0 0 2) = (7 4 5)

   (New available)

8) One cycle is completed. Again system takes all remaining process in sequence. So process P_1 need (7 4 3) is compared with new available (7 4 5).

Need < Available = True

(7 4 3) < (7 4 5)

Available = Available + Allocation

= (7 4 5) + (0 1 0) = (7 5 5)

(New available)

9) Process P_3 need is (6 0 0) is compared with new available (7 5 5).

∴ Need < Available = True

(6 0 0) < (7 5 5) = True

Available = Available + Allocation

= (7 5 5) + (3 0 2)

= (10 5 7) = (New available)

Safe sequence is < P_2 P_4 P_5 P_1 P_3 >

Ex. 4.3.1 *Consider the following snapshot of a system. Using the banker's algorithm.*

Process	Allocation				Request				Available			
	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4	R_1	R_2	R_3	R_4
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

If a request from P_1 arrives for (0, 4, 2, 0), can the request be granted immediately ?

Sol. :

i) Content of need matrix is

Process	R_1	R_2	R_3	R_4
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

Safe state

a) Need of P_0 = (0, 0, 0, 0)

Available = (1, 5, 2, 0)

Need < Available

(Required resources are allocated to P_0)

∴ New available = Allocation + Available

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

= (1, 5, 3, 2)

b) For P_1 process

Need = (0, 7, 5, 0)

Available = (1, 5, 3, 2)

Need > Available

(Request is not granted)

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

c) For P_2 process

Need = (1, 0, 0, 2)

Available = (1, 5, 3, 2)

Need < Available

(Request is granted)

New available = Allocation + Available

= (1, 3, 5, 4) + (1, 5, 3, 2)

= (2, 8, 8, 6)

d) For process P_3

Need of P_3 = (0, 0, 2, 0)

Available = (2, 8, 8, 6)

Need < Available

New available = (2, 14, 11, 8)

e) Process P_4

Need of P_4 = (0, 6, 4, 2)

Available = (2, 14, 11, 8)

Need < Available

New available = (2, 14, 11, 8) + (0, 0, 1, 4)

= (2, 14, 12, 12)

f) Again for P_1 process

Need of P_1 = (0, 7, 5, 0), Available = (2, 14, 12, 12)

Need < Available

New available = (1, 0, 0, 0) + (2, 14, 12, 12)

= (3, 14, 12, 12)

So the safe sequence is P_0, P_2, P_3, P_4, P_1

Request from P_1 = (0, 4, 2, 0)

Available = (1, 5, 2, 0)

After granting the request of P_1, available resource is (1, 1, 0, 0)

a) Need of P_0 = (0, 0, 0, 0)

New available = (1, 1, 0, 0) + (0, 0, 1, 2)

= (1, 1, 1, 2)

b) P_1 need is greater than available.

c) P_2 need is (1, 0, 0, 2)

Need < Available

New available = (1, 1, 1, 2) + (1, 3, 5, 4)

= (2, 4, 6, 6)

d) P_3 Need is (0, 0, 2, 0)

Need < Available

New available = (2, 4, 6, 6) + (0, 6, 3, 2)

= (2, 10, 9, 8)

e) P_4 Need is (0, 6, 4, 2)

Available (2, 10, 9, 8)

Need < Available

∴ New available= (2, 10, 9, 8) + (0, 0, 1, 4)

= (2, 10, 10, 12)

f) Again P_1 need (0, 7, 5, 0)

Need < Available

If a request from P_1 arrives for (0, 4, 2, 0), then **the request is granted immediately.**

Ex. 4.3.2 *Consider the following system snapshot using data structures in the Banker's algorithm, with resources A, B, C and D and process P_0 to P_4.*

	Max				Allocation				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P_0	6	0	1	2	4	0	0	1					3	2	1	1
P_1	1	7	5	0	1	1	0	0								
P_2	2	3	5	6	1	2	5	4								
P_3	1	6	5	3	0	6	3	3								
P4	1	6	5	6	0	2	1	2								

Using Banker's algorithm, answer the following questions :

a) How many resources of type A, B, C and D are there ?

b) What are the contents of the need matrix ?

c) Is the system in a safe state ? Why ?

d) If a request from process P_4 arrives for additional resources of (1, 2, 0, 0), can the Banker's algorithm grant the request immediately ? Show the new system state and other criteria.

Sol. : a) Resource types of A, B, C and D :

	Allocation			
	A	B	C	D
P_0	4	0	0	1
P_1	1	1	0	0
P_2	1	2	5	4
P_3	0	6	3	3
P_4	0	2	1	2
	6	**11**	**9**	**10**

	Available			
	A	B	C	D
	3	2	1	1
	6	11	9	10
Total	**9**	**13**	**10**	**11**

b) Need Matrix :

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

c) Safe State :

Process	Need		Available	T/F	Available
P_0	(2, 0, 1, 1)	<	(3, 2, 1, 1)	T	(7, 2, 1, 2)
P_1	(0, 6, 5, 0)	<	(7, 2, 1, 2)	F	(7, 2, 1, 2)
P_2	(1, 1, 0, 2)	<	(7, 2, 1, 2)	T	(8, 4, 6, 6)

P_3	(1, 0, 2, 0)	<	(8, 4, 6, 6)	T	(8, 10, 9, 9)
P_4	1, 4, 4, 4)	<	(8, 10, 9, 9)	T	(8, 12, 10, 11)
P_1	(0, 6, 5, 0)	<	(8, 12, 10, 11)	T	(9, 13, 10, 11)

Safe sequence : P_0, P_2, P_3, P_4, P_1

d) Request from process P_4 (1, 2, 0, 0)

We assume that, requested is granted immediately. So calculate need matrix :

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

Available = (2, 0, 1, 1)

Calculate safe sequence:

Process	Need		Available	T/F	Available
P_0	(2, 0, 1, 1)	<	(2, 0, 1, 1)	F	(2, 0, 1, 1)
P_1	(0, 6, 5, 0)	<	(2, 0, 1, 1)	F	(2, 0, 1, 1)
P_2	(1, 1, 0, 2)	<	(2, 0, 1, 1)	T	(3, 2, 6, 5)
P_3	(1, 0, 2, 0)	<	(3, 2, 6, 5)	T	(3, 8, 9, 8)
P_4	(0, 2, 4, 4)	<	(3, 8, 9, 8)	T	(4, 12, 10, 10)
P_0	(2, 0, 1, 1)	<	(4, 12, 10, 10)	T	(8, 12, 10, 11)
P_1	(0, 6, 5, 0)	<	(8, 12, 11, 11)	T	(9, 13, 10, 11)

Safe sequence : P_2, P_3, P_4, P_0, P_1

So requested is granted immediately.

Ex. 4.3.3 *Consider a system with five processes and three resource types and at time T_0 the following snapshot of the system has been taken :*

	Allocated			*Maximum*			*Available*		
Process Id	*R1*	*R2*	*R3*	*R1*	*R2*	*R3*	*R1*	*R2*	*R3*
P1	*1*	*1*	*2*	*4*	*3*	*3*	*3*	*1*	*0*
P2	*2*	*1*	*2*	*3*	*2*	*2*			
P3	*4*	*0*	*1*	*9*	*0*	*2*			
P4	*0*	*2*	*0*	*7*	*5*	*3*			
P5	*1*	*1*	*2*	*11*	*2*	*3*			

i) Determine the total amount of resources of each type.
ii) Compute the need matrix
iii) Determine if the state is safe or not using Banker's algorithm.
iv) Would the following request be granted in the current state?
a) P1 < 3, 3, 1 >
b) P2 < 2, 1, 0 >

Sol. :

i) The total amount of resources of each type : Sum of allocated + Available

	Allocated			Available		
Process Id	**R1**	**R2**	**R3**	**R1**	**R2**	**R3**
P1	1	1	2	3	1	0
P2	2	1	2			
P3	4	0	1			
P4	0	2	0			
P5	1	1	2			
Total	**8**	**5**	**7**			

	R1	R2	R3
The total amount of resources of each type =	8	5	0 (Allocated)
	3	1	0 (Available)
The total amount of resources of each type =	11	6	7

R1 = 11, R2 = 6, R3 = 7

ii) The need matrix

Need matrix = Maximum – Allocated

	Maximum				Allocated				Need matrix		
Process Id	**R1**	**R2**	**R3**		**R1**	**R2**	**R3**		**R1**	**R2**	**R3**
P1	4	3	3		1	1	2		3	2	1
P2	3	2	2	–	2	1	2	=	1	1	0
P3	9	0	2		4	0	1		5	0	1
P4	7	5	3		0	2	0		7	3	3
P5	11	2	3		1	1	2		10	1	1

iii) State is safe or not using Banker's algorithm

Steps for safe state :

First Iteration :

1. Process P1 need is (3, 2, 1) and resource available with system is (3, 1, 0).

Here need is greater than available i.e.

Need (3, **2, 1**) > Available (3, **1, 0**)

Condition is false and algorithm selects next process for resource allocation.

2. Process P2 need is (1, 1, 0) and resource available with system is (3, 1, 0).

Here need is less than available i.e.

Need (1, 1, 0) < Available (3, 1, 0)

Condition is true and algorithm allocates required resources to that process (P2 Process). After completion of work, process P2 releases all the resources. So resources available with system are increased. So,

New available = Previous available + Allocated of particular process

= (3, 1, 0) + (2, 1, 2)

New available = (5, 2, 2)

3. Process P3 need is (5, 0, 1) and resource available with system is (5, 2, 2).

Here need is less than available i.e.

Need (5, 0, 1) < Available (5, 2, 2).

Condition is true and algorithm allocates required resources to that process (P3 Process). After completion of work, process P3 releases all the resources. So resources available with system are increased. So,

New available = Previous available + Allocated of particular process

= (5, 2, 2) + (4, 0, 1)

New available = (9, 2, 3)

4. Process P4 need is (7, 3, 3) and resource available with system is (9, 2, 3).

Here need is greater than available i.e.

Need (7, **3,** 3) > Available (9, **2**, 3)

Condition is false and algorithm selects next process for resource allocation.

5. Process P5 need is (10, 1, 1) and resource available with system is (9, 2, 3).

Here need is greater than available i.e.

Need (**10,** 1, 1) > Available (**9**, 2, 3)

Condition is false and algorithm selects next process for resource allocation.

One loop is completed; again we start for remaining three processes P1, P4 and P5

Second Iteration: for process P1, P4 and P5

1. Process P1 need is (3, 1, 2) and resource available with system is (9, 2, 3).

Here need is less than available i.e.

Need (3, 1, 2) < Available (9, 2, 3).

Condition is true and algorithm allocates required resources to that process (P1 Process). After completion of work, process P1 releases all the resources. So resources available with system are increased. So,

New available = Previous available + Allocated of particular process

= (9, 2, 3) + (1, 1, 2)

New available = (10, 3, 5)

2. Process P4 need is (7, 3, 3) and resource available with system is (10, 3, 5).

   Here need is less than available i.e.

   Need (7, 3, 3) < Available (10, 3, 5).

   Condition is true and algorithm allocates required resources to that process (P4 Process). After completion of work, process P4 releases all the resources. So resources available with system are increased. So,

New available = Previous available + Allocated of particular process

= (10, 3, 5) + (0, 2, 0)

New available = (10, 5, 5)

3. Process P5 need is (10, 1, 1) and resource available with system is (10, 5, 5).

   Here need is less than available i.e.

   Need (10, 1, 1) < Available (10, 5, 5).

Condition is true and algorithm allocates required resources to that process (P5 Process). After completion of work, process P5 releases all the resources. So resources available with system are increased. So,

New available = Previous available + Allocated of particular process

= (10, 5, 5) + (1, 1, 2)

New available = (11, 6, 7)

Safe sequence = P2, P3, P1, P4, P5

iv) Would the following request be granted in the current state ?

a) P1 <3, 3, 1>

We assume that requested is granted for process P1. Then the need matrix will be

	Maximum				Allocated				Need matrix		
Process Id	R1	R2	R3		R1	R2	R3		R1	R2	R3
P1	4	3	3		4	4	3		0	0	0
P2	3	2	2		2	1	2		1	1	0
P3	9	0	2	−	4	0	1	=	5	0	1
P4	7	5	3		0	2	0		7	3	3
P5	11	2	3		1	1	2		10	1	1

But available with the system is less that the required one. So P1 requested is not granted.

b) P2 < 2, 1, 0 >

We assume that requested is granted for process P2. Then the need matrix will be

	Maximum				Allocated				Need matrix		
Process Id	R1	R2	R3		R1	R2	R3		R1	R2	R3
P1	4	3	3		1	1	2		3	2	1
P2	3	2	2	–	4	2	2	=	0	0	0
P3	9	0	2		4	0	1		5	0	1
P4	7	5	3		0	2	0		7	3	3
P5	11	2	3		1	1	2		10	1	1

New available = (3, 1, 0) – (2, 1, 0) = (1, 0, 0)

Compute the safe sequence :

1. Need of P1 (3, 2, 1) is greater than available (1, 0, 0). Condition is false. Select next process.
2. Need of P2 (0, 0, 0) is less than available (1, 0, 0). Condition is true. Then
   New available = (1, 0, 0) + (4, 2, 2) = (5, 2, 2).
3. Need of P3 (5, 0, 1) is greater than available (5, 2, 2). Condition is false. Select next process.
4. Need of P4 (7, 3, 3) is greater than available (5, 2, 2). Condition is false. Select next process.
5. Need of P5 (10, 1, 1) is greater than available (5, 2, 2). Condition is false. Select next process.

Again repeat for second iteration

1. Need of P1 (3, 2, 1) is greater than available (5, 2, 2). Condition is true. Then
   New available = (5, 2, 2) + (1, 1, 2) = (6, 3, 4)

There is no safe sequence because need of process P3, P4 and P5 is greater than the available. So request is not granted immediately.

Board Questions

1. *Define deadlock. What are necessary conditions for deadlock ?* **MSBTE : Summer-15, Marks 4**
2. *Write steps for Banker's algorithm to avoid deadlock. Also give one example.* **MSBTE : Summer-15, Marks 4**
3. *Explain multilevel queue scheduling with example.* **MSBTE : Summer-15, Marks 8**
4. *Write steps for Banker's algorithm to avoid deadlock.* **MSBTE : Winter-15, Marks 4**
5. *State necessary condition for deadlock.* **MSBTE : Winter-15, Summer-17, Marks 4**
6. *Explain in detail how deadlock can be handled.* **MSBTE : Winter-16, Marks 4**
7. *Write steps for Banker's algorithm to avoid deadlock. Also give one example showing working of Banker's algorithm.* **MSBTE : Summer-17, Marks 8**

8. *Enlist and describe in details conditions leading to Deadlocks.* **MSBTE : Winter-17, Marks 4**
9. *List four Deadlock prevention condition and explain the following terms.*
   1) *Removal of "No preemption" condition.*
   2) *Elimination of "Circular wait" related to deadlock prevention condition.*

   MSBTE : Summer-18, Marks 6
10. *Explain Deadlock avoidance with example.* **MSBTE : Summer-18, Marks 4**
11. *Describe any four condition for deadlock.* **MSBTE : Winter-18, Marks 4**
12. *Write steps for Banker's algorithm to avoid deadlock.* **MSBTE : Winter-18, Marks 4**

4.4 Two Marks Questions with Answers

Q.1 What is non-preemptive scheduling ?

Ans. : Non-preemptive scheduling ensures that a process relinquishes control of the CPU only when it finishes with its current CPU burst.

Q.2 Define priority inversion problem.

Ans. : The higher priority process would be waiting for the low priority one to finish. This situation is known as priority inversion problem.

Q.3 What is Shortest-Remaining-Time-First (SRTF) ?

Ans. : If a new process arrives with CPU burst length less than remaining time of current executing process, preempt. This scheme is known as the Shortest-Remaining-Time-First.

Q.4 What is response time ?

Ans. : Response time is the amount of time it takes from when a request was submitted until the first response is produced, not output.

Q.5 What is round robin CPU scheduling ?

Ans. : Each process gets a small unit of CPU time (time quantum). After this time has elapsed, the process is preempted and added to the end of the ready queue.

Q.6 What is meant by starvation in operating system ?

Ans. : Starvation is a resource management problem where a process does not get the resources (CPU) it needs for a long time because the resources are being allocated to other processes.

Q.7 What is an aging ?

Ans. : Aging is a technique to avoid starvation in a scheduling system. It works by adding an aging factor to the priority of each request. The aging factor must increase the requests priority as time passes and must ensure that a request will eventually be the highest priority request.

Q.8 What is convoy effect ?

Ans. : A convoy effect happens when a set of processes need to use a resource for a short time, and one process holds the resource for a long time, blocking all of the other processes. Essentially it causes poor utilization of the other resources in the system.

Q.9 Distinguish between CPU-bounded and I/O bounded processes.

Ans. :

CPU-bounded	I/O bounded processes
CPU Bound processes are ones that are implementing algorithms with a large number of calculations	Processes that are mostly waiting for the completion of input or output (I/O) are I/O Bound
Given a lower priority by the scheduler	Given high priority by the scheduler
CPU bound does too much computation to keep I/O busy	I/O bound does too much I/O to keep CPU busy
Example : matrix multiplication	Example : netscape

Q.10 State any four criteria in CPU scheduling. **MSBTE : Winter-18**

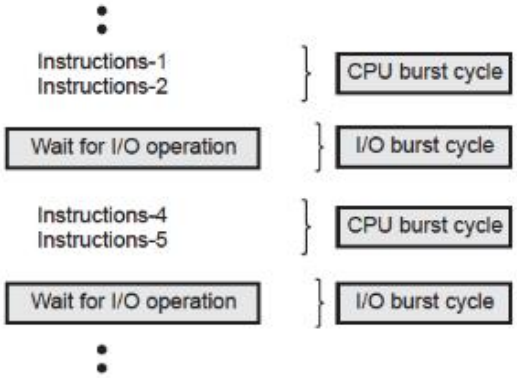
Ans. : Criteria in CPU scheduling are throughput, waiting time, response time, fairness, priority, turnaround time etc.

Q.11 Define deadlock. **MSBTE : Winter-18**

Ans. : Deadlock can be defined as the permanent blocking of a set of processes that either complete for system resources.

Q.12 Describe CPU and I/O burst cycle with suitable diagram. **MSBTE : Winter-18**

Ans. : Process starts with CPU burst cycle then I/O burst cycle again CPU burst cycle again I/O burst cycle. In this way the process completes its executions. Process will terminates after final CPU burst cycle completions.



Q.13 What is resource-allocation graph ?

Ans. : Deadlocks can be described more precisely in terms of a directed graph called a system resource-allocation graph.

Q.14 Write the three ways to deal the deadlock program.

Ans. : Three ways to deal with the deadlock problem :

1. Use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.
2. Allow the system to enter a deadlock state, detect it and recover.
3. Ignore the problem altogether and pretend that deadlocks never occur in the system.

Q.15 Define safe state.

Ans. : A state is safe if the system can allocate resources to each process (up to its maximum) in some order and still avoid a deadlock.

Q.16 Define request edge and assignment edge.

Ans. : There is a request edge from process P to resource R if and only if P is blocked waiting for an allocation of R. There is an assignment edge from resource R to process P if and only if P is holding an allocation of R.

Q.17 What is banker's algorithm ?

Ans. : Banker's algorithm is a deadlock avoidance algorithm that is applicable to a resource-allocation system with multiple instances of each resource type.

❑❑❑

Notes

UNIT - V

5 MEMORY MANAGEMENT

5.1 Basic Memory Management

- The memory management modules of an operating system are concerned with the management of primary memory. In a multiprogramming system, the user part of memory must be further subdivided to accomodate multiple processes. The task of subdivision is carried out dynamically by the operating system and is known as memory management.
- Memory consists of a large array of words or bytes each with its own address. Memory that the processors directly access for instructions and data.

5.1.1 Functions of Memory Management

1. Keeping track of the status of each memory location. Whether the memory location is allocated or free.
2. Determining allocation policy for memory.
3. Memory allocation technique, memory allocation information must be updated.
4. Deallocation technique and policy. A process may release the allocated memory.

- Central processing unit fetches instructions from memory according to the value of program counter. These instructions may cause additional loading from and storing to specific memory addresses. In this section, we discuss the memory related parameters like address binding, physical address, logical address and dynamic loading etc.

5.1.2 Basic Hardware of Memory

- CPU can access content of main memory and register directly. If the data is not available into the memory, it load into memory from disk.
- Registers are built on the processor. Using one cycle of the CPU clock, processor access data from register.
- Accessing memory may take many CPU clock cycle. Mismatch of speed between CPU and memory is overcome by using cache memory.
- The use of base and bound (limit) registers are restrict a process memory references upto a certain limit. Hardware is used to protect user address space.
- Each process requires its own address space operating system define legal address for each process. Maximum and minimum limit is also decided so that process can access only these legal address.
- Fig. 5.1.1 shows the protection of process by using registers.

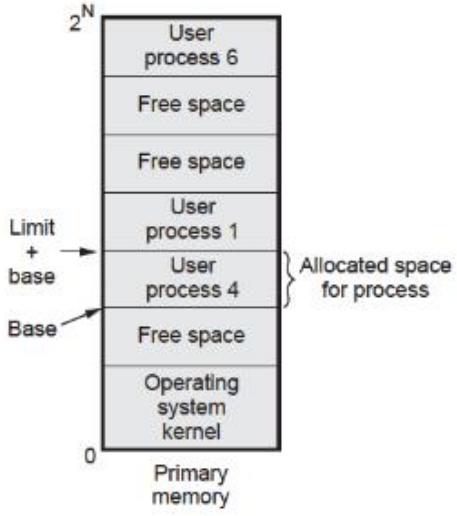


Fig. 5.1.1 Address space and memory

- An address space is the set of addresses that a process/program can use to address main memory. Each process has its own address space.
- User programs are loaded into consecutive memory locations by using base and limit register. When process is executing, the base register is loaded with the physical address where its program begins in memory and the limit register is loaded with the length of the program.
- Memory protection is used to avoid interference between programs existing in main memory. The memory protection hardware compares every memory address used by the program with the contents of two registers (base and limit) to ensure that it lies with the allocated memory area.
- Multiple hardware memories are used to provide a larger address space.
- The simplest method of memory protection is adding two registers to the CPU. This works good for all memory is allocated contiguously. Non-contiguous memory is harder to protect.
- When a process reads from or writes to address, the memory decoder adds on the value of the base register. The actual operation of read or write to address = base register + limit register.
- If the input address is higher than limit or lower than zero, then the memory hardware generates error. This is informed to the operating system by using interrupt. Processes can only access memory within these limits.
- Each process has its own pair of base register and limit register.

5.1.3 Address Binding

- Secondary storage device stores program in binary executable format. Before executing, the program is loaded into the main memory.
- Most of the operating systems allow a user process to store in any section of the main memory. Source program uses symbolic addresses.
- Fig. 5.1.2 shows processing of user program.
- Binding of instruction and data to main memory address is following ways :

  1. Compile time 2. Load time 3. Execution time

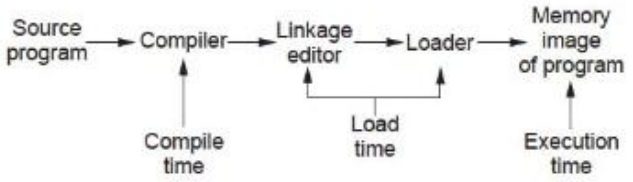


Fig. 5.1.2 Processing of user program

- **Compile time :** Source program is translated at compile time to produce a relocatable object module. At compile time, the translator generates code to allocate storage for the variable. This storage address is used for code reference. Target address is unknown at compile time, it cannot be bound at compile time. Example of compile time binding is MS DOS.com programs.
- **Load time :** Compiler generates relocatable code if compile time binding is not performed. The loader modifies the addresses in the load module at load time to produce the executable image stored in main memory. Final binding is delayed until program load time.
- **Execution time :** Memory address of the program is changed at execution time, then execution time binding is used. Binding is delayed until the run time of the program. Normally all operating system uses execution time binding. Special hardware is used for execution time binding.
- Memory allocation and deallocation is done using run-time support of the programming language in which a program is coded. Allocation and deallocation requests are made by calling appropriate routines of the run time library.
- Kernel is not involved in this kind of memory management.

5.1.4 Logical and Physical Address

- **Logical address** is generated by the CPU. This address is also called virtual address.
- Main memory address uses **physical address**. This address also called real address.
- **Logical address space :** Set of all logical addresses generated by a program.
- Logical address and physical address is identical when load time and compile time address binding is performed. The execution time address binding generates different physical and logical address.

- Memory Management Unit (MMU) is responsible for run time address mapping from vitual to physical address.

Dynamic Relocation

- Base register is sometimes called as a relocation register. The value of the relocation register is added to every address generated by a user process at the time it is sent to main memory.
- User can load a process with only absolute addresses for instructions and data, only when those specific addresses are free in main memory. Program's instruction, data and any other data structure required by the process can be accesssed easily if the addresses are relative.
- Fig. 5.1.3 shows dynamic relocation. User programs never reads the main memory physical address.

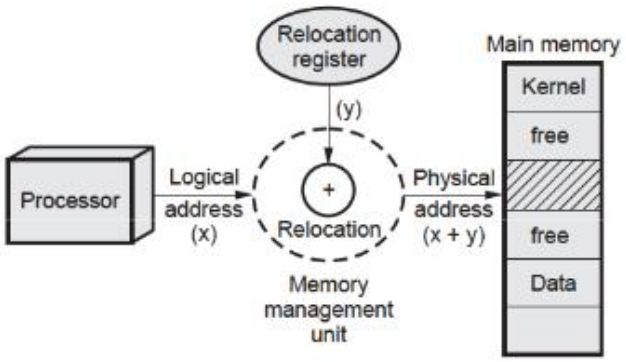


Fig. 5.1.3 Dynamic relocation

- Dynamic relocation requires extra hardware. It is mapping of the virtual address space to the physical address space at run time.
- Dynamic relocation makes it possible to move a partially executed process from one area of main memory into another without affecting other process.
- Problem with relocation is that, it is necessary to perform an addition and a comparison on every memory reference.
- For good memory management, logical address space is bound with a separate physical address space.

5.1.5 Swapping

- Keeping all processes in memory all the time requires a large amount of main memory. For execution of the process, it must be swapped.
- **Swapping** is method of temporarily removing inactive process/programs from the physical memory.
- Inactive program means, which is neither executing on the CPU, nor performing an I/O operation.
- Fig. 5.1.4 shows swapping.

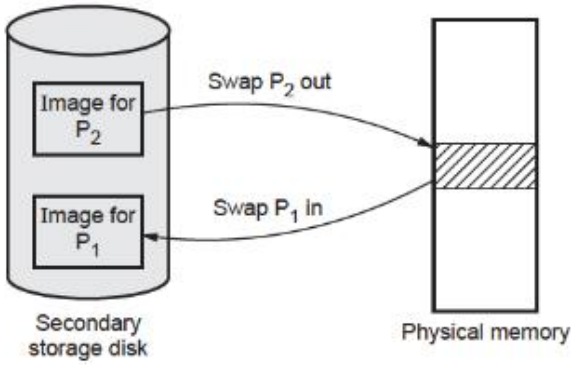


Fig. 5.1.4 Swapping

- When process P_2 send requests for an I/O operation, OS blocked P_2 process and will not return to the ready state for long period of time. Process P_2 is in blocked state. Process manager place the process P_2 in blocked state and inform to the memory manager regarding this action.
- Process manager moves a blocked process P_1 into the ready state and inform to memory manager. Process P_1 is loaded into main memory by memory manager when there is space available.
- Swapped out process is stored on the secondary storage disk. It contains executable image with code, data and stack.
- Relocation hardware supports for swapping. Because of address binding problem, swapping is difficult without relocation hardware.
- Round robin and priority based CPU scheduling algorithm uses the swapping concept.
- Time sharing system uses swapping. In time sharing, kernel can estimate when a program is likely to be scheduled next.

- A swapped out process will occupy same memory location when process swap in by OS. But this totally depends on address binding. If address binding is load time or assembly time, then swapped out process uses same memory location when swap-in. But execution-time binding uses different memory location.
- Secondary storage is required to implement the swapping technique.
- Content switching time increases when swapping increases. Major part of the swap time is transfer time. Swapping increases the operating system overhead due to the disk I/O involved.
- When swapping creates multiple holes in memory, it is possible to combine them all into one by moving all the processes to one side of memory. This technique is known as **memory compaction.**
- Compaction requires a lot of CPU time, so most operating systems not used this technique.
- Compaction means movement of processes in the memory during their execution. It is sometimes called dynamic relocation of a program.

Board Questions

1. *Write in short on basic memory management.* **MSBTE : Summer-16, Marks 6**
2. *Define swapping ? When it is used ?* **MSBTE : Winter-16, Marks 4**
3. *With neat diagram, explain file access methods.* **MSBTE : Summer-17, Marks 4**
4. *Explain major activities of memory management component of an operating system.* **MSBTE : Summer-18, Marks 4**
5. *Explain swapping in operating system with diagram and example.* **MSBTE : Summer-18, Marks 8**
6. *Explain memory management in detail.* **MSBTE : Winter-18, Marks 4**
7. *Differentiate between contiguous and linked memory allocation method.* **MSBTE : Winter-18, Marks 4**

5.2 Fixed Partitioning

- In a fixed size partitioning of the main memory all partitions are of the same size. The operating system divides main memory into a number of fixed size partition. Each partition holds a single program.
- Fixed sized partitions are relatively simple to implement. Degree of multiprogramming depends on the number of partitions.
- Normally main memory is divided into two partitions :
  1. For resident program 2. For user processes
- Batch operating system uses the fixed size partition scheme. The operating system keeps the record of memory allocation and deallocation in the form of table. Initially all the memory is available for user processes.
- When the process arrives in the system and needs memory, operating system search for large hole for this process. Hole is one large block of free available memory. If any free hole is found, process is allocated to the free hole of memory as is needed.
- After allocating number of holes for the processes, a set of various size holes is scattered throughout memory at any given time. When a process arrives and searches for (memory) set of holes. But the holes must be large enough to accomodate the process.
- Fixed sized partitions are simple to implement. Any process whose size is less than or equal to the partition size can be loaded into any available partition.
- Fig. 5.2.1 shows an example of fixed partitioning of a memory.

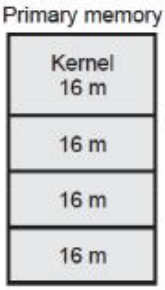


Fig. 5.2.1 Fixed size partition

- There are two difficulties with the use of equal size fixed partitions :
  1. A program may be too big to fit into a partition.
  2. Memory utilization is extremely inefficient.

- First difficulty is solved by using overlays. Overlays involves moving data and program segments in and out of memory.

Advantages and disadvantages of fixed partition size

1) Advantages

1. Simple to implement.
2. Does not require expertise to understand and use such a system.
3. Less overhead.

2) Disadvantages

1. Memory is not fully utilized.
2. Poor utilization of processors.
3. User's process being limited to the size of available main memory.
4. Requires contiguous loading of entire program.

5.2.1 Dynamic Memory Partitions

- Memory partitions are of variable length and number. Processes are loaded into the size nearest to its requirements.
- Fig. 5.2.2 shows the effect of dynamic memory partitioning. Initially in main memory, only OS is loaded and rest of the memory is free. Memory size is 64 Mbytes. How the process is loaded with dynamic partitions is shown below. After loading all the process some free space remains at one side of the memory.
- Four process/programs are loaded into memory. These program starts loading where the OS ends. After loading four programs, it leaves a hole at the end of memory that is too small for a next process.
- Operating system swap out program 2 and leaves sufficient room to load new program into memory. The size of newly loaded program is smaller than the program in memory. Again another small hole is created.

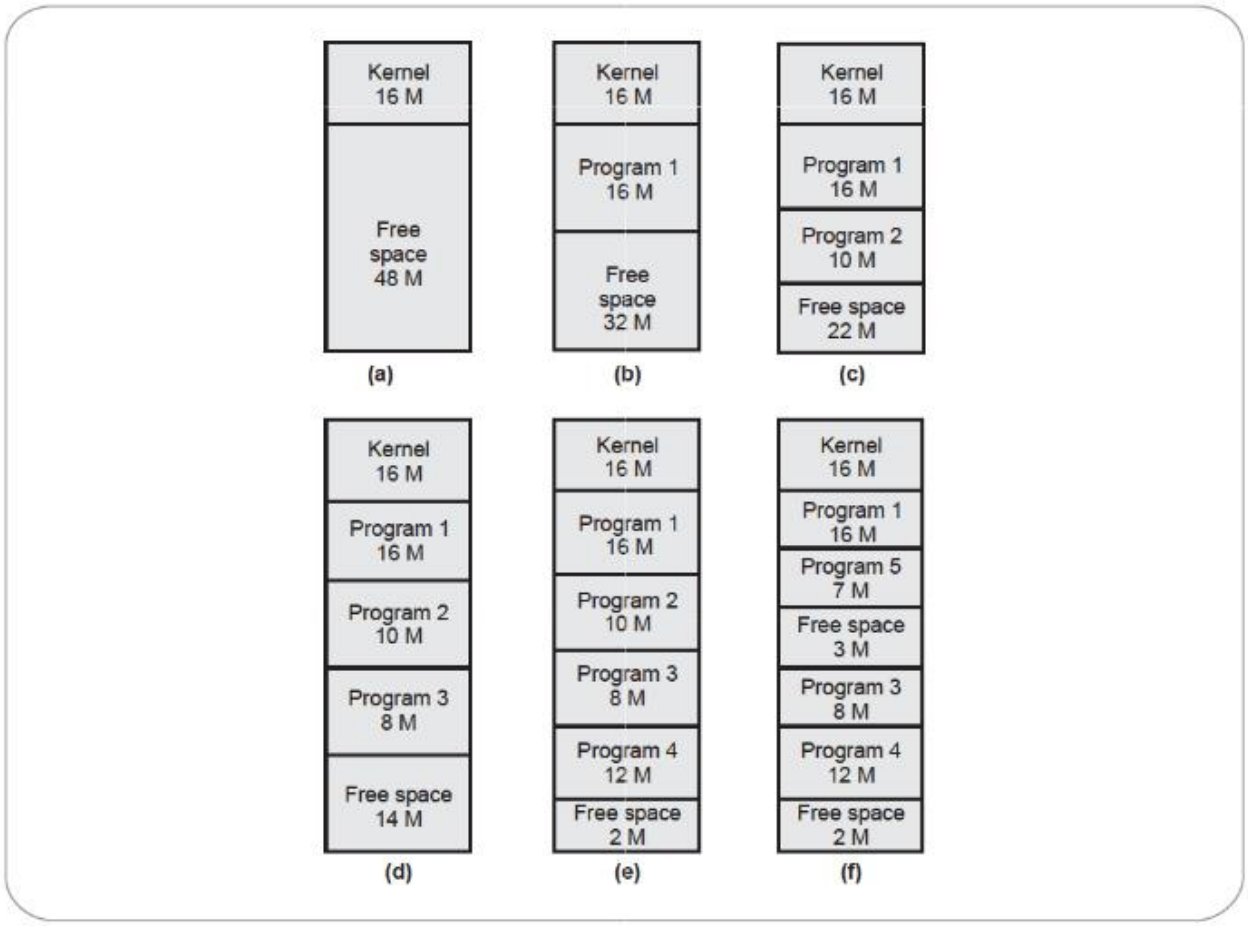


Fig. 5.2.2 Dynamic partitioning example

5.2.2 Memory Protection and Mapping

- Memory allocated to the process need to be protected against interference by other processes. Memory protection provides this facility.
- Relocation register and limit register are used for memory protection. Physical address is stored in the relocation register and range of virtual address is stored in the limit register.
- Fig. 5.2.3 shows logical address space.

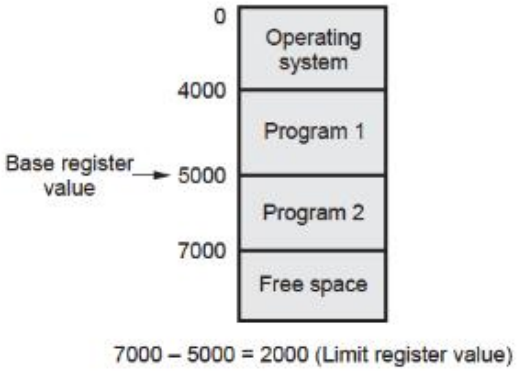


Fig. 5.2.3 Logical address space

- Use of registers method for memory protection requires proper arrangement. This method has to be used individually for each memory area allocated to a process.
- Every logical address generated by the CPU is checked against these registers. A protection violation interrupt is raised if any of these checks fail.
- Operating system load a base register and limit register. OS uses privileged instruction for this purpose. Operating system executes in monitor mode and privilege instruction also executes in privilege mode.
- Fig. 5.2.4 shows memory protection by using relocation and limit registers.
- Operating system prevents the user programs from changing the content of the registers. While executing in the monitor mode, operating system is given unrestricted access to both monitor and user's memory.

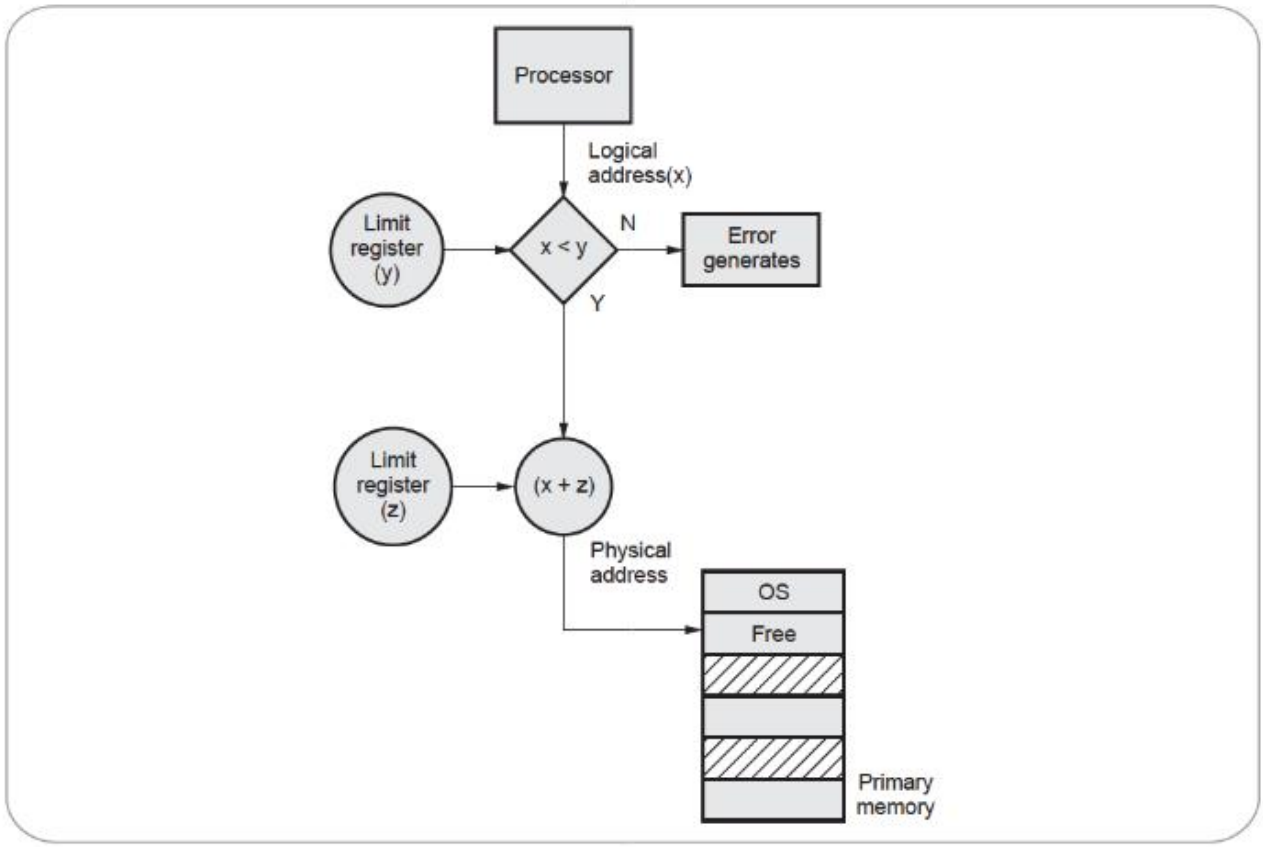


Fig. 5.2.4 Memory protection hardware

- In a multiprogramming environment, protection of main memory is essential. Paging or segmentation or both in combination provides an effective means of managing main memory.
- An example of the hardware support that can be provided for memory protection is that of the IBM system/370 family of machines, on which VMS runs. Microsoft Windows 3.1/95 offer memory protection. Microsoft WinNT also offers memory protection. In UNIX, almost impossible to corrupt another process memory.

Board Questions

1. *With suitable diagram explain contiguous allocation method.* **MSBTE : Summer-15, Marks 4**
2. *Describe the contiguous allocation method for file, state any two merits and demerits.* **MSBTE : Summer-16, Marks 6**
3. *Give difference between : External fragmentation and Internal fragmentation (four points)* **MSBTE : Summer-18, Marks 4**

5.3 Variable Partitioning

- The use of unequal size partitions provides a degree of flexibility to fixed partitioning. In dynamic partitioning, the partitions are of variable length and number.
- In noncontiguous memory allocation, a program is divided into blocks that the system may place in nonadjacent slots in main memory.
- This allocation method do not suffer from internal fragmentation, because a process partition is exactly the size of the process.
- Fig. 5.3.1 shows noncontiguous memory allocation method. Operating system maintain the table which contains the memory areas allocated to process and free memory. Memory management unit use this information for allocating processes.
- Table contains information about memory starting address for process/program and their size.
- CPU sends the logical address of the process to the MMU and the MMU uses the memory allocation information stored in the table for calculating logical address. We called this address as effective memory address of the data/instruction.

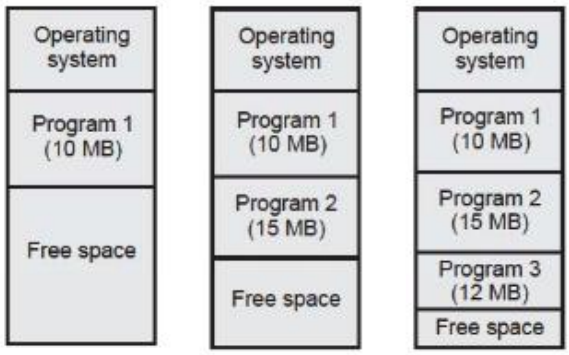


Fig. 5.3.1 (a) Variable size partition

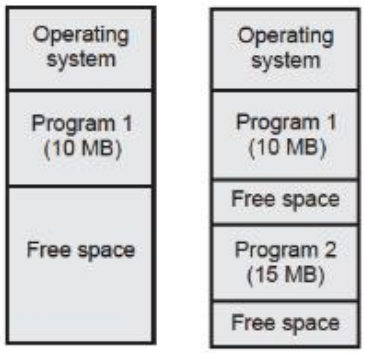


Fig. 5.3.1 (b) Noncontiguous memory allocation

5.3.1 Difference between Contiguous and Noncontiguous Memory Allocation

Sr. No.	Contiguous allocation	Noncontiguous allocation
1.	Program execution take place without overhead.	Address translation is overhead.
2.	Swapped-in processes are placed in the original area.	Swapped-in processes can be placed any where in memory.
3.	Suffer from internal fragmentation.	Only paging, suffers from internal fragmentation.
4.	Allocates single area of memory for process.	Allocates more than one block of memory for process.
5.	Wastage of memory.	No wastage of memory.

5.3.2 Fragmentation

- Fragmentation are of two types :
  1. Internal fragmentation.
  2. External fragmentation.
- In fragmentation, OS cannot use certain area of available memory.

Internal fragmentation

- There is wasted space internal to a partition due to the fact that the block of data loaded is smaller than the partition is called as internal fragmentation.
- Fig. 5.3.2 shows an internal fragmentation.

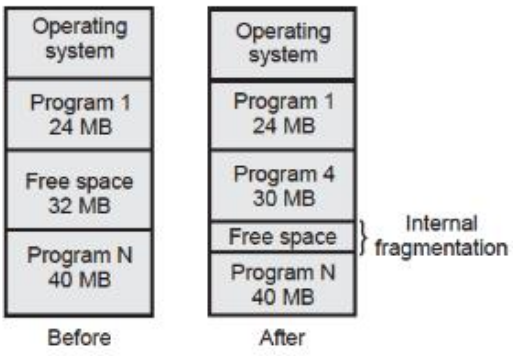


Fig. 5.3.2 Internal fragmentation

- To keep track of this free hole is overhead for system.

External fragmentation

- It occurs when enough total main memory space exists to satisfy a request, but it is not contiguous, storage is fragmentated into a large number of small holes.
- Fig. 5.3.3 shows external fragmentation.

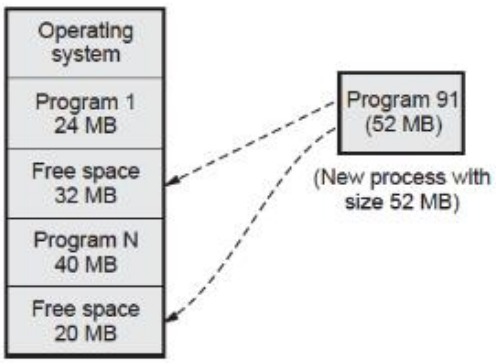


Fig. 5.3.3 External fragmentation

- Following are the solution for external fragmentation :
  1. Compaction
  2. Logical address space of a process/program

5.3.3 Difference between Internal and External Fragmentation

Internal Fragmentation	External Fragmentation
Internal fragmentation is the area occupied by a process but cannot be used by the process.	External fragmentation exists when total free memory is enough for the new process but it's not contiguous and can't satisfy the request.
First fit and best fit memory allocation does not suffer from internal fragmentation.	First fit and best fit memory allocation suffers from external fragmentation.
In fixed partitioning, inefficient use of memory due to internal fragmentation.	In dynamic partitioning, inefficient use of processor due to need for compaction to counter external fragmentation.
Paging's suffer from internal fragmentation.	Paging does not suffer from external fragmentation
Segmentation does not suffer from internal fragmentation.	Segmentation suffer from external fragmentation

5.3.4 Compaction

- Compaction solves problem of external fragmentation. Fig. 5.3.4 shows compaction.

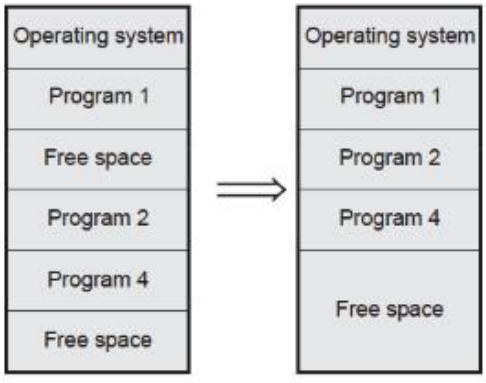


Fig. 5.3.4 Compaction

- Operating system moves all the free holes to one side of main memory and creates large block of free size.
- It must be performed on each new allocation of process to memory or completion of process for memory. System must also maintain relocation information.
- All free blocks are brought together as one large block of free space. Compaction requires dynamic relocation.
- Compaction has a cost and selection of an optimal compaction strategy is difficult. One method for compaction is swapping out those processes that are to be moved within the memory and swapping them into different memory locations.
- Compaction is not always possible. Compaction is time consuming method and wastage of the CPU time.

Garbage collection

- Some programs use dynamic data structures. These programs dynamically use and discard memory space. Technically, the deleted data items release memory locations.
- But, in practice the OS does not collect such free space immediately for allocation. This is because that affects performance. Such areas, therefore are called **garbage.**
- When such garbage exceeds a certain threshold the OS would not have enough memory available for any further allocation.

5.3.5 Placement Algorithms

- In an environment that supports dynamic memory allocation, the memory manager must keep a record of the usage of each allocatable block of memory. This record could be kept by using almost any data structure that implements linked lists.
- An obvious implementation is to define a free list of block descriptors, with each descriptor containing a pointer to the next descriptor, a pointer to the block and the length of the block.
- The memory manager keeps a free list pointer and inserts entries into the list in some order conductive to its allocation strategy. A number of strategies are used to allocate space to the processes that are competing for memory.
- Fig. 5.3.5 shows the placement algorithm.

1. **Best fit** : The allocator places a process in the smallest block of unallocated memory in which it will fit. For example, suppose a process requests 12 kB of memory and the memory manager currently has a list of unallocated blocks of 6 kB,

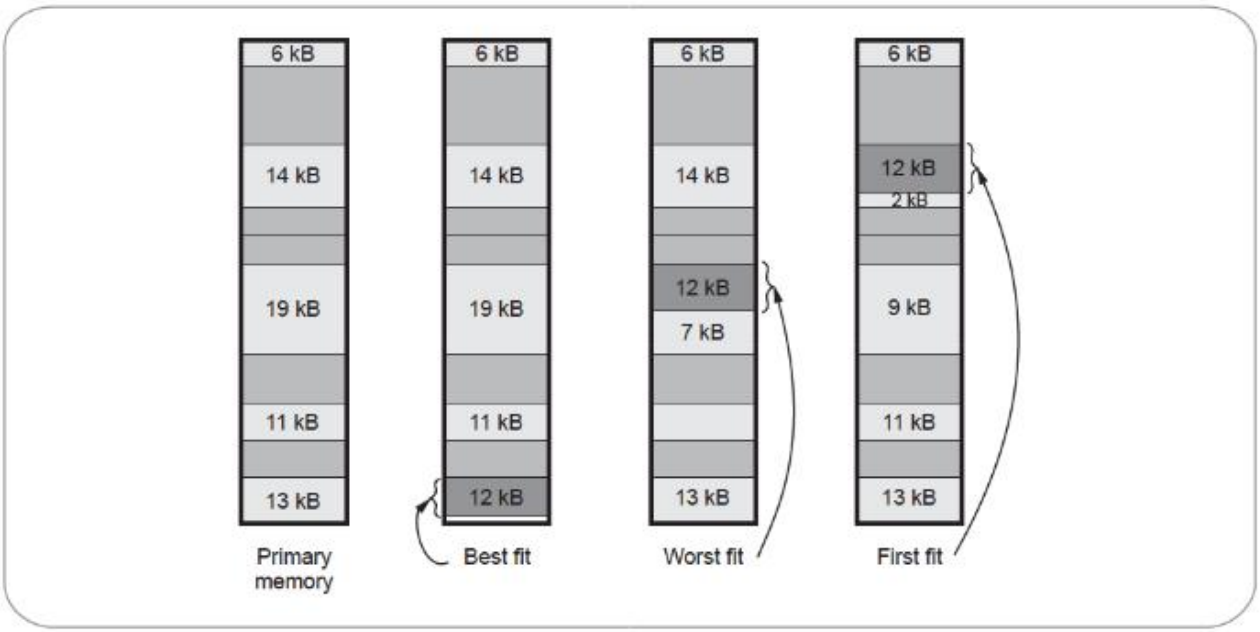


Fig. 5.3.5 Placement algorithm

14 kB, 19 kB, 11 kB and 13 kB blocks. The best-fit strategy will allocate 12 kB of the 13 kB block to the process.

- Best fit chooses the block that is closest in the size to the request.

Problems of best fit

- It requires an expensive search of the entire free list to find the best hole.
- More importantly, it leads to the creation of lots of little holes that are not big enough to satisfy any requests. This situation is called *fragmentation* and is a problem for all memory-management strategies, although it is particularly bad for best-fit.

  Solution : One way to avoid making little holes is to give the client a bigger block than it asked for. For example, we might round all requests up to the next larger multiple of 64 bytes. That doesn't make the fragmentation go away, it just hides it. Unusable space in the form of holes is called *external fragmentation.*

2. **Worst fit :** The memory manager places a process in the largest block of unallocated memory available. The idea is that this placement will create the largest hold after the allocations, thus increasing the possibility that compared to best fit ; another process can use the remaining space. Using the same example as above, worst fit will allocate 12 kB of the 19 kB block to the process, leaving a 7 kB block for future use.
3. **First fit :** There may be many holes in the memory, so the operating system, to reduce the amount of time it spends analyzing the available spaces, begins at the start of primary memory and allocates memory from the first hole it encounters large enough to satisfy the request. Using the same example as above, first fit will allocate 12 kB of the 14 kB block to the process.
4. **Next fit :** The first fit approach tends to fragment the blocks near the beginning of the list without considering blocks further down the list. Next fit is a variant of the first-fit strategy. The problem of small holes accumulating is solved with next fit algorithm, which starts each search where the last one left off, wrapping around to the beginning when the end of the list is reached (a form of one-way elevator).

- Notice in the diagram above that the best fit and first fit strategies both leave a tiny segment of memory unallocated just beyond the new process.

Ex. 5.3.1 *Given memory partitions of 100 K, 500 K, 200 K, 300 K and 600 K (in order) how would each of the first-fit, best-fit and worst-fit algorithms place processes of 212 K, 417 K, 112 K and 426 K (in order) ? Which algorithm makes the most efficient use of memory ?*

Sol. : First-fit :

212 K is put in 500 K partition

417 K is put in 600 K partition

112 K is put in 288 K partition

(new partition 288 K = 500 K – 212 K)

426 K must wait.

Best-fit :

212 K is put in 300 K partition

417 K is put in 500 K partition

112 K is put in 200 K partition

426 K is put in 600 K partition

Worst-fit :

212 K is put in 600 K partition

417 K is put in 500 K partition

112 K is put in 388 K partition (600 K – 212 K)

426 K must wait

In this example, Best-fit turns out to be the best.

Board Questions

1. *Explain static and dynamic memory partioning with advantages and drawback.*
   MSBTE : Summer-15, Marks 4
2. *Explain the concept of variable memory partitioning with example.*
   MSBTE : Winter-16, Marks 6
3. *Explain static and dynamic memory partitioning method.*
   MSBTE : Winter-18, Marks 4

5.4 Free Space Management Techniques

- Space that is allocated to files must be managed and space which is not allocated to any file must be

managed. To keep track of free disk space, the system maintains a free space list. File allocation table and disk allocation both are required to manage free space properly.

- When a file is deleted, its disk space is added to the free-space list.
- Following are the techniques used for free space management :
  1. Bit tables or bit vector
  2. Chained free portions or linked list
  3. Indexing or grouping
  4. Free block list or counting

1. Bit tables or bit vector

- Each block on the disk is represented by bit. It uses vector chain for blocks. Free block is represented by o and used block represented by 1.
- For example, consider a disk where blocks 3, 4, 5, 7, 9, 14, 15, 19, 30, 31, 32, 35, 36, 37, 38 are free blocks and rest of the blocks are allocated. The free space is shown below :

11100010101111001110111111111000110000

- The memory required for a block bitmap = Disk size / 8 × (block size of file system)
- Bit map requires extra space. For example :

$$\text{Block size} = 2^{12} \text{ bytes}$$

$$\text{Disk size} = 2^{30} \text{ bytes}$$

$$\text{Block number (n)} = \frac{\text{Block size}}{\text{Disk size}} = \frac{230 \text{ bytes}}{212 \text{ bytes}}$$

$$= 2^{18} \text{ bytes}$$

- When bit table is stored into the memory, then exhaustive search of the table can slow file system performance. Most of the file system use auxiliary data structure for bit table. File system also maintain summary table. Summary table contains sub-range, number of free blocks and maximum sized contiguous number of free blocks.
- Summary table is used to store information about contiguous free blocks. Whenever file system needs a number of contiguous blocks, it can scan the summary table to find an appropriate sub-range and then search that sub-range.
- This method is used in Apple Macintosh.

Advantage :

a. Easy to find a free blocks

b. It is as small as possible.

Disadvantage :

It may not be feasible to keep the bitmap in memory for large disks.

2. Link list

- In this method, all free space disk blocks are linked, keeping a pointer to the first free block. All file allocation methods used link list free space techniques.
- There is small space overhead because there is no need for a disk allocation table. In the above example, free blocks are 3, 4, 5, 7, 9, 14, 15, 19, 30, 31, 32, 35, 36, 37, 38. Here free block pointer is on block number 3. Block 3 contains pointer to block 4, block 4 contains pointer to block 5 and block 5 contains pointer to block 7 and so on.
- This method is not efficient because to reach free block 9, we have to traverse block 3, 4, 5 and 7 then we will reach to block 9.
- Disk will become quite fragmented after some use. Many portions will be a single block long.

3. Grouping

- First free block contains the addresses of n free blocks. The first n–1 of these blocks is actually free. The last block also contains the address of another n free block.
- Consider the free blocks : 3, 4, 5, 7, 9, 14, 15, 19, 30, 31, 32, 35, 36, 37, 38
  Then

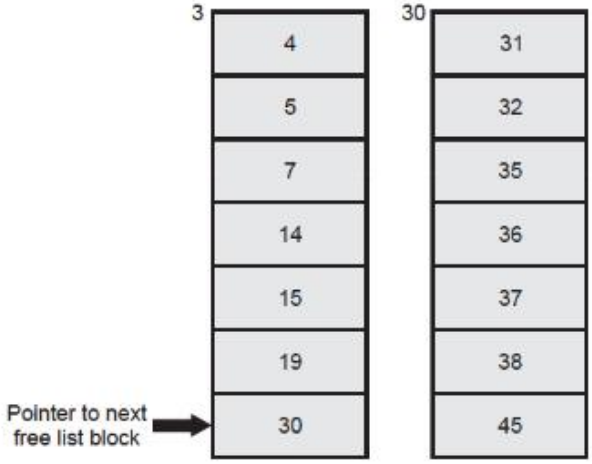


Fig. 5.4.1

- When all the blocks in the group have been allocated, then use the block that held the pointer.
- Because of grouping method, address of a large number of free blocks can be found quickly.

4. Counting

- It keeps the address of the first free block and the number *n* of free contiguous blocks that follow the first block. Each entry in the free space list then consists of a disk address and a count.

Board Questions

1. *Explain Bit map free-space management technique.*

MSBTE : Summer-17, Marks 4

2. *Explain "Bitmap" method in free space management technique.*

MSBTE : Summer-18, Marks 4

5.5 Virtual Memory

- Virtual memory is a method of using hard disk space to provide extra memory. It simulates additional main memory.
- In Windows operating system, the amount of virtual memory available equals the amount of free main memory plus the amount of disk space allocated to the swap file.
- Fig. 4.8.1 shows logical view of virtual memory concept.

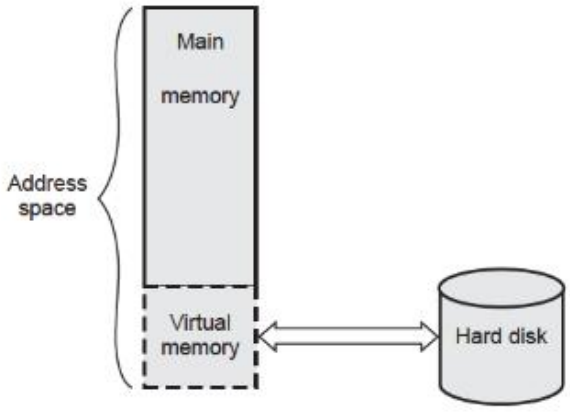


Fig. 5.5.1 Logical view of virtual memory

- A swap file is an area of your hard disk that is set aside for virtual memory. Swap files can be either temporary or permanent.
- Virtual memory is stored in the secondary storage device. It helps to extend additional memory capacity and work with primary memory to load applications. The virtual memory will reduce the cost of expanding the capacity of physical memory. The implementations of virtual memory will different for operating systems to operating system.
- Each process address space is partitioned into parts that can be loaded into primary memory when they are needed and written back to secondary storage otherwise. Address space partitions have been used for the code, data and stack identified by the compiler and relocating hardware. The portion of the process that is actually in main memory at any time is defined to be the **resident set** of the process.
- The logical addressable space is referred to as virtual memory. The virtual address space is much larger than the physical primary memory in a computer system. The virtual memory works with the help of secondary storage device and its speed is low compared to the physical storage location.
- Virtual memory uses two types of addresses : ***Virtual address and physical address.***
- Virtual address is referred by process and physical address is actual address of the main memory.
- Whenever a process accesses a virtual address, the system must translate it to a physical address. Virtual memory system uses special purpose hardware. This hardware is called as memory management unit.
- Most virtual memory system use a technique called paging. The virtual address space is divided into fixed size units called pages. The corresponding units in the physical memory are called page frames. The pages and page frames are normally the same size. The area on a hard disk that stores page frames is usually called the paging file or the swap file.
- Fig. 5.5.2 shows concept of virtual memory.
- When the system is ready to run a process, the system loads the process code and data from secondary storage into primary memory. Only a small portion of these needs to be in primary memory at once for the process to execute. The success of implementing virtual memory system is how to map virtual addresses to physical addresses. Because processes reference only virtual addresses.

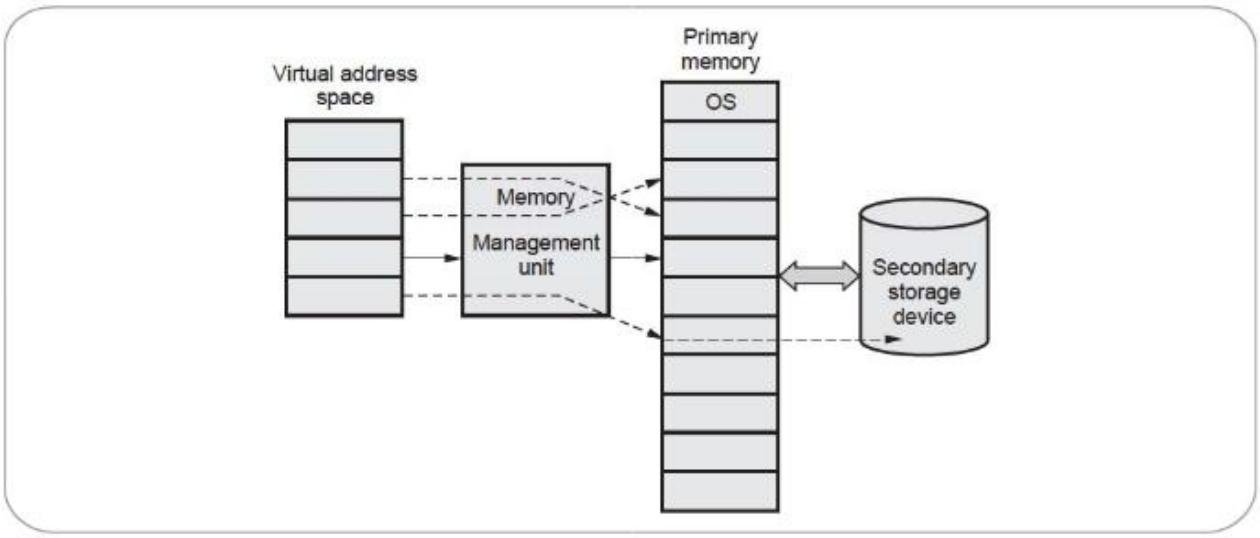


Fig. 5.5.2 Concept of virtual memory

- Virtual memory can be implement in one of two ways :
  1. Paging 2. Segmentation

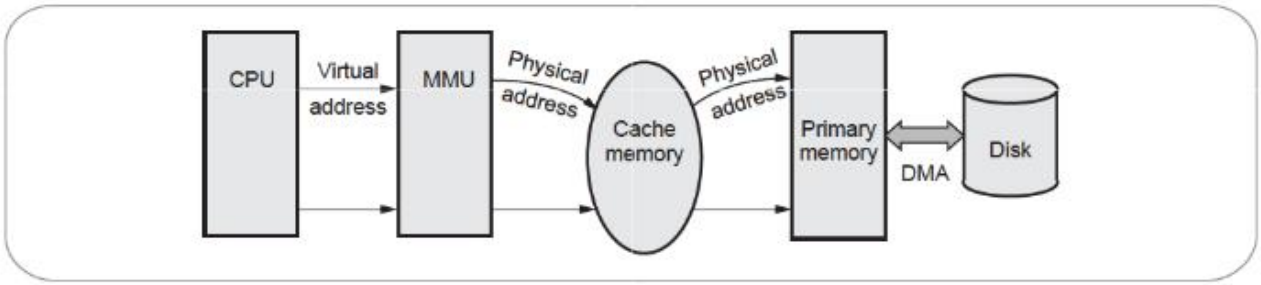


Fig. 5.5.3 Virtual memory organization

- Following are the situations, when entire program is not required to load fully.

1. User written error handling routines are used only when an error occurs in the data or computation.
2. Certain options and features of a program may be used rarely.
3. Many tables are assigned a fixed amount of address space even though only a small amount of the table is actually used.
4. Many routines are commonly used at mutually exclusive times during a run.

- Virtual memory makes the task of programming much easier. Virtual memory is commonly implemented by **demand paging**. Virtual memory mechanism bridges the size and speed gaps between the main memory and secondary storage and is usually implemented in part by software technique.

Benefits

1. Less I/O needed to load/swap a process.
2. Degree of multiprogramming can be increased.
3. A program's logical address space can be larger than physical memory.

Board Questions

1. *What is virtual memory ? Explain paging and page fault.* **MSBTE : Winter-15, Marks 4**
2. *Describe concept of virtual memory with suitable example.* **MSBTE : Summer-16, Marks 4**
3. *Describe the concept of virtual memory with suitable example.* **MSBTE : Winter-17, Marks 4**
4. *Describe virtual memory management.* **MSBTE : Winter-18, Marks 4**

5.6 Paging

- In paging, operating system divides each incoming programs into pages of equal size. The sections of a disk are called block or sectors. The sections of main memory are called page frames. One sector will hold one page of job instructions and fit into one page frame of memory.
  - In paging, logical address space of a program can be noncontiguous. It solves external fragmentation problem.
  - The relation between virtual addresses and physical memory addresses given by page table.
- Fixed sized blocks are called frames and breaking of logical memory into blocks of same size called pages.
- Memory manager prepares following things before executing a program :
  1. Find out the number of pages in the program.
  2. Free space in the main memory.
  3. Loading of all the programs pages into memory.
- Pages are not loaded continuosly in main memory. Each page can be stored in any available page frame anywhere in main memory. Memory manager keeps the track of pages of program. Paging avoids external fragmentation and need for compaction.
- Memory manager uses three tables to keep track of process and memory.
  1. **Job table :** It stores the size of the active job and memory location where its page table is stored.
  2. **Page map table :** It contains vital information about each page. PMT contains apage number and corresponding page frame memory address. It includes only one entry per page. Page numbers are in sequential order.
  3. **Memory map table :** MMT is used to store page frame location and its status.
- Page size depends upon underlying hardware. Operating system maintains a page table for each process. The page table shows the frame location for each page of a process. Fig. 5.6.1 shows the virtual address format for paging system.

Fig. 5.6.1 Virtual address format

- Processor hardware performs the logical to physical address translation. Logical address contains page number and offset. The physical address contains frame number and offset. Offset is a relative factor. It is used to locate that line within its page frame.
- Fig. 5.6.2 shows address translation.

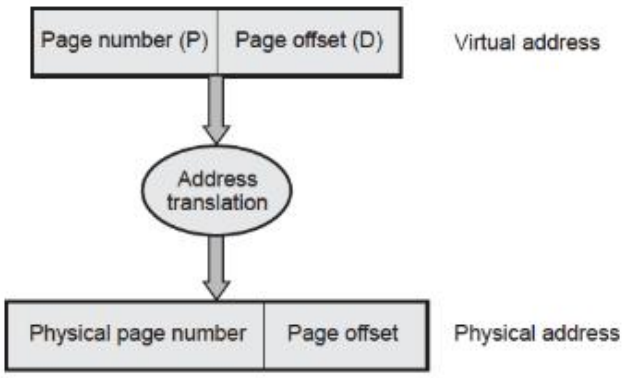


Fig. 5.6.2 Address translation

- Suppose a system uses n-bit for representing both physical and virtual address. The page number is represented by the most significant bit (n - m bits) and the displacement is represented by m bit.
- The table, which holds virtual address to physical address translations is called page table. As displacement is constant, so only translation of virtual page number to physical page is required.

- Fig. 5.6.3 shows paging hardware. CPU generates logical addresses containing page number and an offset. This page number is used to retrieve the frame number from a page table which gives the base address so the physical address is calculated as base + offset.
- Paging implementation requires CPU and operating system support. Page table may itself be resident in main memory.
- Active process is loaded into memory and it occupies number of pages. This set of pages forms its resident set. Hardware defined the page size.
- The processor generates virtual (logical) address and it consists of two parts :

  1. Page number 2. Page offset
- Page table contains page number and index frame number. Page number is used as an index into a page table.
- In case of simple partition, a logical address is the location of a word relative to the beginning of the program the processor translates that into a real address.
- But in the paging method the logical to real address translation is still done by processor hardware. The processor must know how to access the page table of the current process.
- The operating system partitions the memory into areas called **page frames.** Each page frame size is equal. Page size is normally power of 2.
- Fig. 5.6.4 shows a paging model. Operating system maintains the free frame list. During execution of process the memory management unit consults the page table to perform address translation.

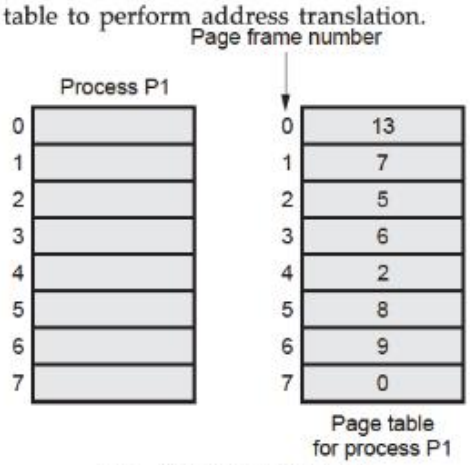


Fig. 5.6.4 Paging model

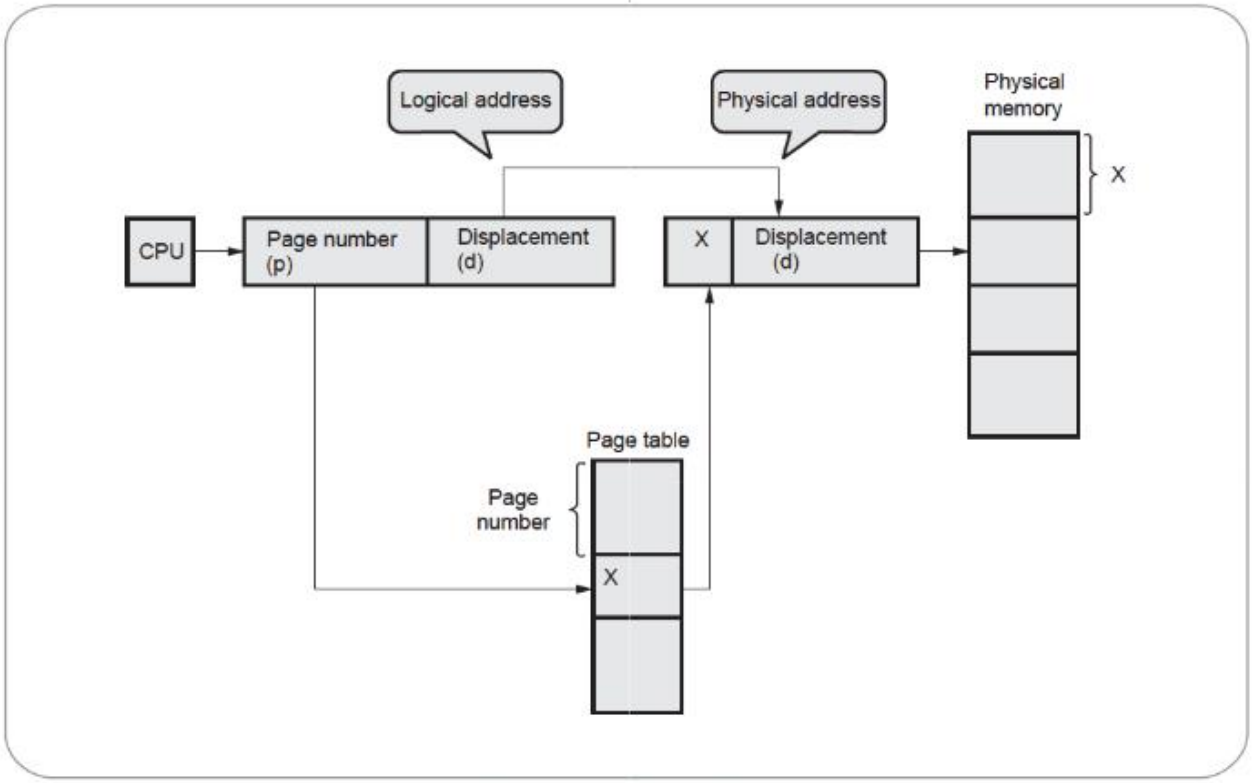


Fig. 5.6.3 Paging hardware

- The physical address is the portion of the primary memory allocated to the process. The page frames allocated to the process need not be contiguous.
- The aim of the paging system is to identify the set of pages needed for the process current locality and then to load only those pages into page frames in primary memory.
- The logic of the address translation process in paged system is shown in the following Fig. 5.6.5.
- For example, the virtual address is 03200H. This virtual address is split by hardware into the page number and offset within that page. High order 12 bit is used as page number i.e. 003H and lower order 12 bit is used for the offset i.e. (200H).
- Page number is used to index the page table and to obtain the corresponding physical frame number (FFFH). This value is then concatenated with the offset to produce the physical address (FFF200H) which is used to reference the target item in memory.
- The operating system keeps track of the status of each page frame by means of a physical memory map that may be structured as a static table. **The logical address space may be the same size as the physical address space or it may be smaller.**
- If logical address space is small, it prevents one process from monopolizing all of the memory. If the logical address space is larger than the physical address space, all pages could not be resident in physical memory. Simple paging has no capability for dealing with references to pages that are not in memory.

5.6.1 Protection and Sharing

- Protection bit is used for provide protection to the memory. Page table adds one more column for providing protection bit.
- One bit field define a page to be read only or read-write operation. This field is read at the time of calculating physical address of memory.
- Fig. 5.6.6 shows protection bit with page table.
- When valid bit is set, the required page is in the process logical address space and the page is valid. If invalid bit is set, the page is not in the process logical address space.

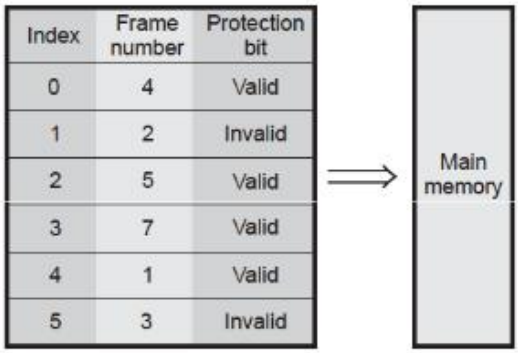


Fig. 5.6.6 Protection bit

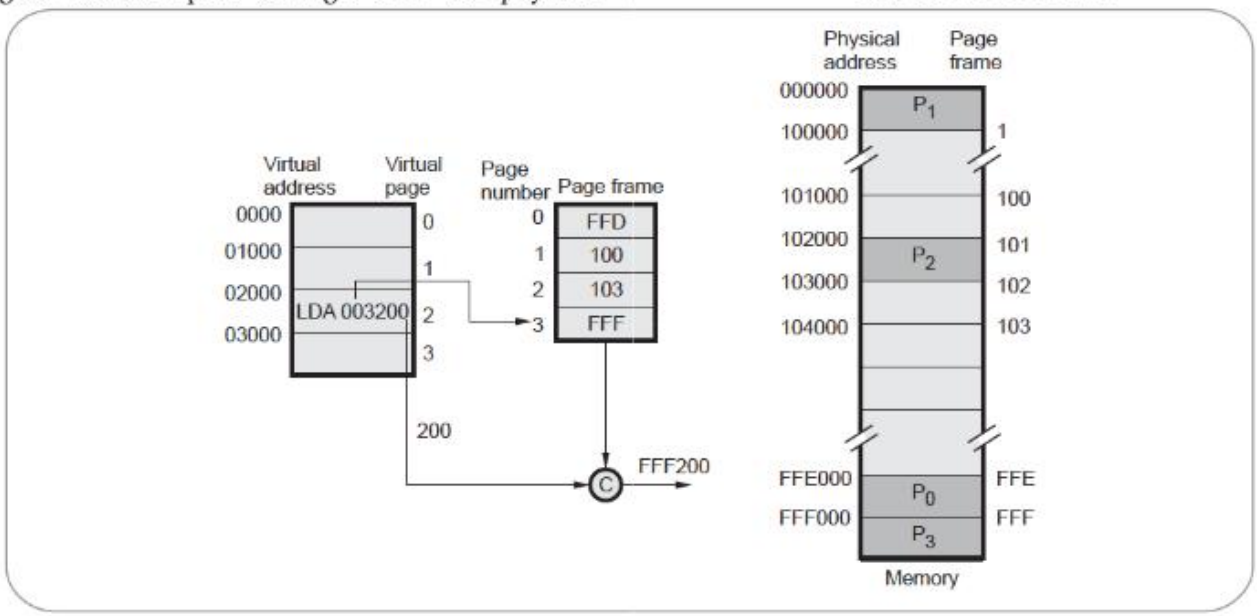


Fig. 5.6.5 (Virtual address to physical address) Paging

5.6.2 Paging with TLB

- Set of registers are used for implementing page table. Registers are suitable only for small page table. If page table size increases then special purpose hardware cache is used for page table.
- Special cache memory called a Translation Lookaside Buffer (TLB) is used with the address translation hardware. The full page table is kept in primary memory.
- TLB is very effective in improving the performance of page frame access. The cache buffer is implemented in a technology which is faster than the primary memory technology.
- Fig. 5.6.7 shows paging with TLB.
- The TLB contents may be controlled by the operating system or by hardware, depending on the architecture.
- When a page is first translated to a page frame, the map is read from primary memory into the TLB. The TLB entry contains the page number, page frame's physical address etc.
- The page mapping mechanism first tries to find page number in the TLB. If the TLB contains the page number, then it is called as **page hit** or **TLB hit.**
- If the TLB does not contain an entry for page p, then it called as page miss or TLB miss. In case of TLB miss, the operating system locates the page table entry in the primary memory, which increases execution time.
- The TLB is associative, high-speed memory. A translation buffer is used to store a few of the translation table entries. It is very fast, but only for a small number of entries. On each memory reference.
  1. First ask TLB if it knows about the page. If so, the reference proceeds fast.
  2. If TLB has no information for page, must go through page and segment table to get information. Reference takes a long time, but gives the info for this page to TLB so it will know it for next reference.
- The greatest performance improvement is achieved when the associative memory is significantly faster than the normal page table lookup and the hit ratio is high. The hit ratio is the ratio between accesses that find a match in the associative memory and those that do not.

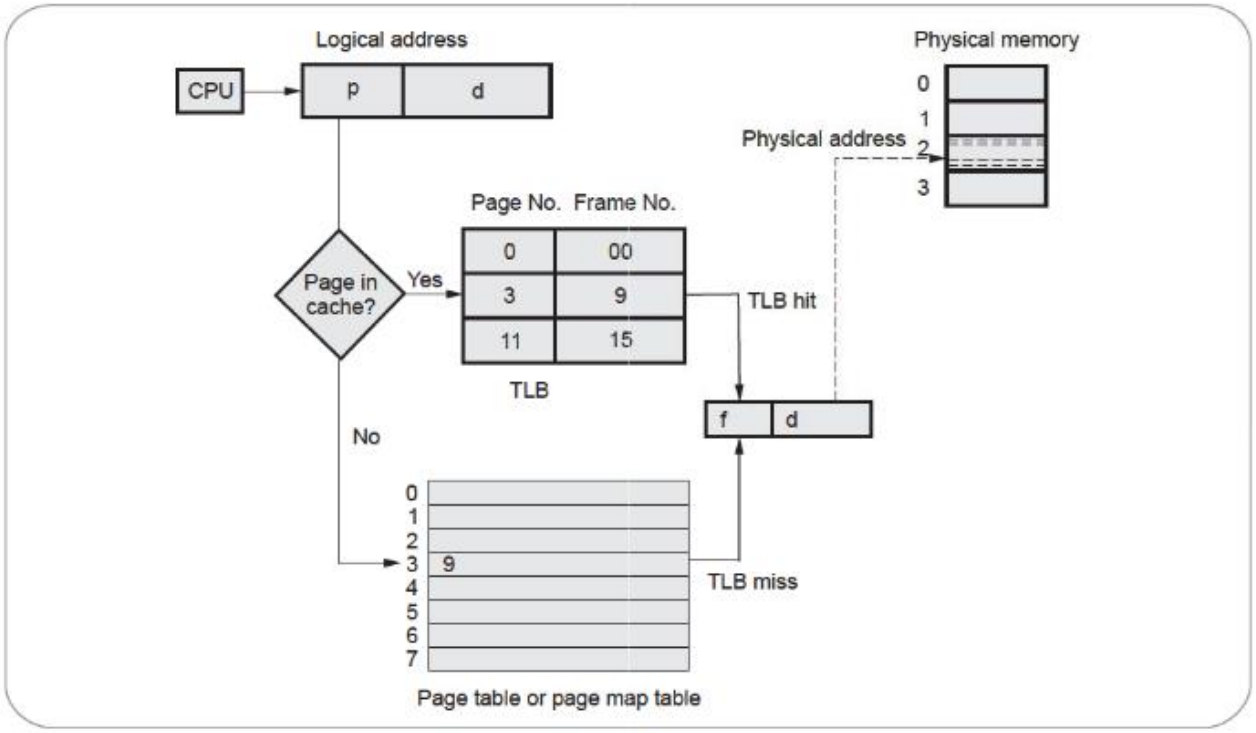


Fig. 5.6.7 Paging hardware with TLB

- **Disadvantage of TLB is that :** If two pages use the same entry of the memory, only one of them can be remembered at once. If process is referencing both pages at same time, TLB does not work very well.
- Each entry in the TLB must include the page number as well as the complete page table entry. The processor is equipped with hardware that allows it to simultaneously interrogate a number of TLB entries to determine if there is a match on page number. This technique is referred to as *associative mapping*.

5.6.3 Page Table Structure

5.6.3.1 Multilevel or Hierarchical Page Table

- Page tables can consume a significant amount of memory.
- For example : A 32-bit virtual address space using 4 kB pages.

$$\text{Page size} = 2^{12} \text{ bytes}$$

$$\text{Space for page numbers} = 2^{32} - 2^{12}$$

$$= 2^{20} \text{ bytes}$$

- So page table may consists of upto 1 million entries, one for each page, for a total address capacity of about four billion bytes.
- Hierarchical page table is also called as forward mapped page table. This approach is so effective that many modern operating systems employ it.
- In this method, each level containing a table that stores pointers to tables in the level below. Each table in the hierarchy is the size of one page. It enables the operating system to transfer page tables between main memory and secondary storage device easily.

For two levels of page table

- Virtual address contains page number and displacement into that page.

Virtual address (v) = (p, t, d)

where (p, t) = Page number

d = Displacement

- Here p is an index into the outer page table.
- Fig. 5.6.8 shows hierarchical page table.
- Intel IA - 32 architecture support two levels of page tables.

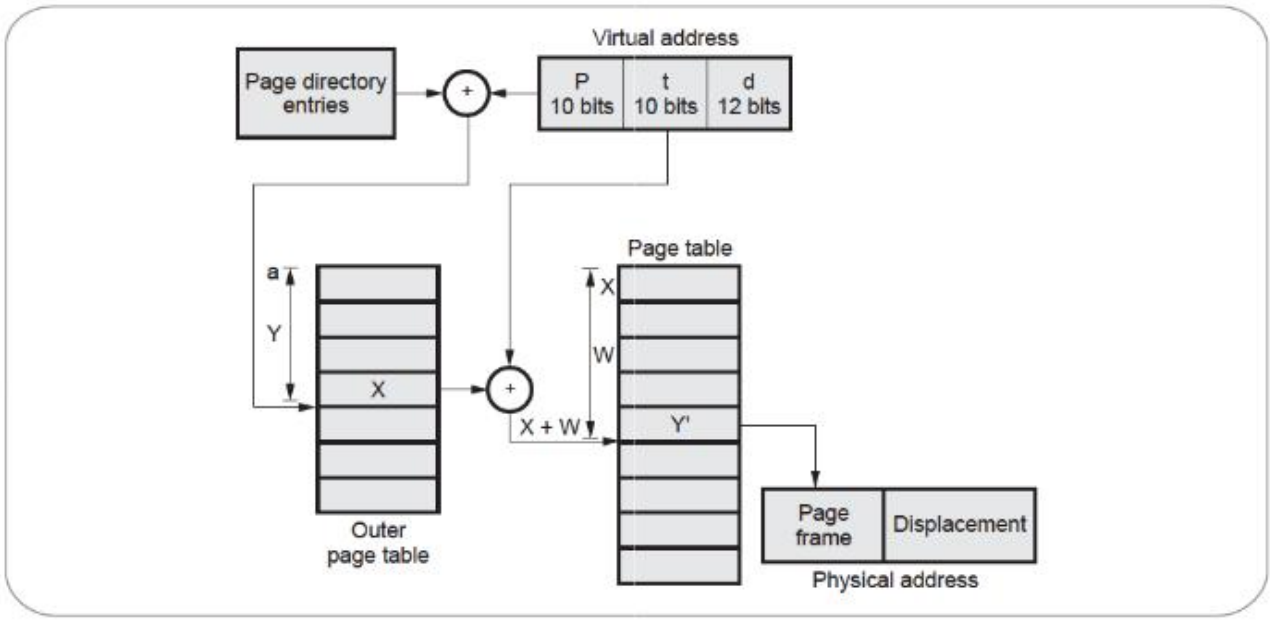


Fig. 5.6.8 Two level page table method

Advantages

1. It is compact and support sparse address space.
2. It easily manage the memory.
3. It reduce table fragmentation.

Disadvantages

1. Overhead due to one more page table.
2. On TLB miss, two loads from memory will be required to get the right address translation information from the page table.

5.6.3.2 Hashed Page Tables

- Hashed page table support address space of 32 bits and more. Hash value is used as virtual page number.
- Hash table contains a link list of elements. Each elements consists of following fields :
  1. Virtual page number
  2. Mapped page frame value.
  3. Pointer to the next element.
- Hash table improves search speed.
- Fig. 5.6.9 shows block diagram of hashed page table.

Working

1. Virtual page number is taken from virtual address.
2. Virtual page number is hashed into the hash table.
3. Virtual page number is compared with the first element of linked list.
4. Both the value is matched, that value is (i.e. page frame) used for calculating physical address.
5. If the both value is not matched, the entire linked list is searched for a matching.

- Clustered page table is same as hashed page table but only difference is that each entry in the hash table refers to several pages.

5.6.3.3 Inverted Page Table

- It uses a page table that contains an entry for each physical frame. This ensure that the page table occupies a fixed fraction of memory. The size is proportional to the physical memory.
- Page table overhead increases with address space size. Page table get too big to fit in memory.
- Inverted page table has one entry for each real page of memory. Lookup time is increased because it requires a search on the inverted table.

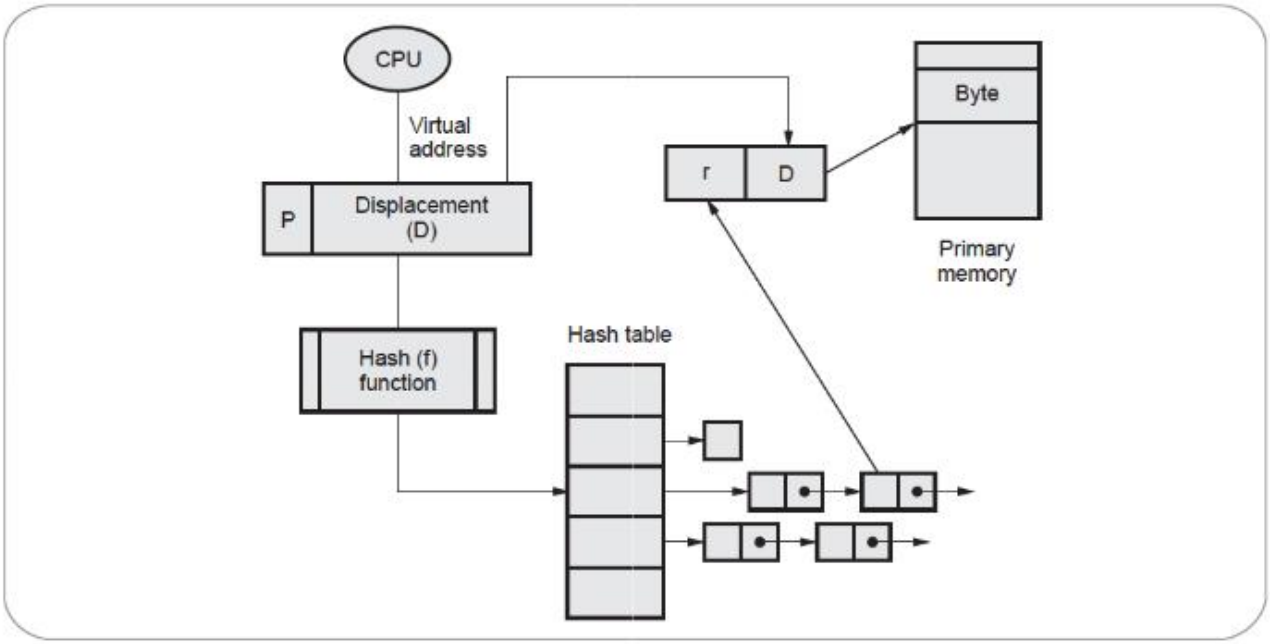


Fig. 5.6.9 Hashed page table

- Logical address is as follows

Process id	Page number	Offset

- Inverted page table is used by Itanium, Power PC and Ultra SPARC.

5.6.4 Advantages of Paging

1. Paging eliminates fragmentation.
2. Support higher degree of multiprogramming.
3. Paging increases memory and processor utilization.
4. Compaction overhead required for the relocatable partition scheme is also eliminated.

5.6.5 Disadvantages of Paging

1. Page address mapping hardware usually increases the cost of the computer.
2. Memory must be used to store the various tables like page table, memory map table etc.
3. Some memory will still be unused if the number of available block is not sufficient for the address spaces of the jobs to be run.

5.7 Segmentation

- In segmentation, a program's data and instructions are divided into blocks called **segments.** A segment is a logical entity in a program.
- Fig. 5.7.1 shows programmer's view.

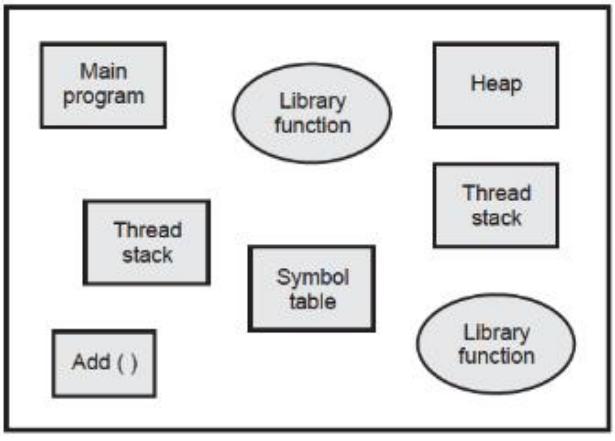


Fig. 5.7.1 Programmer's view

- **Logical view** : A process consists of a set of segments.
- **Physical view** : It consists of non-adjacent areas of memory allocated to segments.
- All segment size may be equal or may not be equal. Segmentation support user view of memory. Fig. 5.7.1 shows programmer's view of a program.
- Segmentation can be implemented with or without paging.
- Collection of segment is called as logical address space. Each segment is identified by its name.
- Processor generates logical addresses. These addresses consists of a segment number and an offset into the segment. Segment number is used as an index to segment page table.
- Fig. 5.7.2 shows logical address.

Segment number	Offset

Fig. 5.7.2 Logical address

- Segment names are normally symbolic names.
- Operating system maintains a segment table for each process. It is usually stored in main memory as a segment that is not to be loaded as long as the process can run.
- Each entry in the segment table has a segment base and a segment limit. The base field contains the segment relocation register for the target segment. The segment limit field contains the length of the segment.
- Fig. 5.7.3 shows an address translation in segmentation.
- Segment table contains the physical address of the start of the segment. Then add the offset to the base and generate the physical address.
- If the required reference is not found in one of the segment registers, then error is generated. At the same time, operating system does lookup in segment table and loads new segment descriptor into the register.

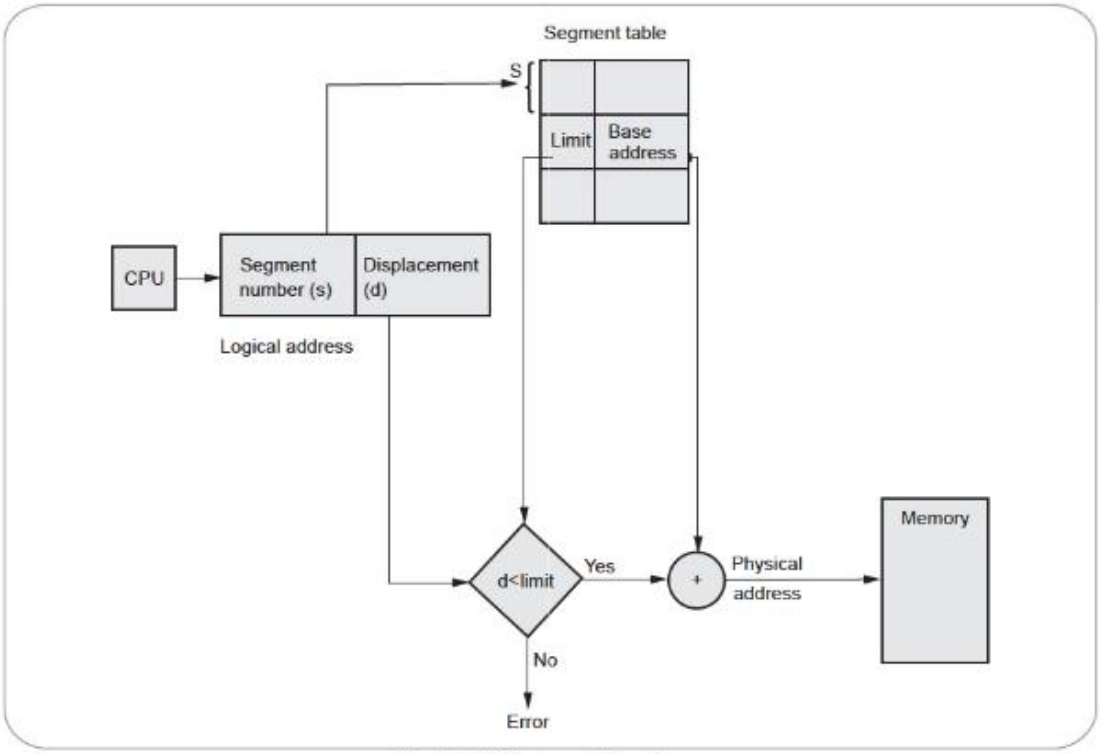


Fig. 5.7.3 Segment hardware

5.7.1 Protection and Sharing

- Sharing of segment provides less overhead than the pure paging segment. Multiple processes can share a segment.
- Two processes share a segment when their segment table entries point to the same segment in main memory. Operating system maintains global segment table for everyone and also used by OS.
- Access rights for segment are usually included in table entry. An illegal access of memory is protected by memory mapping hardware. It checks the protection bits associated with each segment table.

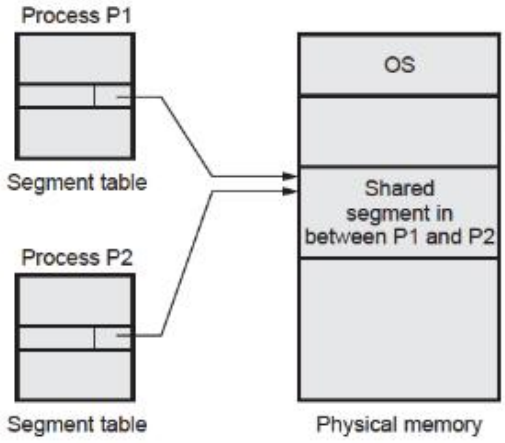


Fig. 5.7.4 Segmentation with sharing

- Fig. 5.7.4 shows sharing in segmentation. Access rights for segments are : Read, write, append and execute.
- Segmentation suffer from external fragmentation because segment sizes vary. Entire segment is either memory or on secondary storage disk.

5.7.2 Advantages

1. It provides virtual memory.
2. Allows dynamic segment growth.
3. Segmentation assists dynamic linking.
4. Segmentation is visible.

5.7.3 Disadvantages

1. Maximum size of a segment is limited by the size of main memory.
2. Difficulty to manage variable size segments on secondary storage.
3. Fragmentation and complicated memory management.

5.7.4 Difference between Segmentation and Paging

Sr. No.	Segmentation	Paging
1.	Program is divided into variable size segments.	Program is divided into fixed size pages.
2.	User (or compiler) is responsible for dividing the program into segments.	Division into pages is performed by the operating system.
3.	Segmentation is slower than paging.	Paging is faster than segmentation.
4.	Segmentation is visible to the user.	Paging is invisible to the user.
5.	Segmentation eliminates internal fragmentation.	Paging suffers from internal fragmentation.
6.	Segmentation suffers from external fragmentation.	There is no external fragmentation.
7.	Processor uses page number, offset to calculate absolute address.	Processor uses segment number, offset to calculate absolute address.
8.	OS maintain a list of free holes in main memory.	OS must maintain a free frame list.

Ex. 5.7.1 *Describe a mechanism by which one segment could belong to the address space of two different processes.*

Sol. : Segment tables are a collection of base-limit registers, segments can be shared when entries in the segment table of two different jobs point to the same physical location. The two segment tables must have identical base pointers, and the shared segment number must be the same in the two processes.

Board Questions

1. *Explain concept of page replacement with suitable diagram.* **MSBTE : Summer-15, Marks 6**
2. *Differentiate between paging and segmentation.* **MSBTE : Winter-15, Marks 4**
3. *Differentiate between paging and segmentation. (any six points)* **MSBTE : Summer-17, Marks 6**
4. *Compare paging and segmentation memory management techniques.* **MSBTE : Winter-17, Marks 4**

5.8 Page Replacement Algorithm

- When a page fault occurs, page replacement algorithms are used for loading the page in memory and no free page frame exist in memory.
- Page fault occurs if a running process references a nonresident page.
- The goal of a replacement strategy is to minimize the fault rate. To evaluate a replacement algorithm, following parameters are used :
- 1. The size of a page 2. A set of reference strings 3. The number of page frames.
- Given these inputs, we can determine the number of page faults and hence the page-fault rate. A reference string is a list of pages in order of their reference by the processor.
- When we find that a page frame reference is in main memory then we have a page hit and when page fault occurs we say it is page miss. A poor choice of page replacement algorithm may result in lot of page misses.
- Page replacement algorithm deals with how the kernel decides which page reclaim. Operating systems selects the local or global page replacement policy. Local replacement policy allocates a certain number of pages to each process. If a process needs new page, it must replace one of its own pages. If the OS uses a global policy, it can take a page from any process, using global selection criteria.
- UNIX OS uses global page replacement policy.

5.8.1 First-In-First-Out

- FIFO page replacement algorithm removes the pages that have been in memory the longest. Here system keeps track of the order in which pages enter primary memory.

- When page must be replaced, the oldest page is selected. System maintain a FIFO queue of pages. Pages are inserted at the tail of the queue.
- FIFO algorithms interfere with the locality of the process.
- Consider the reference string : 1, 2, 3, 4, 2, 5, 3, 4, 2, 6, 7, 8, 7, 9, 7, 8, 2, 5, 4, 9
- Page frame size = 3

Reference string	1	2	3	4	2	5	3	4	2	6	7	8	7	9	7	8	2	5	4	9
Page frame 1	1	1	1	4	4	4	4	4	4	6	6	6	6	9	9	9	9	9	4	4
Page frame 2		2	2	2	2	5	5	5	5	5	7	7	7	7	7	7	2	2	2	2
Page frame 3			3	3	3	3	3	3	2	2	2	8	8	8	8	8	8	5	5	9
Page fault	PF	PF	PF	PF		PF			PF	PF	PF	PF		PF			PF	PF	PF	PF

Total number of page fault = 14

Page frame size = 4

Reference string	1	2	3	4	2	5	3	4	2	6	7	8	7	9	7	8	2	5	4	9
Page frame 1	1	1	1	1	1	5	5	5	5	5	5	5	5	9	9	9	9	9	9	9
Page frame 2		2	2	2	2	2	2	2	2	6	6	6	6	6	6	6	2	2	2	2
Page frame 3			3	3	3	3	3	3	3	3	7	7	7	7	7	7	7	5	5	5
Page frame 4				4	4	4	4	4	4	4	4	8	8	8	8	8	8	8	4	4
Page fault	PF	PF	PF	PF		PF				PF	PF	PF		PF			PF	PF	PF	

Total number of page fault = 12

5.8.1.1 Belady's Anomaly

- Using FIFO algorithm for some reference strings actually generate more page faults when more page frames are allotted. This truly unexpected result was first demonstrated by Belady in 1970 and is known as Belady's anomaly.
- FIFO does not satisfy the inclusion property and is not a stack algorithm.
- A page replacement algorithm that satisfies the inclusion property is free from Belady anomaly.
- Consider the reference string : 1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5.
- Page frame size = 3

Reference string	1	2	3	4	1	2	5	1	2	3	4	5
Page frame 1	1	1	1	4	4	4	5	5	5	5	5	5
Page frame 2	-	2	2	2	1	1	1	1	1	3	3	3
Page frame 3	-	-	3	3	3	2	2	2	2	2	4	4
Page fault	PF	PF	PF	PF	PF	PF	PF			PF	PF	

Total number of page fault = 9

Page frame size = 4

Reference string	1	2	3	4	1	2	5	1	2	3	4	5
Page frame 1	1	1	1	1	1	1	5	5	5	5	4	4

Page frame 2	-	2	2	2	2	2	2	1	1	1	1	5
Page frame 3	-	-	3	3	3	3	3	3	2	2	2	2
Page frame 4	-	-	-	4	4	4	4	4	4	3	3	3
Page fault	PF	PF	PF	PF			PF	PF	PF	PF	PF	PF

Total number of page fault = 10

- Paging algorithm has worse performance when the amount of primary memory allocated to the process is increased. The problem creates because the set of pages loaded with a memory allocation of 3 is not same as the with a memory allocation of 4.
- There are a set of demand paging algorithms whereby the set of pages loaded with an allocation of m frames is always a subset of the set of pages loaded with an allocation of m + 1 frame. This property is called the inclusion property.
- FIFO does not satisfy the inclusion property and is not a stack algorithm. LRU can be a stack algorithm.
- FIFO algorithm uses only the modified and status bits when swapping is used.

Advantage of FIFO :

1. FIFO is very simple and easy to implement.

Disadvantages of FIFO :

1. It suffers from Belady's anomaly.
2. It is not very effective.
3. System needs to keep track of each frame.

5.8.2 LRU Page Replacement Algorithm

- LRU stands for least recently used. This replacement policy chooses to replace the page which has not been referenced for the longest time. This policy assumes the recent past will approximate the immediate future. The operating system keeps track of when each page was referenced by recording the time of reference or by maintaining a stack of references.
- System must maintain a time stamp for every page that indicates the time of the last access. When deciding which page to remove, the operating system must scan all the pages of memory for finding the oldest time stamp. Instead of keeping an extra time stamp bits, LRU is implement by maintaining pages in an ordered list.
- System can manage LRU with a list called the LRU stack or the paging stack. In the LRU stack, the first entry describes the page referenced least recently, the last entry describes to the last page referenced. If a page is referenced, move it to the end of the list.
- LRU algorithm does not interfere with the locality of the process.
- Consider the reference string : 1, 2, 3, 4, 2, 5, 3, 4, 2, 6, 7, 8, 7, 9, 7, 8, 2, 5, 4, 9
- Page frame size = 3

Reference string	1	2	3	4	2	5	3	4	2	6	7	8	7	9	7	8	2	5	4	9
Page frame 1	1	1	1	4	4	4	3	3	3	6	6	6	6	9	9	9	2	2	2	9
Page frame 2	-	2	2	2	2	2	2	4	4	4	7	7	7	7	7	7	7	5	5	5
Page frame 3	-	-	3	3	3	5	5	5	2	2	2	8	8	8	8	8	8	8	4	4
Page fault	PF	PF	PF	PF		PF	PF	PF	PF	PF	PF	PF		PF			PF	PF	PF	PF

Total number of page fault = 16

- Page frame size = 4

Reference string	1	2	3	4	2	5	3	4	2	6	7	8	7	9	7	8	2	5	4	9
Page frame 1	1	1	1	1	1	5	5	5	5	6	6	6	6	6	6	6	2	2	2	2
Page frame 2	-	2	2	2	2	2	2	2	2	2	2	2	2	9	9	9	9	5	5	5
Page frame 3	-	-	3	3	3	3	3	3	3	3	7	7	7	7	7	7	7	7	4	4
Page frame 4	-	-	-	4	4	4	4	4	4	4	4	8	8	8	8	8	8	8	8	9
Page fault	PF	PF	PF	PF		PF				PF	PF	PF		PF			PF	PF	PF	PF

Total number of page fault = 13

- LRU is a stack algorithm. Increase in memory size will never cause an increase in the number of page interrupts.

Advantages :

1. It is quite good algorithm.
2. It is amenable to full statistical analysis.
3. Never suffers from Belady's anomaly.

Disadvantages :

1. Problem is how to implement.
2. It requires hardware assistance.

5.8.3 LRU Approximation Algorithms

- LRU is a desirable algorithm to use, but it is expensive to implement directly. It is often approximated. By using one reference bit, user can do a second-chance or clock replacement algorithm.
- Page frames are arranged in circular list. Initially each reference bit is set to 0. It is set to 1 any time the page is referenced. Algorithm uses pointer to points to the next candidate for a page replacement.
- When a page replacement is needed, the frame pointed to is considered as a candidate. If its reference bit is 0, it is selected. If its bit is 1, the bit is set to 0, and the pointer is incremented to examine the next frame. This continues until a frame is found with a reference bit of 0. This may require that all frames be examined; bringing us around to the one we started with.

Second chance replacement

- In some books, the second chance replacement policy is called the clock replacement policy. In the second chance page replacement policy, the candidate pages for removal are consider in a round robin matter and a page that has been accessed between consecutive considerations will not be replaced.
- The page replaced is the one that, considered in a round robin matter. It has not been accessed since its last consideration.
- Working :

1. Add a "second chance" bit to each memory frame.
2. Each time a memory frame is referenced, set the "second chance" bit to ONE (1) - this will give the frame a second chance...
3. A new page read into a memory frame has the second chance bit set to ZERO (0)
4. When you need to find a page for removal, look in a round robin manner in the memory frames :
   i. If the second chance bit is ONE, reset its second chance bit (to ZERO) and continue.
   ii. If the second chance bit is ZERO, replace the page in that memory frame.

String		0		4		1		4		2		4		3		4		2		4		0		4		1		4		2		4		3		4
Frame 1	0	0	0	0	0	0	0	0	0	2	0	2	0	2	0	2	1	2	1	2	0	2	0	2	0	1	0	1	0	1	0	1	0	3	0	3
Frame 2		-		4	0	4	0	4	1	4	1	4	0	4	1	4	1	4	1	4	0	4	1	4	1	4	1	4	0	4	1	4	1	4	1	4
Frame 3		-		-		1	0	1	0	1	0	1	0	3	0	3	0	3	0	3	0	0	0	0	0	0	0	0	0	2	0	2	0	2	0	2
PF		PF		PF		PF				PF				PF								PF				PF				PF				PF		

Total number of page fault = 9

5.8.4 Optimal Page Replacement

- The data which will not be accessed permanently or for a longest time should be replaced in optimal page replacement algorithm.
- Optimal page replacement algorithm gives less page fault as compared to FIFO and LRU. This algorithm never suffer from Belady's anomaly.
- Use of this page-replacement algorithm guarantees the lowest possible page-fault rate for a fixed number of pages
- The key distinction between FIFO and optimal is that FIFO uses the time when a page was brought into memory whereas optimal uses the time when a page is to be used.
- It is difficult to implement because of future reference.
- Consider the reference string : 1, 2, 3, 4, 2, 5, 3, 4, 2, 6, 7, 8, 7, 9, 7, 8, 2, 5, 4, 9
- Page frame size = 3

Reference string	1	2	3	4	2	5	3	4	2	6	7	8	7	9	7	8	2	5	4	9
Page frame 1	1	1	1	4	4	4	4	4	4	6	7	7	7	7	7	7	2	2	4	4
Page frame 2		2	2	2	2	5	5	5	5	5	5	8	8	8	8	8	8	5	5	5
Page frame 3			3	3	3	3	3	3	2	2	2	2	2	9	9	9	9	9	9	9
Page fault	PF	PF	PF	PF		PF			PF	PF	PF	PF		PF			PF	PF	PF	

Total number of page fault = 13

- Page frame size = 4

Reference string	1	2	3	4	2	5	3	4	2	6	7	8	7	9	7	8	2	5	4	9
Page frame 1	1	1	1	1	1	5	5	5	5	5	5	5	5	9	9	9	9	9	9	9
Page frame 2	-	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	5	5	5
Page frame 3	-	-	3	3	3	3	3	3	3	6	7	7	7	7	7	7	7	7	4	4
Page frame 4	-	-	-	4	4	4	4	4	4	4	4	8	8	8	8	8	8	8	8	8
Page fault	PF	PF	PF	PF		PF				PF	PF	PF		PF				PF	PF	

Total number of page fault = 11

Advantages :

1. It has lowest page fault rate.
2. Optimal never suffer from Belady's anomaly.
3. Used for comparison studies.

Disadvantages :

1. Difficult to implement.
2. It requires future reference string.

5.8.5 Difference between FIFO, LRU and Optimal

Sr. No.	FIFO	LRU	Optimal
1.	Number of page fault is more than LRU and optimal.	Number of page fault is more than optimal and less than FIFO.	Number of page fault is less both.
2.	Suffer from Belady's anomaly.	Does not suffer from Belady's anomaly.	Does not suffer from Belady's anomaly.
3.	Used in OS.	Used in OS.	Not used in OS.
4.	Future predication is not required.	Future predication is not required.	Future predication is required.
5.	Simple to implement.	Considered to be good.	Difficult to implement.

Ex. 5.8.1 *Consider the string 0, 1, 2, 3, 0, 1, 2, 3, 0 ,1 2, 3, 4, 5, 6, 7. With frame size 3 and 4. Calculate page fault.*

Sol. : 1) FIFO

Page frame size = 3

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	3	3	3	2	2	2	1	1	1	4	4	4	7
1		1	1	1	0	0	0	3	3	3	2	2	2	5	5	5
2			2	2	2	1	1	1	0	0	0	3	3	3	6	6
Page fault	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

* Page fault

This example incurs 16 page faults in FIFO algorithm,

FIFO algorithm with four page frames

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
1		1	1	1	1	1	1	1	1	1	1	1	1	5	5	5
2			2	2	2	2	2	2	2	2	2	2	2	2	6	6
3				3	3	3	3	3	3	3	3	3	3	3	3	7
Page fault	*	*	*	*									*	*	*	*

Number of page fault is 8.

2) LRU

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	3	3	3	2	2	2	1	1	1	4	4	4	7
1		1	1	1	0	0	0	3	3	3	2	2	2	5	5	5
2			2	2	2	1	1	1	0	0	0	3	3	3	6	6
Page fault	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

Number of page fault = 16.

Consider the same reference string with 4 page frames

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
1		1	1	1	1	1	1	1	1	1	1	1	1	5	5	5
2			2	2	2	2	2	2	2	2	2	2	2	2	6	6
3				3	3	3	3	3	3	3	3	3	3	3	3	7
Page fault	*	*	*	*									*	*	*	*

Number of page fault = 8.

3) Optimal

Frame	0	1	2	3	0	1	2	3	0	1	2	3	4	5	7
0	0	0	0	0	0	0	0	0	0	1	1	1	4	4	7
1		1	1	1	1	1	2	2	2	2	2	2	2	5	5
2			2	3	3	3	3	3	3	3	3	3	3	3	6
Page fault	*	*	*	*			*			*			*	*	*

Number of page fault = 10.

With page frame = 4 for same reference string.

Frame	0	1	2	3	0	1	2	3	0	1	3	4	5	6	7
0	0	0	0	0	0	0	0	0	0	0	0	4	4	4	4
1		1	1	1	1	1	1	1	1	1	1	1	5	5	5
2			2	2	2	2	2	2	2	2	2	2	2	6	6
3				3	3	3	3	3	3	3	3	3	3	3	7
Page fault	*	*	*	*								*	*	*	*

Number of page fault = 8.

Ex. 5.8.2 *Explain FIFO (First in First out) page replacement algorithm for reference string 7012030423103.*

MSBTE : Winter-17, Marks 4

Sol. :

	7	0	1	2	0	3	0	4	2	3	1	0	3
1	7	7	7	2	2	2	2	4	4	4	4	1	1
2	-	0	0	0	0	3	3	3	2	2	2	0	0
3	-	-	1	1	1	1	0	0	0	3	3	3	3

Number of page fault = 11

Board Questions

1. *Explain LRU page replacement algorithm by taking suitable example.* **MSBTE : Summer-15, Marks 6**
2. *Describe optimal page replacement algorithm with example.* **MSBTE : Summer-16, Marks 4**
3. *Explain FIFO (First in First out) page replacement algorithm for reference string 7012030423103.* **MSBTE : Winter-17, Marks 4**

5.9 Two Marks Questions with Answers

Q.1 *What is concept of paging ?* **MSBTE : Winter-18**

Ans. : In paging, operating system divides each incoming programs into pages of equal size. The sections of a disk are called block or sectors. The sections of main memory are called page frames. One sector will hold one

page of job instructions and fit into one page frame of memory.

Q.2 *What is internal fragmentation ?*

Ans. : Internal fragmentation exists when the smallest available block is larger than the requested memory. It is the memory which is internal to a partition, but is not being used.

Q.3 *Distinguish between internal and external fragmentation.*

Ans. : Internal fragmentation is the area in a region or a page that is not used by the process it is allocated to. The space is wasted until the process terminates. External fragmentation occurs when there is enough free space to satisfy a request for memory, but none of the free holes between processes in memory is large enough to satisfy the request.

Q.4 *State the function of memory manager.*

Ans. : **Functions of memory manager**

- Allocates primary memory to processes.
- Maps process address space to primary memory.
- Minimizes access time using cost-effective memory configuration.
- May use static or dynamic techniques

Q.5 *What are memory pages and segments ?*

Ans. : Logical memory is also broken into blocks of the same size called **pages**. It divides a program into a smaller block called segments, each of which is allocated to memory independently.

Q.6 *Name two differences between logical and physical addresses.*

Ans. : A logical address does not refer to an actual existing address; rather, it refers to an abstract address in an abstract address space. Contrast this with a physical address that refers to an actual physical address in memory. A logical address is generated by the CPU and is translated into a physical address by the memory management unit (MMU). Therefore, physical addresses are generated by the MMU.

Q.7 *Differentiate segmentation from paging.*

Ans. : In segmentation, program is divided into variable size segments whereas program is divided into fixed size pages in paging. Segmentation is visible to user and paging is invisible.

Q.8 *What is a translation look aside buffer used for ?*

Ans. : Problem with paging is that, extra memory references to access translation tables can slow programs down by a factor of two or three. Too many entries in translation tables to keep them all loaded in fast processor memory. To solve this problem TLB is used. TLB is used to store a few of the translation table entries.

Q.9 *What is segmentation ?*

Ans. : Segmentation divides a program into a number of smaller blocks called segments.

Q.10 *What is called pages ?*

Ans. : The address space is usually broken into fixed-size blocks, called pages. Each page resides either in main memory or on disk.

Q.11 *Define external fragmentation ?*

Ans. : Total memory space exists to satisfy a request, but it is not contiguous; storage is fragmented into a large number of small holes.

Q.12 *What is frame table ?*

Ans. : Allocation and availability of the frame information is kept in a data structure called a frame table. Frame table has one entry for each physical page frame, indicating whether the latter is free or allocated and if it is allocated, to which page of which process or processes.

Q.13 *Define TLB.*

Ans. : TLB is the memory cache of the most recently used page table entires within the memory management unit.

Q.14 *What is meant by Belady's anomaly ?*

Ans. : For some page replacement algorithms, the page fault rate may increase as the number of allocated frames increases.

Q.15 *What is optimal page replacement ?*

Ans. : The optimal policy selects for replacement the page that will not be used for longest period of time.

❑❑❑

UNIT - VI

6 FILE MANAGEMENT

6.1 File

6.1.1 File Concept

- Storing and retrieving information is the necessity of all computer application. There are three problems which occur while storing and retrieving information.

1. A process can store limited amount of information within its own address space when a process is executed.
2. When process gets terminated information gets lost.
3. It is necessary for multiple processes to access the information at same time.

- To overcome the above problems information must be stored on files. File management is the most visible service of an operating system. Computer can store information on various storage media such as tapes, disks etc.
- A file is a collection of related information defined by its creator. A file may be the collection of various types of data depending upon data types. In general file is a collection of bits, bytes, lines or records the meaning of which is defined by the creator.

6.1.2 File Attributes

- A file attribute may vary from one O.S to another. But typically consists of these :
- **Name :** The name attribute is specified by the user which is the information kept in human readable form. A file is named for the convenience of its human users.
- **Identifier :** It is a unique tag or a number which is used to identify file within the file system.
- **Type :** It specifies the file type that is supported by the file system such as document file, image file etc.
- **Size :** This attribute specifies the size of the file on the system. It may be stored in the form of bytes, kilobytes, megabytes etc.
- **Location :** This information specifies where exactly the file is stored on the device.
- **Protection :** It is done by means of access permission or password. It is usually a secret code used by the user to protect his/ her file/data.
- **Time, date and user identifier :** This information may be kept for monitoring the usage of file/data. And also can be used for protection and security purpose.

6.1.3 File Types

File types	Usual extension
Executable	.exe
Object	.obj
Source	.c,.java
Batch	.bat
Text	.txt
Library	.lib

1. **Executable file :** It is the series of code section that the loader can bring into the memory and execute.
2. **Object file :** It is the sequence of bytes organized into blocks, understandable by system linker.

3. **Source file :** It is the sequence of subroutines and functions each of which is further organized as declaration followed by executable statements.
4. **Batch file :** It is the collection of the commands to the command interpreter.
5. **Text file :** It is the sequence of characters organized into lines.
6. **Library files :** It is collection of library functions for the program.

6.1.4 File Operations

- The O.S. can provide system calls for doing file operations.

1. **Creating a file :** Two steps are necessary for creating a file :
   - A space in the file system
   - An entry for new file must be made in the directory
2. **Writing a file :** While performing the write operation, the system call is activated which specifies the name of the file and the information to be written in the file. The system finds out the directory to find out the location of the file when we specify the name of the file. The pointer is the logical reference, which is used when the directory entry is done. It is necessary to store a pointer to the end of current file. Using this pointer information the address of the next block can be easily derived and the information can be written to the specified file. The pointer information must be updated as soon as the write operation is performed.
3. **Reading a file :** This operation reads the data from the file. The system call is specifies the name and location where read operation is to be performed. Again the directory is searched for associated entry and the system needs to keep the read pointer to the location in the file where the next read is to take place.
4. **Repositioning within the file :** A directory is searched for the appropriate entry and current file position is set to a given value. This file operation is also known as "file seek".
5. **Deleting a file :** The system call searches the directory entry for the specified file to be deleted from the file system.
6. **Truncating a file :** Sometimes a user may want to erase the contents of a file but keep its attributes. In these cases truncating is done, this allows attributes to remain unchanged.

- These all operations are those which are always required. Other common operations are appending renaming copying etc.

6.1.5 File System Structures

- Structure of a file is of mainly three types. Some files store data plainly that is one by one character

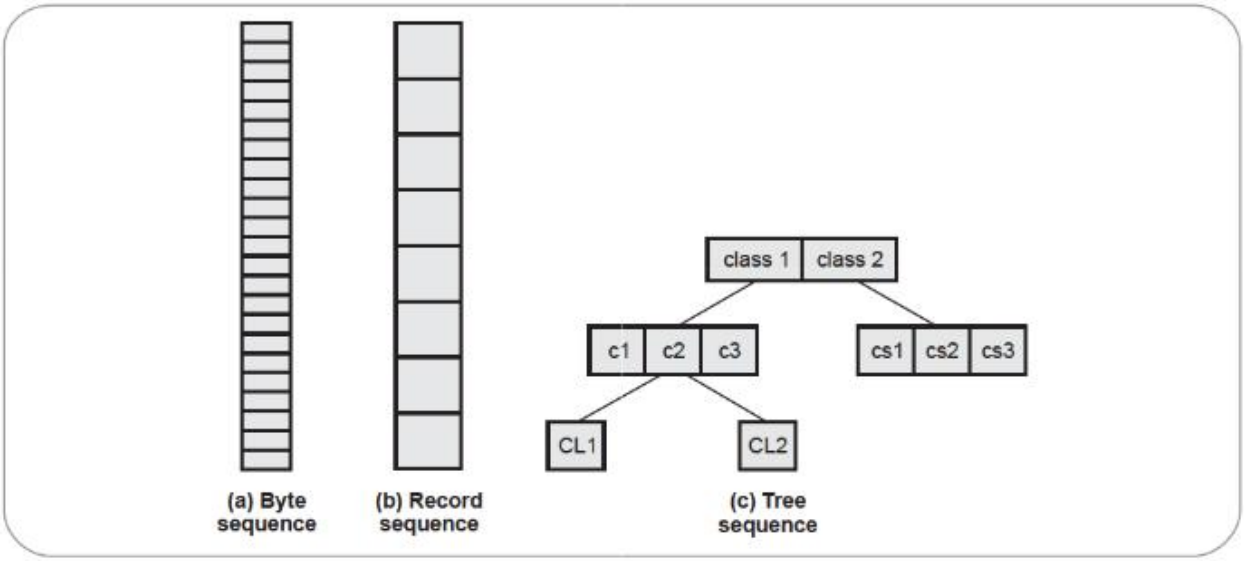


Fig. 6.1.1

or numbers are stored in other types there might be different arrangements made by the file system. Files can be structured in a different ways. Here are some common possibilities following are three major file structures :

1) Unstructured sequence of bytes (byte sequence)
2) Sequence of fixed length of records (record sequence)
3) Tree records of different or may be of same length (tree sequence)

Fig. 6.1.1 shows the type of file structures.

- In first type of file structure contents are stored in the form of byte i.e. sequence of zero and one. though this is an binary file its contents from user point of view is a normal file so we can say that internally data is stored in byte but user program make these file accessible and readable by user.
- In the next type where data is stored in the form of records. Records are collection of collective data means detail of one employee like employee code, name, salary, address etc. could be form of one record.
- The third type is critical means used in commercial applications means data searching is quite tedious. In this type as data is hierarchical order.

Board Questions

1. *List different types of files. Explain basic operations on file.* **MSBTE : Winter-15, Marks 4**
2. *State and explain any four file attributes.* **MSBTE : Summer-15, Marks 4**
3. *List and explain any four file attributes.* **MSBTE : Winter-16, 17, Marks 4**
4. *What is file ? List and explain attributes of files.* **MSBTE : Summer-16**
5. *Explain any six file operations performed by OS.* **MSBTE : Winter-17, Marks 6**
6. *Explain file structure with example.* **MSBTE : Summer-17, Marks 4**
7. *Describe concept of file, its types and operations on file attributes in detail.* **MSBTE : Summer-18, Marks 8**

6.2 Access Methods

6.2.1 Sequential Access Method

- The sequential access method is the simplest method. Information in a file is processed in order one record after the other the bulk of operation on the file is read and writes. In a read operation "read next "reads the next portion of file and automatically advances a file pointer. Similarly the write operations "write next "appends to the end of the file and advance to the end of newly written material i.e. the new end of the file.

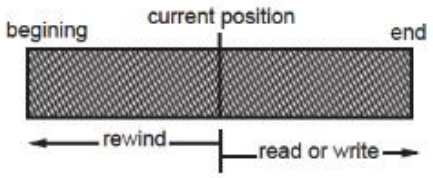


Fig. 6.2.1 Sequential access file

- The above Fig. 6.2.1 shows the tape model for the file storage. The current portion shows the current access of the file. In the sequential access method the required information is to be stored or scanned sequentially one after another.

6.2.2 Direct Access Method

- It is also called as relative access method. A file is made up of fixed length records that allows program to read and write records rapidly in no particular order. Direct access method is based on disc types storage devices since disk allows random access to file block. For direct access file is viewed as no sequence of bocks or records.
- For e.g. We may read block 14 then we may read block 53 and then write block 7. There are no restrictions on order of reading or writing files for direct access.
- For direct access method the file operation must be modified to include the block number as a parameter. Thus we have read 'n' or write 'n' where n is the block number rather than read next or write next. Table below shows the simulation of sequential access on direct access.

Sequential access	Direct access
reset	cp = 0
read next	Read cp
	cp = cp+1
write next	Write cp
	cp = cp = 1

Table 6.2.1 Simulation of sequential access on direct access

6.2.3 Swapping

- A process must reside in memory to be ready to execute. A process can be moved temporarily out of memory to a backing store, and then brought back into memory for continued execution. For eg : In a multiprogramming environment when number of processes is queued for execution and if the system is implemented with a round robin algorithm, then when time quantum/time slice expires then the process will be swapped out and the new process will be swapped into the memory space that just has been freed. The CPU scheduler selects the process for scheduling. When the time slice or time quantum gets over, the CPU switches to next job. The Fig. 6.2.2 shows the swapping of two processes using disk as backing store.

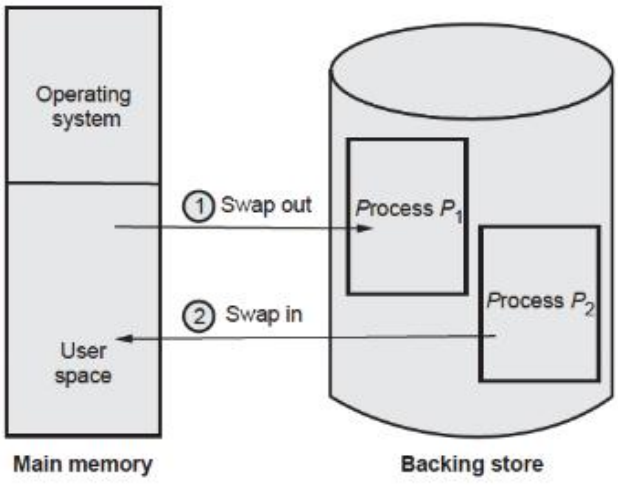


Fig. 6.2.2 Swapping of two processes using disk as backing store

- The swapping policy is used for priority based scheduling algorithms.
- If a higher priority process comes to the system and needs service, the memory manager can swap out the lower priority process and then load and execute the higher priority process. When higher priority process finishes its execution the lower priority process can be moved back into the system and continue with execution. This variant is sometimes called **"roll-in and roll-out"**. Swapping involves a backing store which is generally a large capacity disk. The system maintains the ready queue which consists off processes ready to run. Then the scheduler selects the process and dispatcher gives the control of CPU for execution.

6.2.4 Allocation Methods

6.2.4.1 Contiguous Allocation

- It requires that each file occupy a set of contiguous block on disk. This address is defined linear ordering i.e. proceeding straight from one stage to another. Note that with this ordering, accessing block b+1 after the block b requires no head movement. When head movement is needed, (i.e only one sector of cylinder to next) it is only one track. That is from last sector to the first sector of next cylinder. Disk address and the length define contiguous allocation of the file. If the file is n blocks long and starts at location a then it occupies a, a+1, a+2,..., a+n–1. The directory entry for each file indicates the address of starting block and the length of the area allocated for this file. Fig. 6.2.3 shows directory of each file indicating the address of starting block and the length of the area allocated for this file. (See Fig. 6.2.3 on next page)

Advantages :

1. Accessing of a file that has been allocated contiguously is easy.
2. Both sequential and direct access can be supported by contiguously allocated.

Disadvantages :

1. In contiguous allocation, the problem is that one couldn't predict or determine the space required for the allocation of the file.
2. Even if the total amount of space needed for a file is known in advance, tree allocation may be inefficient because it may lead to external fragmentation.

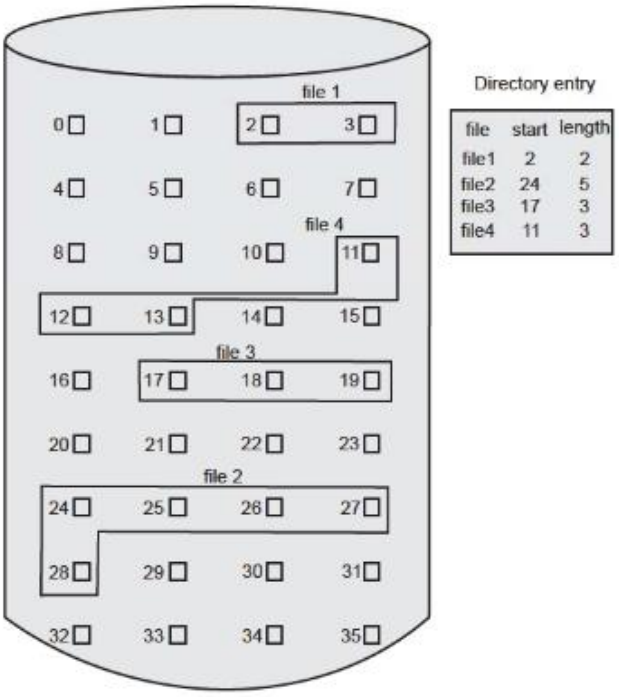


Fig. 6.2.3 Contiguous allocation method

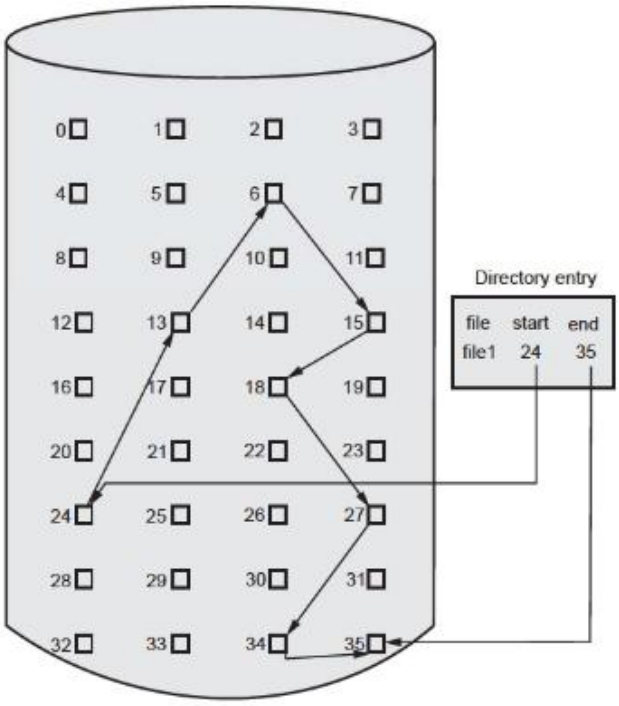


Fig. 6.2.4 Linked allocation

External Fragmentation

- External fragmentation exists when enough total disk space exists to satisfy the request but it is not contiguous. Storage is fragmented into large number of small partitions depending upon the total amount of disk storage and average file size; external fragmentation may be either a minor or a major problem.

6.2.4.2 Linked Allocation

- The problems associated in contiguous allocation are resolved in linked allocation technique. In linked allocation each file is a linked list of disk blocks. The disk blocks are distributed anywhere on the disk. The directory structure contains a pointer to the 1st and last block of the file i.e. start and the end. Fig. 6.2.4 shows the linked allocation.
- For eg. A file of 8 blocks which might start at block 24 and continue at block 13 then block 6, then block 15, then block 18, then block 27, then block 34 and finally block 35. Each block contains a pointer to the next block. These pointers is not known to the users. To create a new file a new entry is created in the directory. Each directory has a pointer to 1st disk block of the file. Initially, when a new file is created the pointer is has null value indicating empty file. The file gets extended according to the pointer value of the next block. These blocks get linked together with the help of the pointer value.

Advantages :

1. There is no external fragmentation wth linked allocation. Any free block can be used to satisfy the request.
2. There is no need of declaring the size of the file when it is created. A file can continue growing as long as free blocks are available.

Disadvantages :

1. The major problem is , it can be used effectively only for sequential access of file.
2. Extra space required for pointer.
3. It is not reliable i.e. If an pointer gets lost or damaged it could not have access to that particular block or file.

6.2.4.3 Indexed Allocation

- Linked allocation cannot support efficient direct access because the pointer to blocks are scattered with the block themselves all over the disk and

must be retrieved in order. Index allocation is the solution to the problems of both the previous techniques. It provides this solution by bringing the entire pointers into one location called "index block". Fig. 6.2.5 shows the indexed allocation.

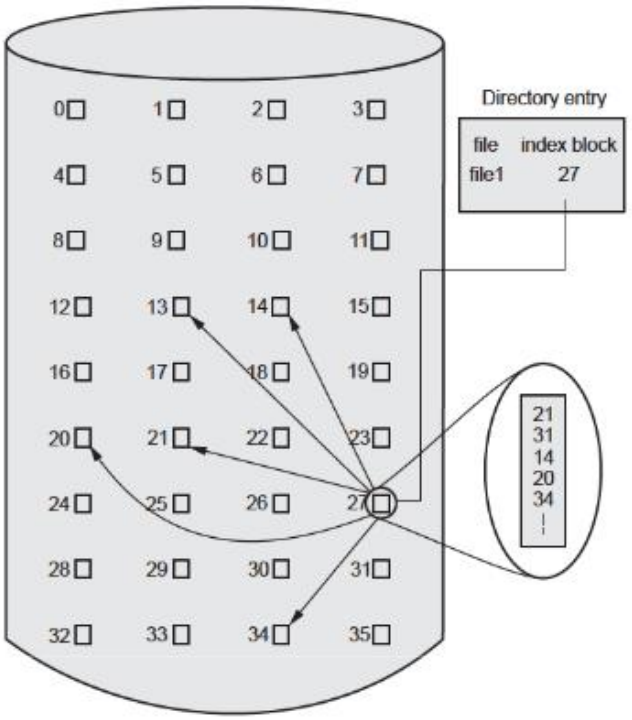


Fig. 6.2.5 Indexed allocation

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

Advantages : • It supports the direct access without suffering from external fragmentation and no extra space required for pointer.

Disadvantage : • Index allocation thus suffers from wasted space. The pointer overhead of index block is generally greater than the pointer overhead of linked allocation.

- For eg. : In a common case where we have a file of only 1 or 2 blocks, then with linked allocation we lose the space of only 1 pointer per block but with index allocation an entire index block must be allocated even if only 1 or 2 pointers will be used, so remaining space is on wastage.

Board Questions

1. *Draw and explain contiguous method for access.* **MSBTE : Winter-15, Marks 6**
2. *With suitable diagram, explain how linked allocation is performed.* **MSBTE : Winter-15, Marks 6**
3. *Describe the sequential file access method.* **MSBTE : Summer-15, Winter-16, Marks 4**
4. *List different file allocation methods. Explain any one with suitable diagram and example.* **MSBTE : Winter-16, Marks 6**
5. *Describe working of sequential and direct access methods.* **MSBTE : Summer-16, Winter-18, Marks 4**
6. *Explain following memory allocation methods: Linked.* **MSBTE : Winter-17, Marks 3**
7. *Explain any two file allocation methods with the help of diagram.* **MSBTE : Summer-17, Marks 6**
8. *Describe indexed allocation method with advantage and disadvantage.* **MSBTE : Winter-18, Marks 4**

6.3 Directory Structures

6.3.1 Single Level Directory Structure

- It is a simple directory structure where all the files are present in the same directory. These directories are very easy to understand. This directory has some restrictions like if the number of files get increases then the file names should be unique under same directory. Fig 6.3.1 shows the single level directory structure.

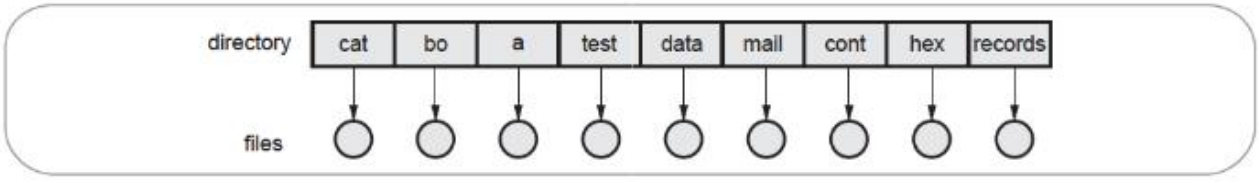


Fig. 6.3.1 Single level directory structure

- There is another drawback that if single user works on this single level system it may be difficult to remember the names of the files, if the files are more. Therefore the two major difficulties are naming and grouping.

Advantages :

i. Single level directory is very simple, easy to understand, maintain and easy to implement.

ii. The software design is also very simple.

iii. The ability to locate files is very fast as there is only one directory.

Disadvantages :

i. It is not used in multiuser system. For e.g. if two users A and B use the same filename for particular file then the file of user A or B may be overwritten.

ii. It is difficult to remember the names of the files if the files are very large in quantity.

6.3.2 Two Level Directory Structure

- In two level directory structure, the system maintains a master block that has one entry for each user. This master block contains the addresses of the directory of the users. In this system each user has his/her own "user file directory"(UFD). When user logs into the system the "master file directory"(MFD) is searched. This MFD contains the index names of the users with their account details. Once login done 1st MFD and then UFD is searched.
- See the Fig. 6.3.2, it indicates that there are different users who have their own directories and their respective files stored in their own directories. The process is very simple that when user logs on he/she can access only his/her own directory and of course files present in it. There might be different users like user1, user2, user3 etc. and it may contain the files directories listed below. Always in UNIX all the devices, directories, files each and every object is treated as file. An advantage of this type of structure is isolation, which keep one user separate from each user. But ultimately if two users need to communicate with each other then it becomes difficult. We also call this structure as inverted root tree structure.
- We use the terminology path name that is nothing but the exact location of the file where it is placed. There could be the same file name for different user. This structure does efficient searching. As different users are there, grouping cannot be done.

6.3.3 Tree Structured Directories

- A tree is the most common and popular directory structure. It is very useful because it solves the problem of grouping. It can group different users or directories. The tree has a root directory and every file in the system has a unique path name. A directory contains a set of files or sub directories. All directories have the same internal organisation. One bit in each directory entry defines the entry as a file or as a subdirectory. These subdirectories further may have files or again some directories. Each process has a current directory. This helps users to organize their data in a particular and efficient manner. Path describes where exactly that file or directory is stored. Path names can be of two types absolute and relative.
- An absolute path name begins at the root and follows a path down to the specified file giving the directory names on the path. A relative path name defines a path from the current directory.

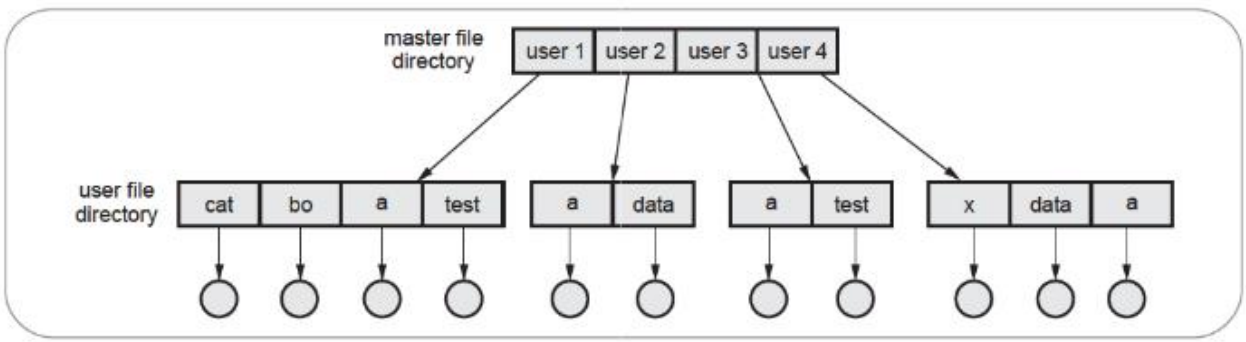


Fig. 6.3.2 Two level directory structure

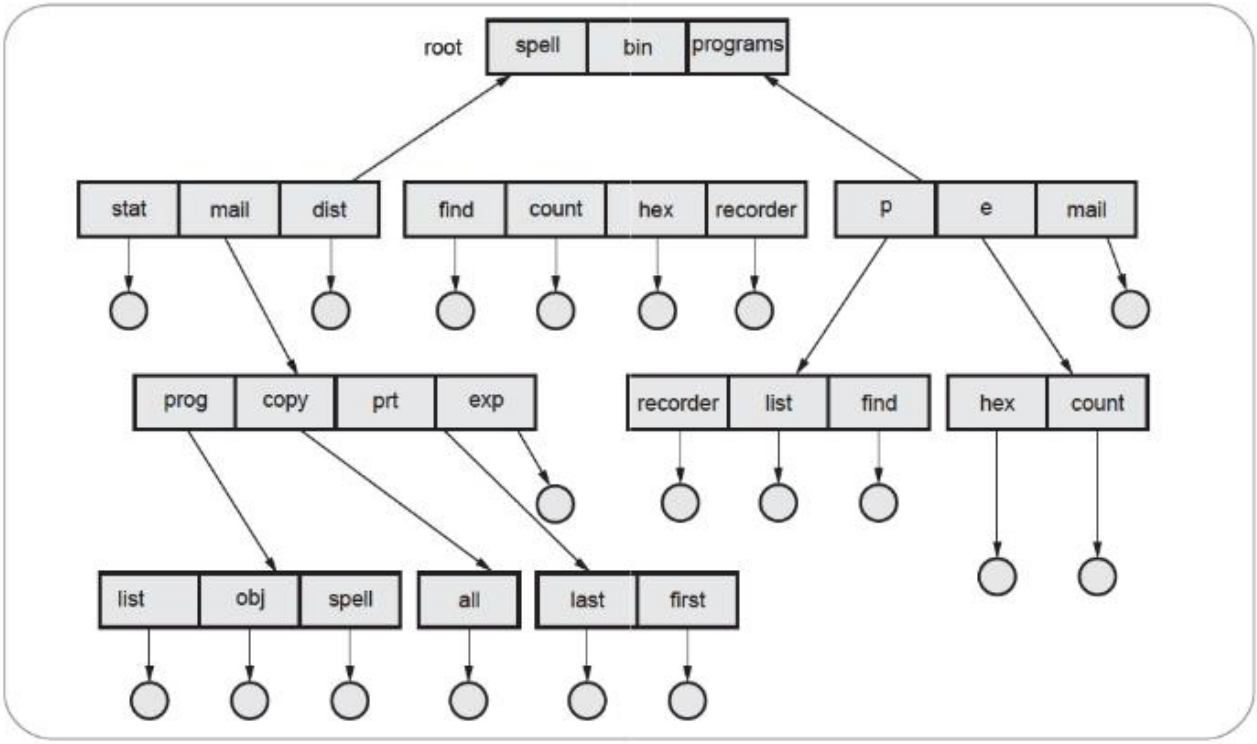


Fig. 6.3.3

- The important issue is how to delete a file directory if it already contains subdirectory or may be files. Some O.S. directly allows deleting this type of directory, But other O.S do not allow the same thing. If a directory is empty, its entry in the directory can simply be deleted. If the directory to be deleted is not empty, the none of the two approaches can be done. User must first delete all the files and sub directories in that directory. If a request is made to delete a directory, the entire directory's files and subdirectories are also to be deleted.

6.3.4 Disk Organization

Traditional disks have the following basic structure :

- One or more **platters** in the form of disks covered with magnetic media. **Hard disk** platters are made of rigid metal.
- Each platter has two working **surfaces**. Older hard disk drives would sometimes not use the very top or bottom surface of a stack of platters, as these surfaces were more susceptible to potential damage.
- 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 are immediately above one another and called a **cylinder**.
- Each track is further divided into **sectors,** traditionally containing 512 bytes of data each, although some modern disks occasionally use larger sector sizes. Sectors also include a header and a trailer. Larger sector sizes reduce the fraction of the disk consumed by headers and trailers but increase internal fragmentation and the amount of disk that must be marked bad in the case of errors.
- 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.
- The storage capacity of a traditional disk drive is equal to the number of heads i.e. the number of working surfaces, at times the number of tracks per surface, at times the number of sectors per track and at times the number of bytes per sector. A particular physical block of data is specified by providing the head-sector-cylinder number at which it is located.

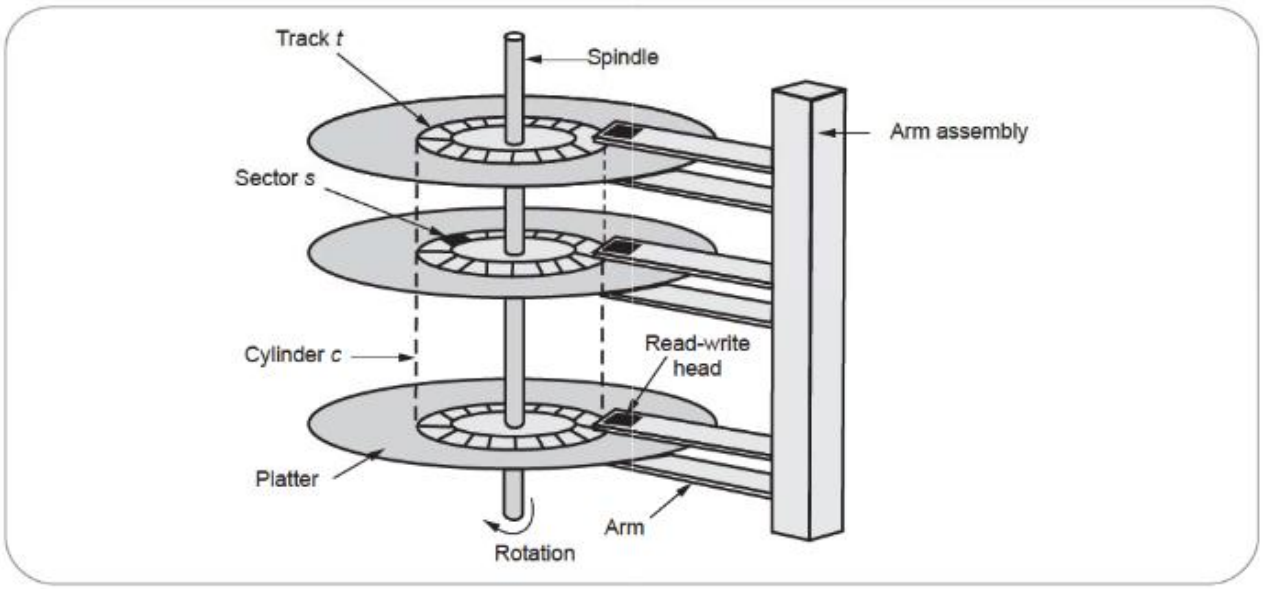


Fig. 6.3.4 Moving-head disk mechanism

- In operation the disk rotates at high speed, such as 7200 rpm i.e. 120 revolutions per second. The rate at which data can be transferred from the disk to the computer is composed of several steps such as :

a) The **positioning time, seek time or random-access time** is the time required to move the heads from one cylinder to another, and for the heads to position at the correct track after the move. This is typically the slowest step in the process and normally it constitutes a very high percentage of total R/W time.

b) The **rotational latency or rotational delay** is the amount of time required for the desired sector to rotate around and come under the R/W head. This can range anywhere from zero to one full revolution, and on the average will equal one-half revolution. This is another physical step and is usually the second slowest step behind seek time.

c) The **transfer rate or transmission time,** which is the time required to move the data electronically from the disk to the computer.

- Disk heads "fly" over the surface on a very thin cushion of air. If they accidentally contact with the disk surface, then a head crash occurs, which may not be repaired and destroys the disk completely and the entire disk must be replaced. For this reason, it is normal to park the disk heads when turning a computer off, which means to move the heads off the disk or to an area of the disk where there is no data stored.
- Disk drives are connected to the computer via a cable known as the **I/O Bus.** Some of the common interface formats include Enhanced Integrated Drive Electronics, (EIDE), Advanced Technology Attachment (ATA), Serial ATA (SATA), Universal Serial Bus (USB), Fiber Channel (FC), and Small Computer Systems Interface (SCSI).
- The **host controller** is at the computer end of the I/O bus, and the **disk controller** is built into the disk itself. The CPU issues commands to the host controller via I/O ports.
- Data is transferred between the magnetic surface and onboard **cache** by the disk controller. Disk usually have an in-built cache. the data is then transferred from that cache to the host controller and the motherboard memory at electronic speeds.

6.3.5 Disk Structure

- The traditional head-sector-cylinder (HSC) numbers are mapped to linear block addresses by numbering the first sector on the first head on the outermost track as sector 0. Numbering proceeds with the rest of the sectors on that same track, and then the rest of the tracks on the same cylinder before proceeding through the rest of the cylinders to the center of the disk. In modern exercise these linear

block addresses are used in place of the HSC numbers for a variety of reasons :

- The linear length of tracks near the outer edge of the disk is much longer than for those tracks located near the center, and hence it is possible to squeeze many more sectors onto outer tracks than onto inner ones.
- All disks have some bad sectors, and therefore disks maintain a few spare sectors that can be used in place of the bad ones. The mapping of spare sectors to bad sectors in managed internally to the disk controller.
- Modern hard drives can have thousands of cylinders, and hundreds of sectors per track on their outermost tracks. These numbers exceed the range of HSC numbers for many older operating systems, and therefore disks can be configured for any convenient combination of HSC values that falls within the total number of sectors physically on the drive.

- There is a limit to how closely packed individual bits can be placed on a physical media, but that limit is growing increasingly as there are many technological advancements are done.
- Modern disks pack many more sectors into outer cylinders than inner ones, using one of these two methods :

a. With **Constant Linear Velocity, CLV,** the density of bits is uniform from cylinder to cylinder. Because there are more sectors in outer cylinders, the disk spins slower when reading those cylinders, causing the rate of bits passing under the read-write head to remain constant. This is the approach used by modern CDs and DVDs.

b. With **Constant Angular Velocity, CAV,** the disk rotates at a constant angular speed, with the bit density decreasing on outer cylinders.

6.3.6 RAID Structure

- Disk drives have continued to get smaller and cheaper, so it is now-a-days economically feasible to attach many disks to a computer system.
- This brings opportunities for improving the rate at which data can be read or written. Also offers the improving the reliability of data storage.
- The general idea behind RAID is to employ a group of hard drives together with some form of duplication, either to increase reliability or to speed up operations.
- **RAID** originally stood for **Redundant Array of Inexpensive Disks** and was designed to use a group of cheap small disks in place of one or two larger more expensive ones. Today RAID systems are used for higher reliability and higher data transfer rate rather than economic reason hence, switching the definition to **Redundant Array of Independent Disks.**

Improvement of Reliability via Redundancy

- The more disks a system has, the greater the likelihood that one of them will go bad at any given time. Hence increasing disks on a system actually **decreases** the **Mean Time to Failure, MTTF** of the system.
- If, however, the same data was copied onto multiple disks, then the data would not be lost unless both or all copies of the data were damaged simultaneously, which is a **MUCH** lower probability than for a single disk going bad. More specifically, the second disk would have to go bad before the first disk was repaired, which brings the **Mean Time to Repair** into play. For example, if two disks were involved, each with a MTTF of 100,000 hours and a MTTR of 10 hours, then the **Mean Time to Data Loss** would be 500 * 10^6 hours, or 57,000 years!
- This is the basic idea behind disk **mirroring,** in which a system contains identical data on two or more disks.
- Note that a power failure during a write operation could cause both disks to contain corrupt data, if both disks were writing simultaneously at the time of the power failure.
- One solution is to write to the two disks in series, so that they will not both become corrupted by a power failure. And alternate solution involves non-volatile RAM as a write cache, which is not lost in the event of a power failure and which is protected by error-correcting codes.

Improvement in Performance via Parallelism

- There is also a performance benefit to mirroring, particularly with respect to reads. Since every block of data is duplicated on multiple disks, read operations can be satisfied from any available copy, and multiple disks can be reading different data blocks simultaneously in parallel.
- Another way of improving disk access time is with **striping**, which basically means spreading data out across multiple disks that can be accessed simultaneously. Stripping can be done in two ways as :

1. With **bit-level striping** the bits of each byte are striped across multiple disks. For example, if 8 disks were involved, then each 8-bit byte would be read in parallel by 8 heads on separate disks. A single disk read would access 8 * 512 bytes = 4K worth of data in the time normally required to read 512 bytes. Similarly, if 4 disks were involved, then two bits of each byte could be stored on each disk, for 2K worth of disk access per read or write operation.
2. **Block-level striping** spreads a file system across multiple disks on a block-by-block basis, so if block N were located on disk 0, then block N + 1 would be on disk 1, and so on. This is particularly useful when file systems are accessed in **clusters** of physical blocks. Other striping possibilities exist, with block-level striping being the most common.

RAID Levels

- Mirroring provides reliability but is expensive; Striping improves performance but does not improve reliability. Accordingly, there are a number of different schemes that combine the principals of mirroring and striping in different ways, in order to balance reliability versus performance versus cost.
- These are described by different **RAID levels** which are as follows :
- (In the diagram that follows, "C" indicates a copy of the data, and "P" indicates parity, i.e. error correcting or checksum bits.)

1. **Raid Level 0 -** This level includes striping only at the level of blocks, with no mirroring (i.e. no redundancy).
2. **Raid Level 1 -** This level includes mirroring only, no striping.
3. **Raid Level 2 -** It is also known as memory-style error-correcting-code (ECC) organization. This level stores error-correcting codes on additional disks, allowing for any damaged data to be reconstructed by subtraction from the remaining undamaged data. Each byte in the memory system may have a parity bit associated with it that records whether the number of bits in the byte set to 1 (parity = 0) or odd(parity = 1). If one of the bits in the byte is damaged, the parity of byte changes and thus does not match the computed parity. Similarly if the stored parity bits is damaged, it does not match the computed parity. Thus, all single bit errors are detected by the memory system. Note that this scheme requires only three extra disks to protect 4 disks worth of data, as opposed to full mirroring. The number of disks required is a function of the error-correcting algorithms, and the means by which the particular bad bits are identified.
4. **Raid Level 3 -** Also called as bit-interleaved parity organization. This level is similar to level 2, except that it takes advantage of the fact that each disk is still doing its own error-detection, so that when an error occurs, there is no question about which disk in the array has the bad data. Disk controllers can detect whether the sector has been read correctly or not. As a result, a single parity bit is all that is needed to recover the lost data from an array of disks. Level 3 also includes striping, which improves performance. The weakness with the parity approach is that every disk must take part in every disk access, and the parity bits must be constantly calculated and checked, reducing performance. Hardware-level parity calculations and NVRAM cache can help with both of those issues. In practice level 3 is greatly preferred over level 2.
5. **Raid Level 4 -** Also called as block-interleaved parity organization. This level is similar to level 3, employing block-level striping instead of bit-level striping. The benefits are that multiple blocks can be read independently, and changes to a block only require writing two blocks (data and parity) rather than involving all disks. If one of

the disk fails, the parity block can be used with the corresponding blocks from the other disk to restore the blocks of the fail disk. Data transfer rates for each access is slower, but multiple read accesses can continue in parallel, leading to a higher overall, I/O rate. The transfer rate for large reads are high, since all the disks can be read in parallel, large writes also have high transfer rate. Small independent writes cannot be performed in parallel. An O.S write of data smaller than a block requires that the block read, modified with the new data, and written back and parity block must also be updated. This is known as read-modify-write cycle. Note that new disks can be added seamlessly to the system provided they are initialized to all zeros, as this does not affect the parity results.

6. **Raid Level 5 -** Also known as block-interleaved distributed parity. This level is similar to level 4, except the parity blocks are distributed over all disks, thereby more evenly balancing the load on the system. For any given block on the disks, one of the disks will hold the parity information for that block and the other N – 1 disks will hold the data. Note that the same disk cannot hold both data and parity for the same block, as both would be lost in the event of a disk crash and will not be recoverable. By spreading the parity across all the disks this level avoids the potential overuse of a single parity disk.

7. **Raid Level 6 -** Also called as P+Q redundancy scheme. It stores extra information to guard against multiple disk failures. This level extends raid level 5 by storing multiple bits of error-recovery codes, such as the Reed-Solomon codesfor each bit position of data, rather than a single parity bit. In the example shown below 2 bits of ECC are stored for every 4 bits of data, allowing data recovery in the face of up to two simultaneous disk failures. Note that this still involves only 50 % increase in storage needs, as opposed to 100 % for simple mirroring which could only tolerate a single disk failure.

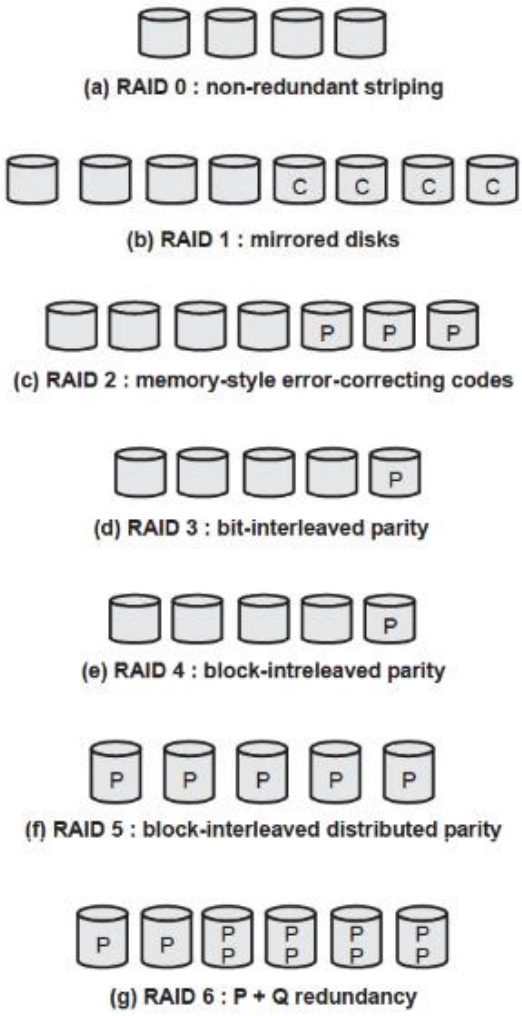


Fig. 6.3.5 RAID levels

Board Questions

1. *Explain two level directory structure with the help of diagram.*
   MSBTE : Winter-15, Summer17, Marks 4
2. *Describe single level and two level directory structures.*
   MSBTE : Winter-16, Marks 4
3. *Explain single level directory structure.*
   MSBTE : Winter-18, Marks 4
4. *Explain the working of two-level directory structure with neat labelled diagram.*
   MSBTE : Summer-18, Marks 4
5. *Explain the RAID structure.*
6. *List all and explain any 2 RAID levels.*
7. *List and explain RAID levels.*

8. *Describe swapping technique with neat diagram*
9. *Explain contiguous allocation with neat diagram.*
10. *Explain linked allocation with neat diagram.*
11. *Explain indexed allocation with neat diagram.*

6.4 Two Marks Questions with Answers

Q.1 *What is a file ? List some operations on it.*

Ans. : File is an unstructured sequence of data, File operations are read, write, create, delete etc.

Q.2 *Write the attributes of a file.*

Ans. : Name, Identifier, Type, Location, Size, Time and date.

Q.3 *What are the two types of system directories ?*

Ans. : A tree structured directory that allows a user to create subdirectories to organize the files.
A general graph structure that allows complete flexibility in the sharing of files and directories.

Q.4 *What are the responsibilities of file manager ?*

Ans. : File manager is responsible for the maintenance of secondary storage. The file manager also provides a logical way for users to organize files in secondary storage.

Q.5 *What is file management system ?*

Ans. : File management system consists of system utility programs that run as privileged applications. The way a user or application may access files and programmer does not need to develop file management software.

Q.6 *What are the disadvantages of log structured file systems ?*

Ans. : It requires cleaning demon to produce clean space, which takes additional CPU time. Reads that are not handled by buffer cache are same performance as normal file system.

Q.7 *Enlist different types of directory structure.*

Ans. : The most common schemes for defining the logical structure of a directory are :

a. Single-level directory
b. Two-level directory
c. Tree-structured directories
d. Acyclic-graph directories
e. General graph directory

Q.8 *List the different operations that can be performed on a file.*

Ans. : Operations

1. Create a file 2. Writing a file
3. Deleting a file 4. Reading a file
5. Repositioning within a file.

Q.9 *Differentiate between file and directory.*

Ans. : The basic difference between the two is that files store data, while directory store files and other directory. File is a sequence of logical records. Directory lists the file by name and includes the file location on the disk, length, type etc.

❑❑❑

Notes

SOLVED SAMPLE TEST PAPER - 1

Operating System

T.Y. Diploma, Sem - V [Computer Engg./IT] [CO/CM/IF/CW]

Time : 1 Hour] **[Marks : 20**

Instructions :

1) *All questions are compulsory.*
2) *Illustrate your answers with neat sketches wherever necessary.*
3) *Figures to the right indicate full marks.*
4) *Assume suitable data, if necessary.*
5) *Preferably, write the answers in sequential order.*

Q.1 Attempt any FOUR **[8]**

a) *Define degree of multiprogramming.* **(Refer Two Marks Q.5 of Chapter-1)**

b) *Write difference between batch system and time sharing system.* **(Refer Two Marks Q.11 of Chapter-1)**

c) *Why API need to be used rather than system calls ?* **(Refer Two Marks Q.3 Chapter-2)**

d) *List different multithreading model.* **(Refer Two Marks Q.12 of Chapter-3)**

e) *What is PCB? Specify the information maintained in it.* **(Refer Two Marks Q.3 of Chapter-3)**

f) *Define context switching.* **(Refer Two Marks Q.1 of Chapter-3)**

Q.2 Attempt any THREE **[12]**

a) *What multiprogramming OS ? Explain.* **(Refer section 1.5.2)**

b) *Explain system view of operating system.* **(Refer section 1.4.2)**

c) *What is Interprocess communication ? List different types of IPC.* **(Refer section 3.3)**

d) *Explain the following Linux command : PS, kill, sleep.* **(Refer section 3.5)**

e) *Write short note on File management and I/O system management.* **(Refer sections 2.3.3 and 2.3.4)**

f) *Define thread? Explain thread life cycle.* **(Refer section 3.4)**

SOLVED SAMPLE TEST PAPER - 2

Operating System
T.Y. Diploma, Sem - V [Computer Engg./IT] [CO/CM/IF/CW]

Time : 1 Hour] **[Marks : 20**

Instructions :

1) *All questions are compulsory.*
2) *Illustrate your answers with neat sketches wherever necessary.*
3) *Figures to the right indicate full marks.*
4) *Assume suitable data, if necessary.*
5) *Preferably, write the answers in sequential order.*

Q.1 Attempt any FOUR **[8]**

a) *What is internal fragmentation ?* **(Refer Two Marks Q.2 of Chapter-5)**

b) *What are memory pages and segments ?* **(Refer Two Marks Q.5 of Chapter-5)**

c) *What is file management system ?* **(Refer Two Marks Q.5 of Chapter-6)**

d) *Enlist different types of directory structure.* **(Refer Two Marks Q.7 of Chapter-6)**

e) *What is Banker's algorithm?* **(Refer Two Marks Q.17 of Chapter-4)**

f) *What is aging ?* **(Refer Two Marks Q.7 of Chapter-4)**

Q.2 Attempt any THREE **[12]**

a) *What is FCFS algorithm? Describe with example.* **(Refer section 4.2.1)**

b) *Explain different free space management techniques.* **(Refer section 5.4)**

c) *List different CPU scheduling criteria.* **(Refer section 4.1.3)**

d) *Write short note on RAID structure.* **(Refer section 6.3.6)**

e) *What is virtual memory ? Explain.* **(Refer section 5.5)**

f) *Define optimal page replacement? Explain with example.* **(Refer section 5.8.4)**

❑❑❑

SOLVED SAMPLE QUESTION PAPER

Operating System

T.Y. Diploma, Sem - V [Computer Engg./IT] [CO/CM/IF/CW]

Time : 3 Hours] **[Marks : 70**

Instructions : *1) All questions are compulsory.*

2) Illustrate your answers with neat sketches wherever necessary.

3) Figures to the right indicate full marks.

4) Assume suitable data, if necessary.

5) Preferably, write the answers in sequential order.

Q.1 Attempt any FIVE of the following. **[10]**

a) Define operating system. **(Refer Two Marks Q.1 of Chapter-1)**

b) Define real time system. **(Refer Two Marks Q.9 of Chapter-1)**

c) What is system call ? **(Refer Two Marks Q.5 of Chapter-2)**

d) What are the benefits of multithreads ? **(Refer Two Marks Q.5 of Chapter-3)**

e) What is response time ? **(Refer Two Marks Q.4 of Chapter-4)**

f) What is concept of paging ? **(Refer Two Marks Q.1 of Chapter-5)**

g) Differentiate between file and directory. **(Refer Two Marks Q.9 of Chapter-6)**

Q.2 Attempt any THREE of the following. **[12]**

a) Explain user view of operating system. **(Refer section 1.4.1)**

b) Explain Andriod architecture with diagram. **(Refer section 1.9.1.1)**

c) Describe difference between symmetric and asymmetric multiprocessor. **(Refer section 1.6.4)**

d) What is GUI based OS? Draw windows architecture in detail. **(Refer section 1.11)**

Q.3 Attempt any THREE of the following. **[12]**

a) Explain the various types of system calls with an example for each. **(Refer section 2.2)**

b) Define Operating System. List functions of operating system. **(Refer section 1.1)**

c) Explain different services provided by operating system. **(Refer section 2.1)**

d) Explain OS components as process management. **(Refer section 2.3.1)**

Q.4 Attempt any THREE of the following. **[12]**

a Define deadlock. What are the necessary conditions for deadlock ? **(Refer section 4.3)**

b Explain difference between user level and kernel level thread. **(Refer section 3.4.6)**

c List the features of message passing system. **(Refer section 3.3.3)**

d Write steps for Banker's algorithm to avoid deadlock. **(Refer section 4.5.3.1)**

e What is priority scheduling? Explain with example. **(Refer section 4.2.3)**

Q.5 Attempt any TWO of the following. **[12]**

a *What is Belady's anomaly ? Explain with example.* **(Refer section 5.8.1.1)**

b *What is paging ? Explain paging hardware with diagram.* **(Refer section 5.6)**

c *What is fragmentation ? Explain different types of fragmentation.* **(Refer section 5.3.2)**

Q.6 Attempt any TWO of the following. **[12]**

a *Describe single level and two level directory structures.* **(Refer sections 6.3.1 and 6.3.2)**

b *Explain any two allocation methods.* **(Refer section 6.2.4)**

c *i) List different file operations.* **(Refer section 6.1.4)**

ii) Write short note on swapping. **(Refer section 6.2.3)**

❑❑❑