DECODE®

A Guide For Engineering Students

OPERATING SYSTEMS

SUBJECT CODE : 217521

S.E. (Artificial Intelligence & Data Science) Semester - III

Published by :

TECHNICAL® PUBLICATIONS
SINCE 1993 — An Up-Thrust for Knowledge

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

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

ISBN 978-93-5585-227-4

9 789355 852274

SPPU 20

9789355852274 [1]

SYLLABUS

Operating Systems - (217521)

Credit	Examination Scheme :
03	Mid_Semester (TH) : 30 Marks End_Semester (TH) : 70 Marks

Unit I Fundamental Concepts of Operating system

Operating system functions and characteristics, historical evolution of operating systems, issues in operating system design, User's view of the OS, Types of OS : Batch, time sharing, multiprogramming, distributed, network and real-time systems, Operating-System Services, Types of System Calls, System Programs. BASH Shell scripting : Basic shell commands, shell as a scripting language. **(Chapter - 1)**

Unit II Process Management

Process concept, Process Control Block(PCB), Process Operations, **Process Scheduling :** Types of process schedulers, Types of scheduling : Preemptive, Non preemptive. Scheduling algorithms : FCFS, SJF, RR, Priority, Inter process Communication(IPC). **Threads :** multithreaded model, implicit threads, threading issues. **(Chapter - 2)**

Unit III Process Coordination

Synchronization : Principles of Concurrency, Requirements for Mutual Exclusion, Mutual Exclusion : Hardware Support, Operating System Support (Semaphores and Mutex), Programming Language Support (Monitors). Classical synchronization problems : Readers/Writers Problem, Producer and Consumer problem, Inter-process communication (Pipes, shared memory : system V) **Deadlock :** Deadlock Characterization, Methods for Handling Deadlocks, Deadlock Prevention, Deadlock Avoidance, Deadlock Detection, Recovery from Deadlock. **(Chapter - 3)**

Unit IV Memory Management

Memory Management : Memory Management Requirements, Memory Partitioning : Fixed Partitioning, Dynamic Partitioning, Buddy System, Relocation, Paging, Segmentation. Virtual Memory : Hardware and Control Structures, Operating System Software. **(Chapter - 4)**

Unit V I/O and File Management

I/O Management : I/O Devices, Organization of I/O function, I/O Buffering, Disk SchedulingDisk Scheduling policies like FIFO, LIFO, STTF, SCAN, C-SCAN. File Management : Concept, Access methods, Directory Structure, Protection, File System implementation, . Directory Implementation, Allocation methods, Free Space management. **(Chapter - 5)**

Unit VI Linux

History Of Unix and Linux , Overview Of Linux - Linux Goals, Interfaces to Linux, The Shell, Linux Utility Programs, Kernel structure, Processes in Linux - Process management system calls in Linux, Implementation of process and threads in Linux, Process scheduling Linux, Booting. **(Chapter - 6)**

TABLE OF CONTENTS

Unit I

Unit II

Unit III

Unit IV

Unit V

Unit VI

Unit I

1 Fundamental Concepts of Operating System

1.1 : Operating System Functions and Characteristics

Q.1 What is operating system ? State and explain the basic functions of operating system.

Ans. : • **OS definition :** *Operating System is a program that controls the execution of application programs. It is an interface between applications and hardware.*

- OS provides different types of view. For user, it is abstract view because it provides features which are important for users. OS is intermediary between user and the computer system.
- The major design goals/ functions of an operating system are :
  1. Efficient use of a computer system
  2. User convenience
  3. Ability to evolve
- An operating system is software that manages the computer hardware. The hardware must provide appropriate mechanisms to ensure the correct operation of the computer system and to prevent user programs from interfering with the proper operation of the system.

Efficient use

- For efficient use of resources, it must be monitored by operating system. Proper scheduling of resources is also required.
- Computer contains different type's resources like CPU, memory and I/O device etc. Proper monitoring is required on these resources to avoid the overhead. As per the resource, scheduling is required.

- Special attention to be given for CPU and memory. If memory is not free then user can not load new program into the memory. Then CPU will be busy with memory management.

User convenience

- User convenience is affected by computing environment of the computer system. The computing environment is comprised of computer system, its interfaces with other systems and nature of computations performed by its users.
- Computer architecture and use change the computing environment of the system. Following factors are considered while considering user convenience :
  1. Good service
  2. Ease of use
  3. New programming model
  4. Evolution
  5. User friendly OS.

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.

Q.2 Describe in detail the functions of OS as a resource manager.

Ans. : • 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. Q.2.1 shows 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.

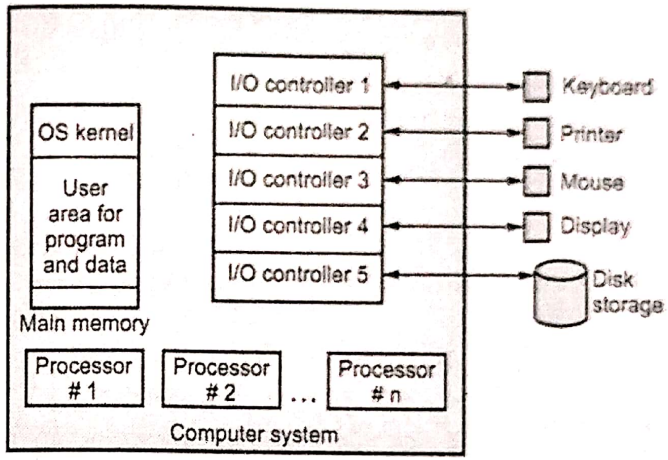


Fig. Q.2.1 OS as a resource manager

- 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.

Q.3 What are the benefits of resource abstraction ?

Ans. : • Resource abstraction is the process of "hiding the details of how the hardware operates, thereby making computer hardware relatively easy for an application programmer to use".

- It provide a single abstract disk interface which will be the same for both the hard disk and floppy disk.
- Saves the programmer from needing to learn the details of both hardware interfaces.

- While making the hardware easier to use, resource abstraction also limits the specific level of control over the hardware by hiding some functionality behind the abstraction.
- Since most application programmers do not need such a high level of control, the abstraction provided by the operating system is generally very useful.

1.2 : Historical Evolution of Operating Systems

Q.4 Discuss evolution of operating systems.

Ans. : Operating systems have been evolving through the years. Following table shows the history of OS.

Generation	Year	Electronic devices used	Types of OS and devices
First	1945-1955	Vacuum tubes	Plug boards
Second	1955-1965	Transistors	Batch system
Third	1965-1980	Integrated circuit (IC)	Multiprogramming
Fourth	Since 1980	Large scale integration	PC

1.3 : Types of OS

Q.5 Describe in brief the evolution of operating system.

Ans. : • The evolution of operating systems is directly dependent to the development of computer systems and how users use them. OS created to process jobs in batches. Later Multitasking and Time-Sharing created to run multiple jobs and allow user interaction to improve efficiency. Multitasking brought challenges to manage I/O operations required by multiple jobs in which computer vendors resolved with interrupts.

1. Batch 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.

2. Multiprogramming 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. Q.5.1 shows the memory layout for a multiprogramming operating system.
- Multiprogramming operating system do not provide user interaction with the program.
- Fig. Q.5.2 Shows working of multiprogramming OS.

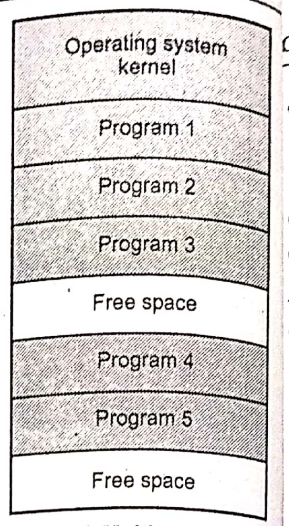


Fig. Q.5.1 Multiprogramming OS memory layout

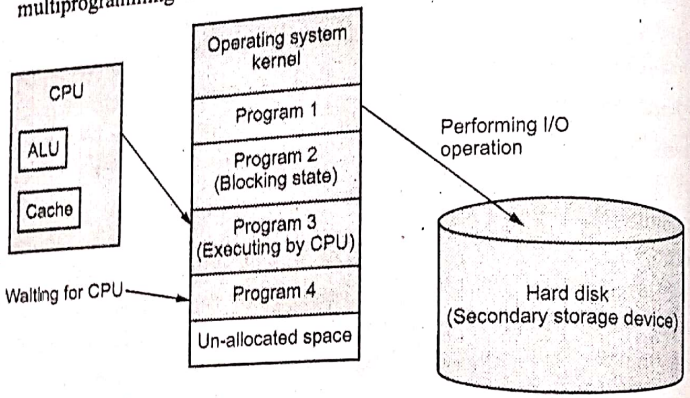


Fig. Q.5.2 Working of multiprogramming OS

- 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.

Q.6 Explain time sharing system. Also explain advantages and disadvantages.

Ans. : • In an interactive system, many users directly interact with the computer from terminals connected to the computer.

- Processor's time which is shared among multiple users simultaneously is termed as time-sharing.
- Time sharing is logical extension of multiprogramming OS. Fig. Q.6.1 shows time sharing system.

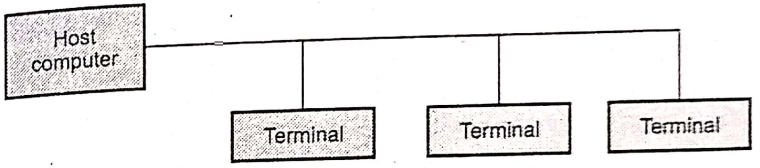


Fig. Q.6.1 Time sharing

- Time sharing is a method that allows multiple users to share resources at the same time. Multiple users in various locations can use a specific computer system at a time
- Time sharing is essentially a rapid time division multiplexing of the processor time among several processes. The processor switching is so frequent that it almost seems each process has its own dedicated processor.
- Time sharing OS is designed to provide a quick response to sub-requests made by users.
- The processor time is shared between multiple users at a time. The processor allows each user program to execute for a small time quantum. Moreover, time sharing systems use multiprogramming and multitasking.

- The operating system provides immediate feedback to the user and response time can be in seconds, depending on the number of active users.
- The scheduling technique used by a time-sharing kernel is called round robin scheduling with time slicing.
- If the time slice elapses before the process completes servicing of a sub-request, the kernel pre-empts the process, moves it to the end of the scheduling queue and schedules another process.
- The main difference between Multi-programmed Batch Systems and Time-Sharing Systems is that in case of Multi-programmed batch systems, the objective is to maximize processor use, whereas in Time Sharing Systems, the objective is to minimize response time.
- **Features of time sharing system :**
  1. User interaction with program is possible.
  2. Time sharing system uses medium term scheduling such as round robin for the foreground.
  3. Each user is given a time slice for executing job in round robin fashion.
- **Advantages of time sharing operating systems :**
  1. Provides the advantage of quick response.
  2. Avoids duplication of software.
  3. Reduces CPU idle time.
- **Disadvantages of time sharing operating systems :**
  1. Problem of reliability.
  2. Question of security and integrity of user programs and data.
  3. Problem of data communication.

Q.7 What do you mean spooling ? Explain in detail.

Ans. : • When a job completes execution, its memory is released and the output for the job gets copied into an outputs pool for later printing.

• 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 same page, getting their output completely mixed together. It also helps in reducing idle time and overlapped I/O and CPU.
- It 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.
- In spooling, I/O of one job is overlapped with the computation of another job. For example, a spooler at a time may read input of one job and at the same time, it may also print the output of another job.
- Spooling can also process data at the remote sites. The spooler only has to notify when a process gets completed at the remote site so that spooler can spool next process to the remote side device.
- Spooling increases the performance of the system by increasing the working rate of the devices. It naturally leads to multiprogramming.

Advantages of spooling

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

Q.8 Give the features of real time operating system and time sharing operating system.

Ans. : Features of Real Time Operating System :

1. Real time system fails if it does not give result within the time limits.
2. Real time task cannot keep waiting for longer time without allocating kernel.
3. Real time operating system uses priority scheduling algorithm

Features of Time Sharing Operating System :

1. User interaction with program is possible in time sharing operating system.
2. Time sharing system uses medium term scheduling such as round robin for the foreground.
3. In time sharing system, each user is given a time slice for executing his/her job in round robin fashion.

Q.9 Explain distributed OS with neat sketch and give its pros and cons.

Ans. : • 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 uniprocessor 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.

- Fig. Q.9.1 shows the distributed system.

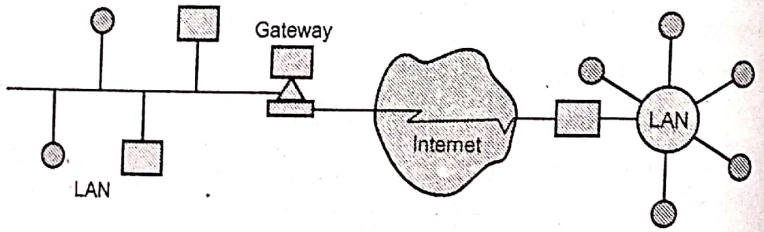


Fig. Q.9.1 Distributed system

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 aspect of reliability. Availability refers to the fraction of time for which a system is available for use. Availability of a hard disk can be increased by having multiple hard disks located at different sites. If one hard disk fails or is unavailable, the program can use some other hard disk.
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.

- Examples of distributed operating system are amoeba, chrous, mach and v-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.

Q.10 Explain the following : Multitasking.

Ans. : • 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 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. Q.10.1 shows multitasking OS.

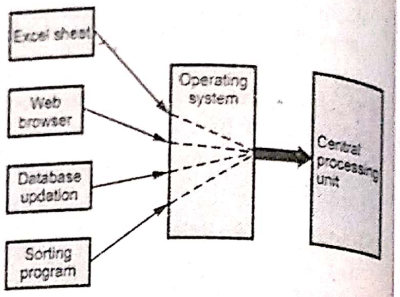


Fig. Q.10.1 Multitasking OS

- 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 users 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.

Q.11 What is real time OS ? Explain its types with suitable example.

Ans. : • 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.
- A real-time system is any information processing system which has to respond to externally generated input stimuli within a finite and specified period.
- Real time systems are of two types : Hard and soft

1. Hard Real Time Systems :

- A hard real-time system is one where the response time is specified as an absolute value. This time is normally dictated by the environment.
- A system is called a hard real-time if tasks always must finish execution before their deadlines or if message always can be delivered within a specified time interval.
- Hard real-time is often associated with safety critical applications. A failure (e.g. missing a deadline) in a safety-critical application can lead to loss of human life or severe economical damage.

2. Soft Real Time System

- A soft real-time system is one where the response time is normally specified as an average value. This time is normally dictated by the business or market.
- A single computation arriving late is not significant to the operation of the system, though many late arrivals might be.
- Soft real time means that only the precedence and sequence for the task-operations are defined, interrupt latencies and context switching latencies are small but there can be few deviations

between expected latencies of the tasks and observed time constraints and a few deadline misses are accepted

1.4 : Operating-System Services

Q.12 State and explain different services provided by an operating system.

Ans. :

1. **Program execution :** Before executing the program, it is loaded into the memory by operating system. Once program loads into memory, its start execution.
2. **Input - output operation :** Program is combination of input and output statement. While executing the program, it requires I/O device.
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.
5. **Communication :** Communication may be inter-process communication and any other type of communication. Data transfer is one type of communication.

1.5 : Types of System Calls

Q.13 What is system call ? Explain working of system call.

Ans. : • 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. Q.13.1 shows the working of system call.

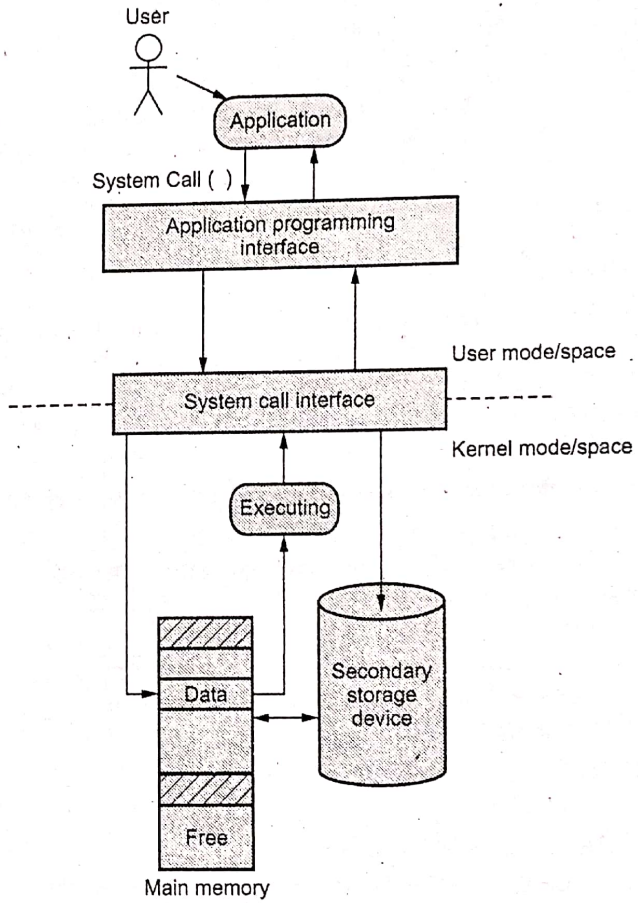


Fig. Q.13.1 Working of system call

- 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.

Q.14 Explain three general method for passing parameters to the OS.

Ans. : 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.

Q.15 Discuss types of system calls.

Ans. : • 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.

1.6 : System Programs

Q.16 Write short note on system programs.

Ans. : • Modern operating system consists of a collection of system programs. System program that provides an application programming environment on top of the hardware. Some of them are simply user interfaces to system calls. They can be divided into these categories :

1. File management
2. Status information
3. File modification
4. Programming language support
5. Program loading and execution
6. Communications.

- File management programs are used to create, delete, copy, rename, list, dump on files and directory. Status information system programs covers the date, time, disk space, available memory and users. All this information is formatted and displayed on output device or printed.
- Text editors are used for file modification. In this, new file is created and content of file is modified. Programming language support includes the compilers, assemblers, interpreters for common programming language like C, C++, Java, Visual Basic.
- For program loading and execution, it is loaded into memory then program is executed by processor. Operating system provides different types loaders and linkers to complete execution operation.
- Debugging facility is provided by the operating system with the help of application program. It is used for checking errors.
- Communication between two devices are performed by creating temporary connection. Communication is in between process, users and other I/O devices. File transfer, remote login, electronic mail, browsing web, downloading multimedia data are the example of communication.

1.7 : BASH Shell Scripting

Q.17 What is Linux BASH shell ?

Ans. : • The Linux Bash is also known as 'Bourne-again Shell.' It is a command language interpreter for the Linux based system. It is a replacement of Bourne shell (sh).

- The Linux/Unix shell allows us to interact with the Linux system through the commands. It let us invoke an executable file to create a running process.

- It also allows us to interact with the Linux file system. It is designed in such a way that we can perform all the Linux operations through Bash.
- The Bash is a command language interpreter as well as a programming language. It supports variables, functions, and flow control, like other programming languages. It can also read and execute the commands from a file, which is called a shell script

Q.18 Explain the following shell commands with example.
i) Chmod ii) Grep iii) Cat iv) Sort.

Ans. : i) Chmod : Access permissions associated with a file or directory can be changed using the chmod command. Permissions associated with a file can be changed only by the owner of the file.

- Consider a file prime.c with the following permissions :

.rwxrwxrwx 1 rakshita student 20 Jan 1 11:20 prime.c

- To change the permissions for the file, the File Owner has to specify :
  1. The type of user for whom permission is to changed.
  2. The type of permission which is to be changed.
  3. Whether the permission is to be given or revoked.
  4. The name of the file for which permissions are to be changed.

UNIX allows you to change the FAP for a specific user - type. The command to do that is :

```
$ chmod u + r prime.c < Enter>
$
```

- Here,

  'u' = indicates File Owner,

  '+' = indicates that the permission is to be given,

  'r' = indicates the read permission and prime.c is the filename.

ii) Grep : It stands for "Get Regular Expression and Print". The grep command allows user to search one file or multiple files for lines that contain a pattern. The syntax is

gerp [options] pattern [files]

iii) Cat :

- The **cat** command will also display file contents to a screen file contents to a screen.

Syntax : **cat** filename

Also, use **cat** for quickly creating a short file, by entering the following.

```
$ cat > filename
```

After pressing the return key, type in the text. To save and exit the file, press **CTRL-D**.

An additional feature of **cat is** that it allows you to concatenate two files together by entering the folllowing :

```
$ cat file 1 >> file 2
```

This command appends file 1 to end of file 2.

iv) Sort :

- The **sort** utility sorts the lines in a file. By default, it sorts the file in ASCII order, with numbers preceding alphabetic characters.

Syntax : **sort** filename

- You can also use the **sort** command in combination with other commands using pipes. To sort the output from a who command, enter :

```
$ who | sort
```

Q.19 Explain the significance of following shell commands
i) in ii) wc iii) umask iv) cut v) grep

Ans. : i) in : • The in command is used to create links. Links are a kind of shortcuts to other files. The general form of command is :

$ ln TARGET LINK_NAME

- There are two types of links, soft links and hard links. By default, hard links are created. If you want to create soft link, use -s option. In this example, both types of links are created for the file usrlisting.

```
$ ln usrlisting hard_link
$ ln -s usrlisting soft_link
$ ls -l
total 12
```

-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file1

-rw-r--r-- 1 raghu raghu 0 2012-07-06 14:20 file3

-rw-r--r-- 2 raghu raghu 491 2012-07-06 14:23 hard_link

lrwxrwxrwx 1 raghu raghu 10 2012-07-09 14:00 soft_link ->usrlisting

-rw-r--r-- 1 raghu raghu 491 2012-07-06 16:02 usrcopy

-rw-r--r-- 2 raghu raghu 491 2012-07-06 14:23 usrlisting

ii) wc

wc : Counting the number of characters, words and lines

Syntax : wc [-c/l/w] file....

Options :

-w : Gives the word count

-I : Counts the number of lines

-c : Count the number of characters.

If given without any parameters the command displays all the three counts.

iii) umask

- umask is a number which defines the default permissions which are not to be given on a file. A umask of 022 means not to give the write permission to the group(022) and others(022) by default.
- The umask command automatically sets the file permissions upon creation of the file.
- The command changes initial Permission of newly created file.
- The value of the argument can be calculated by subtracting the mode you want as default from the current default mode.
- Assume the current default mode is 0666 and you want it as 0644 then 666 - 644 = 022 will be the parameter which we have to pass with umask command.

$umask 0 - sets default mode which is 0666

iv) cut

- Cut command in unix (or linux) is used to select sections of text from each line of files. You can use the cut command to select fields or

columns from a line by specifying a delimiter or you can select a portion of text by specifying the range or characters.

- Basically the cut command slices a line and extracts the text.

$ cut -[cfd] [filename]

Where -c characters

-f field no

-d field separator

$ cut -c2-5 sample

cuts characters from 2 to 5 from file sample.

v) grep : Refer Q.14.

Q.20 Explain the following shell commands with example : i) echo ii) grep iii) touch iv) ls.

Ans. :

i) **echo :** echo is a built-in command in the bash and C shells that writes its arguments to standard output. The syntax for echo is :

echo [option(s)] [string(s)]

- A string is any finite sequence of characters (i.e., letters, numerals, symbols and punctuation marks)

ii) **grep :** Refer Q.18.

iii) **touch :** The touch command allows us to update the timestamps on existing files and directories as well as creating new, empty files. If user are a Terminal - savvy person, user can quickly create a new file in the command line through the following command :

$ touch "filename"

iv) **ls :** ls lists the contents of a directory. If no target directory is given, then the contents of the current working directory are displayed. So, if the current working directory is /,

$ ls <Enter>

- Actually, ls doesn't show you all the entries in a directory - Files and directories that begin with a dot (.) are hidden. The reason for this is that files that begin with a. usually contain important configuration information and should not be changed under normal circumstances.

Q.21 State command line arguments in shell with example.

Ans. : • Arguments or variables may be passed to a shell script. Simply list the arguments on the command line when running a shell script. In the shell script, $0 is the name of the command run (usually the name of the shell script file); $1 is the first argument, $2 is the second argument, $3 is the third argument, etc.

• The variable $# contains the number of command line arguments that were supplied and the variable $* contains all the arguments at once. The variable $# reports the number of command line arguments passed to the shell script program.

Example : Create a shell script to print all argument with script name and total number of arguments passed. Create script file commline.sh using following content.

```
# vim commline.sh
#!/bin/bash
echo Script Name: "$0"
echo Total Number of Argument Passed: "$#"
echo Arguments List -
echo 1. $1
echo 2. $2
echo 3. $3
echo All Arguments are: "$*"
```

END...

Unit II

2 Process Management

2.1 : Process Concept

Q.1 Define process. Differentiate between a process and a program.

Ans. : • 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.

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.

Q.2 How PCB helps in process state management ? Explain the structure of PCB.

Ans. : 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. Q.2.1 shows process control block.

- Process control block will change according to the operating system. PCB is also called **task control block**.
- 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.

Process identification
Priority number
Program counter
Memory allocation
I/O status information
List of open files
Accounting information
Number of registers
Process state

Fig. Q.2.1 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.

Q.3 Draw and explain process state diagram.

Ans. : • 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. Q.3.1 shows a process state diagram.

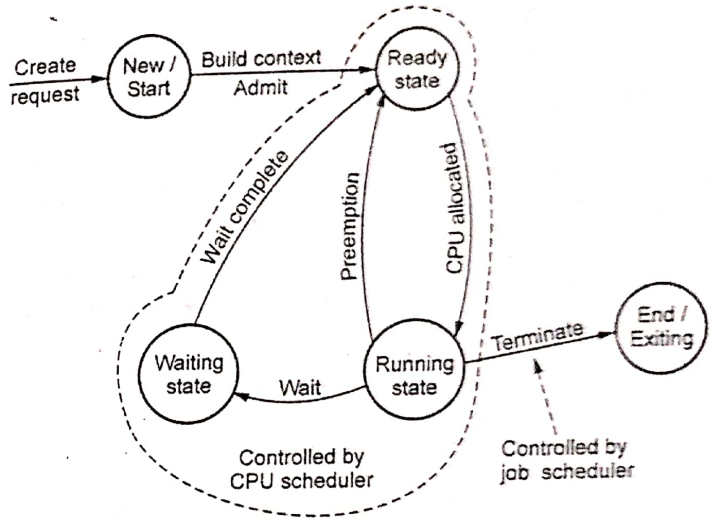


Fig. Q.3.1 Process state diagram

- The process states are as follows :
  1. **New :** Operating system creates new process by using fork( ) system call. These process are newly created process and resources are not allocated.
  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. Operating system allocates all the hardware and software resources to the process for execution.

4. **Blocked / Waiting :** A process is waiting until some event occurs such as the completion of an input-output operation.
5. **Exit / End :** A process is completes its operations and releases it all resources.

2.2 : Process Operations

Q.4 Explain process creation process.

Ans. : • Operating system creates the process in following situations :

1. Starting of new batch job.
2. User request for creating new process.
3. To provide new services by OS.
4. System call from currently running process.

- Operating system creates a new process with the specified or default attributes and identifier. A process may create several new sub-process.
- Parent process is creating process and the new processes are called the children of the process. When operating system creates process, it builds the data structure for managing process and allocates address space in primary main memory.
- Operating system creates foreground and background process. Process is identified by unique process identifier (PID) in UNIX and windows operating system. PID value is an integer number.
- All processes in UNIX are created using the fork( ) system call. The forking process is called the parent process. The new process is called the child process.
- Both the parent and child process have their own and private memory. Open files are shared between parent and child.
- If the parent changes the value of its variable, the modification will only affect the variable in the parent process's address space. Other address spaces created by fork( ) calls will not be affected even though they have identical variable names.

- When a process is created, OS assigns some attributes. These are priority, privilege level, requirement of memory, access right, memory protection, PID etc. To perform operation, process needs software and hardware resources. It includes CPU time, files, memory, I/O device.
- Relation between parent process and child process is as follows :
  1. Parent process continues to execute concurrently with its child process.
  2. Parent process waits until some or all of its children have terminated.

```
void main( )
{
    printf("Operating System\n");
    fork( );
      printf("Technical Publications\n");
    return 0;
}
```

- In above program Operating System is printed only once and Technical Publications is printed two times.

Q.5 List the four events that cause processes to be created. Explain each in brief.

Ans. : There are four events that cause processes to be created they are :

1. System initialization (Process creation at boot time),
2. Execution of a process creation system call by a running process,
3. A user request to create a new process,
4. Initiation of a batch job.

- Several processes are created when the computer is booted.
- Process Creation System Call : A running process issue system calls to create one or more new processes to help it do its job. When the work can be divided between several related, but otherwise independent interacting processes. For example, a process fetches data from the internet and stores it in a shared buffer, another process, processes this data.

Q.6 Explain various reasons of process termination.

Ans. : • When process finishes its normal execution then that process is terminate. Operating system delete that process using exit ( ) system call. After deleting process, memory space becomes free.

• OS passes the child's exit status to the parent process and then discards the process. At the same time, it de-allocate all the resources hold by this process.

• Following are the various reasons for process termination :

1. Normal completion of operation
2. Memory is not available
3. Time slice expired.
4. Parent termination
5. Failure of I/O
6. Request from parent process
7. Misuse of access rights

2.3 : Process Scheduling

Q.7 Define the following :

a) Scheduling queue b) Device queue c) Ready queue

Ans. : a) 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.

b) 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.

c) 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.

Q.8 Explain different types of schedulers in operating system.

Ans. : • 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. Q.8.1 shows queueing diagram for process scheduling.

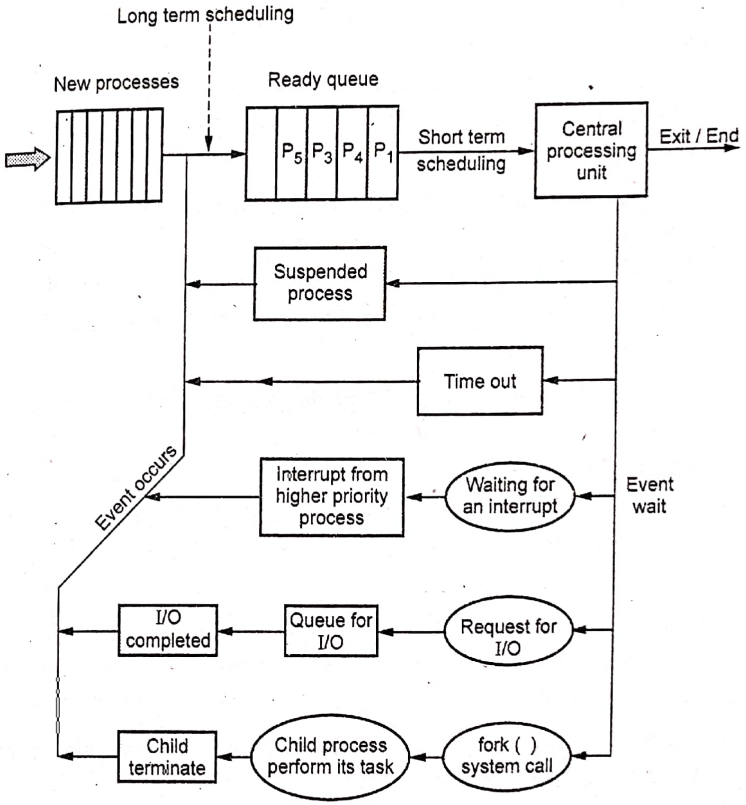


Fig. Q.8.1 Process scheduling queueing diagram

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.

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.

Q.9 Explain concept of context switching with the help of neat diagram.

Ans. : • Fig. Q.9.1 shows context switching.

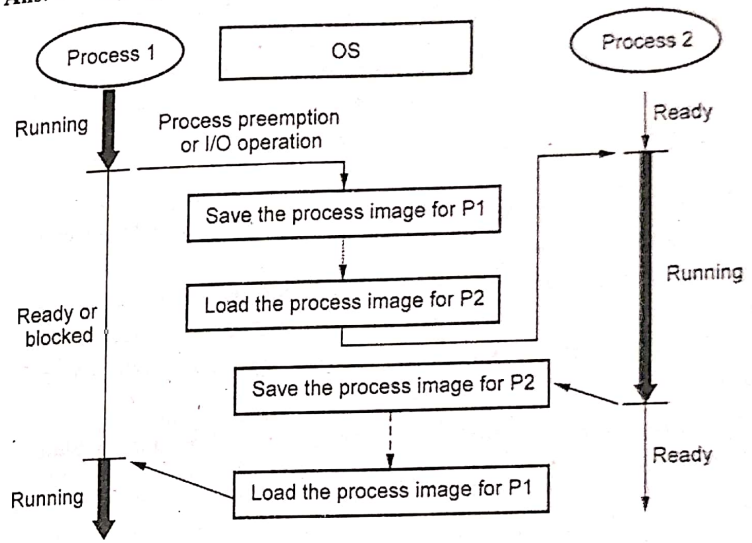


Fig. Q.9.1 Context switching

- 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.

Q.10 Explain difference between long term, short term and medium term scheduler.

Ans. :

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 multiprogramming.	Less control over degree of multiprogramming.	Reduce the degree of multiprogramming.
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.	–

2.4 : Types of Scheduling

Q.11 What should be scheduling criteria for scheduling algorithm ?

Ans. : 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.

Q.12 Explain difference between preemptive and non-preemptive CPU scheduling.

Ans. :

Sr. No.	Preemptive scheduling	Nonpreemptive 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.	Nonpreemptive 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.	Nonpreemptive 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.

Q.13 Define term scheduler, scheduling and scheduling algorithm with example.

Ans. : • **Scheduler :** Schedulers are used to handles process scheduling. It is one type of system software and selects the jobs from the system and decides which process to run process of selecting processes from among these queues is carried out by a scheduler.

- **Scheduling :** In a multiprogramming environment, usually more programs to be executed than could possibly be run at one time. In

CPU scheduling, it switches from one process to another process. CPU resource management is commonly known as scheduling.

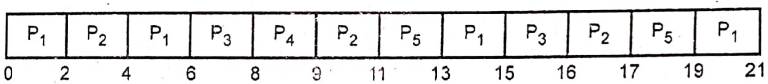
Scheduling Algorithms : Scheduling algorithms or scheduling policies are mainly used for short-term scheduling. The main objective of short-term scheduling is to allocate processor time in such a way as to optimize one or more aspects of system behavior. Scheduling algorithms decide which of the processes in the ready queue is to be allocated to the CPU is basis on the type of scheduling policy and whether that policy is either preemptive or non-preemptive. For scheduling arrival time and service time are also will play a role.

2.5 : Scheduling Algorithms

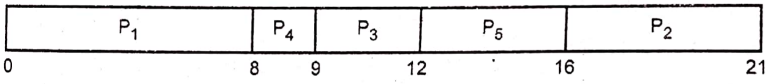
Q.14 For the table given below calculate average waiting time and average turnaround time and draw a Gantt Chart illustrating the process execution using following scheduling algorithms.
i) RR (Time slice - 2 units) ii) SJF (non - preemptive)

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	5
P_3	3	3
P_4	4	1
P_5	6	4

Ans. : i) RR (time slice 2 units) : Gantt chart



ii) SJF (non-preemptive) : Gantt chart



Waiting time and turnaround time :

Process	Waiting time		Turnaround time	
	RR	SJF	RR	SJF
P_1	0 – 0 + 4 – 2 + 13 – 6 + 19 – 15 = 13	0 – 0 = 0	21	8
P_2	2 – 1 + 9 – 4 + 16 – 11= 11	16 – 1 = 15	16	20
P_3	6 – 3 + 15 – 8 = 10	9 – 3 = 6	13	9
P_4	8 – 4 = 4	8 – 4 = 4	5	5
P_5	11 – 6 + 17 – 13 = 9	12 – 6 = 6	13	10

Average waiting time for RR $= \frac{13+11+10+4+9}{5} = 9.4$

Average waiting time for SJF $= \frac{0+15+6+4+6}{5} = 6.2$

Average turnaround time for RR $= \frac{21+16+13+5+13}{5} = 13.6$

Average turnaround time for SJF $= \frac{8+20+9+5+10}{5} = 10.4$

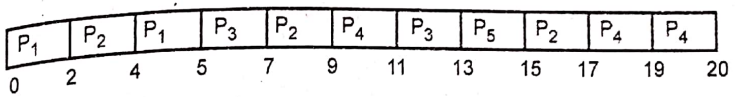
Q.15 For the table given below, calculate average waiting time and average turnaround time and draw a Gantt chart illustrating the process execution using following scheduling algorithms.
i) Round robin (time slice - 2 units) ii) Priority (non-preemptive)

Process	Arrival Time	Burst Time	Priority
P_1	0	3	5
P_2	2	6	2
P_3	4	4	4
P_4	6	5	3
P_5	8	2	1

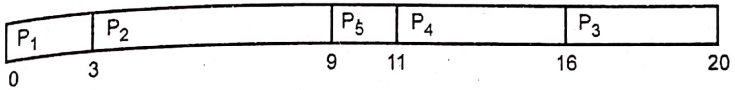
Note : For priority scheduling, minimum value indicates higher priority.

Ans. : Gantt chart :

i) Round robin (time slice = 2 units)



ii) Priority (non - preemptive)



Waiting time and turnaround time :

Process	Waiting time		turnaround time	
	Round robin	Priority (non-preemptive)	Round robin	Priority (non-preemptive)
P_1	0 – 0 + 4 – 2 = 2	0 – 0 = 0	5	3
P_2	2 – 2 + 7 – 4 + 15 – 9 = 9	3 – 2 = 1	15	21
P_3	5 – 4 + 11 – 7 = 5	16 – 4 = 12	9	13
P_4	9 – 6 + 17 – 11 = 11	11 – 6 = 5	16	21
P_5	13 – 8 = 5	9 – 8 = 1	7	9

Average waiting time of Round Robin

= (2 + 9 + 5 + 11 + 5) / 5 = 6.4

Average waiting time of Priority (non-preemptive)

= (0 + 1 + 12 + 5 + 1) / 5 = 3.8

Average turnaround time of Round Robin

= (5 + 15 + 9 + 16 + 7) / 5 = 10.4

Average turnaround time of Priority (non-preemptive)

= (3 + 21 + 13 + 21 + 9) / 5

= 13.4

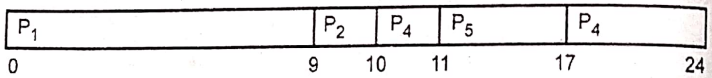
Q.16 For the table given below, calculate average waiting time and average turnround time and draw a Gantt chart illustrating the process execution using following scheduling algorithms.
i) SJF (non-preemptive) ii) Priority (preemptive)

Process	Arrival Time	Burst Time	Priority
P_1	0	9	3
P_2	1	1	2
P_3	2	7	1
P_4	3	1	5
P_5	4	6	4

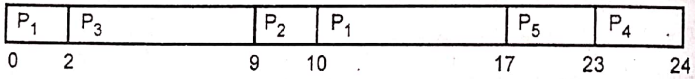
Note : For priority scheduling, minimum value indicates higher priority.

Ans. : Gantt Chart :

i) SJF (non - preemptive)



ii) Priority (Preemptive)



Waiting time and turnaround time :

Process	Waiting time		Turnaround time	
	SJF (non-preemptive)	Priority (Preemptive)	SJF (non-preemptive)	Priority (Preemptive)
P_1	0 – 0 = 0	0 – 0 + 10 – 2 = 8	9	17
P_2	9 – 1 = 8	9 – 1 = 8	9	9
P_3	17 – 2 = 15	2 – 2 = 0	22	7
P_4	10 – 3 = 7	23 – 3 = 20	8	21
P_5	11 – 4 = 7	17 – 4 = 13	13	20

Average waiting time of SJF (non-preemptive)

= (0 + 8 + 15 + 7 + 7) / 5 = 7.4

Average waiting time of Priority (preemptive)

= (8 + 8 + 0 + 20 + 13) / 5 = 9.8

Average turnaround time of SJF (non-preemptive)

= (9 + 9 + 22 + 8 + 13) / 5 = 12.2

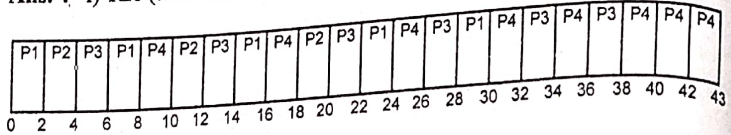
Average turnaround time of Priority (preemptive)

= (17 + 9 + 7 + 21 + 20) / 5 = 14.8

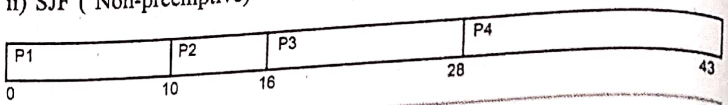
Q.17 For the table given below calculate the average waiting time and average turn around time and draw a Gantt chart illustrating the process execution using following scheduling algorithms :
i)RR (Time slice -2 units) ii) SJF (non-preemptive)

Process	Arrival time	Burst time
P1	0	10
P2	1	06
P3	2	12
P_4	3	15

Ans. : i) RR (time slice = 2 units)



ii) SJF (Non-preemptive)



Process	Waiting time RR	Waiting time SJF
P1	20	0
P2	13	9
P3	24	14
P4	25	25
Average	= 79/4	= 48/4
	= 19.75	= 12

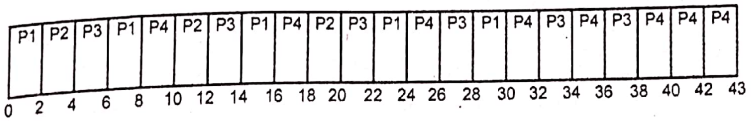
Process	Turnaround time RR	Turnaround time SJF
P1	30	10
P2	19	15
P3	36	26
P4	40	40
	= 125/4	= 91/4
	= 31.25	= 22.75

Q.18 For the table given below calculate the average waiting time and average turn around time and draw a Gantt Chart illustrating the process execution using following scheduling algorithms.

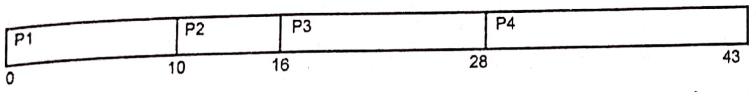
i) RR (Time slice - 2 units) ii) FCFS

Process	Arrival time	Burst time
P1	0	4
P2	1	5
P3	2	2
P4	3	1
P5	4	6
P6	6	3

Ans. : i) RR (Time slice = 2 units)



ii) SJF (Non-preemptive)



Process	Waiting time RR	Waiting time SJF
P1	20	0
P2	13	9
P3	24	14
P4	25	25
Average	= 79/4 = 19.75	= 48/4 = 12

Process	Turnaround time RR	Turnaround time SJF
P1	30	10
P2	19	15
P3	36	26
P4	40	40
	= 125/4 = 31.25	= 91/4 = 22.75

2.6 : IPC

Q.19 What is Interprocess Communication ? Explain types of IPC.

Ans. : • 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.

Q.20 What is purpose of IPC ?

Ans. : • 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.
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.

Q.21 What is pipe ? How to create pipe ?

Ans. : • A pipe is a unidirectional, first-in first-out, unstructured data stream of fixed maximum size. Writers add data to the end of the pipe; readers retrieve data from the front of the pipe.

- Once read, the data is removed from the pipe and is unavailable to other readers. A pipe provides a simple flow control mechanism.
- A process attempting to read from empty pipe blocks until more data is written to the pipe. A process trying to write to a full pipe lock until another process reads data from pipe.
- The pipe system call creates a pipe and returns two file descriptors: one for reading and one for writing. These descriptors are inherited by child processes, which thus share access to the file.
- Each pipe can have several readers and writers. A given process may be a reader or writer or both. Fig. Q.21.1 shows data flow through a pipe.

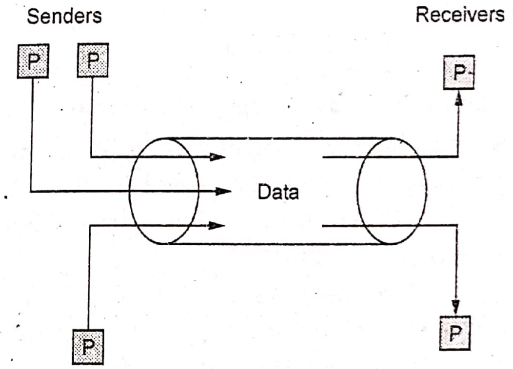


Fig. Q.21.1 Data flow through a pipe

- Pipes can be used only between processes that have a common ancestor. Normally, a pipe is created by a process, that process calls fork and the pipe is used between the parent and the child.

- Example to show how to create and use a pipe :

```
main( )
{
   int pipefd[2], n;
   char buff[100];

   if (pipe(pipefd) < 0 )
   err_sys("pipe error");
   printf("read fd = %d, write fd = %d\n", pipefd[0], pipefd[1]);

  if (write(pipefd[1], "hello world\n", 12) != 12)
   err_sys("write error");
 if ( (n=read(pipefd[0], buff, sizeof(buff))) < =0)
   err_sys("read error");
  write (1, buff, n); /*fd=1=stdout*/
  }
```

2.7 : Threads

Q.22 What is threads ? Explain advantages of threads.

Ans. : • 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. Q.22.1 shows a 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.

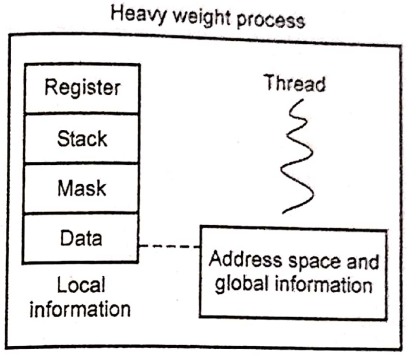


Fig. Q.22.1 Thread

Thread advantages

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.

Q.23 Differentiate between process and thread.

Ans. :

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 eve n 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.

Q.24 Explain thread life cycle.

Ans. : • Fig. Q.24.1 shows a thread lifecycle. A thread has one of the following states.

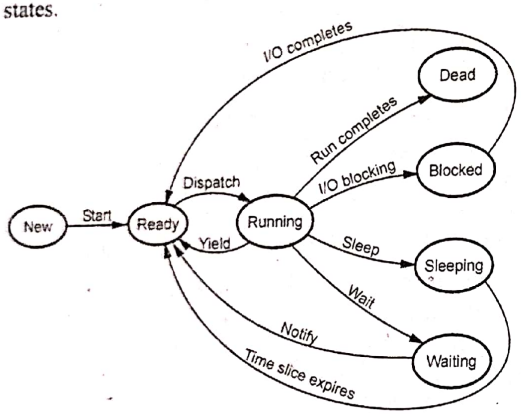


Fig. Q.24.1 Thread lifecycle

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.

Q.25 What resources are used when thread is created ? How do they differ from those used when a process is created ?

Ans. : • Thread is smaller than a process, so thread creation typically uses fewer resources than process creation.

- Creating a process requires allocating a process control block (PCB), a rather large data structure.
- The PCB includes a memory map, list of open files, and environment variables.
- Allocating and managing the memory map is typically the most time-consuming activity.
- Creating either a user or kernel thread involves allocating a small data structure to hold a register set, stack, and priority.
- New process creation is very heavyweight because it always requires new address space to be created and even if they share the memory then the inter process communication is expensive when compared to the communication between the threads.

Q.26 Define thread ? List and explain different thread scheduling approaches.

Ans. : • Different thread scheduling approaches are load sharing, dynamic scheduling, dedicated processor assignment.

- Load sharing - Processes are not assigned to a particular processor
- Gang scheduling - A set of related threads scheduled to run on a set of processors at the same time, on a one-to one basis
- Dedicated processor assignment - Provides implicit scheduling defined by the assignment of threads to processors
- Dynamic scheduling - The number of threads in a process can be altered during the course of execution.
- Also refer Q.22.

Q.27 Explain user level thread with their advantages and disadvantages.

Ans. : • 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. Q.27.1 shows user level thread.

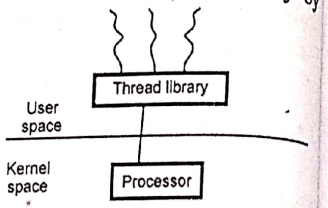


Fig. Q.27.1 User level thread

- 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.
- 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.

Q.28 Explain kernel level thread with their advantages and disadvantages.

Ans. : • 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. Q.28.1 shows Kernel level thread.

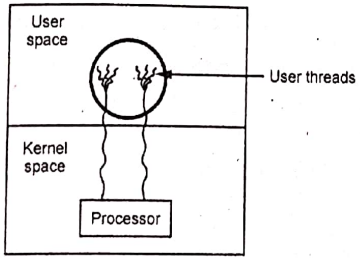


Fig. Q.28.1 Kernel level thread

- 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.

Q.29 Differentiate between user-level and kernel-level threads.

Ans. :

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 tak e 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.

2.8 : Multithreading Model

Q.30 Provide two programming examples in which multithreading provides better performance than a single threaded solution.

Ans. : Following are the programming examples :

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.

Q.31 What is multithreading ?

Ans. : • Multithreading is the ability of a program or an operating system process to manage its use by more than one user at a time and to even manage multiple requests by the same user without having to have multiple copies of the programming running in the computer.

- 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

2.9 : Threading Issue

Q.32 What is thread cancellation ? What are situation of thread cancellation ?

Ans. : • Cancellation allows one thread to terminate another. One reason to cancel a thread is to save system resources such as CPU time. When your program determines that the thread's activity is no longer necessary then thread is terminated.

- Thread cancellation is a task of terminating a thread before it has completed.
- The thread cancellation mechanism allows a thread to terminate the execution of any other thread in the process in a controlled manner. Each thread maintains its own cancelability state. Cancellation may only occur at cancellation points or when the thread is asynchronously cancelable.
- The target thread can keep cancellation requests pending and can perform application-specific cleanup when it acts upon the cancellation notice.
- A thread's initial cancelability state is enabled. Cancelability state determines whether a thread can receive a cancellation request. If the cancelability state is disabled, the thread does not receive any cancellation requests.
- Target thread cancellation occurs in two different situations:
  1. Asynchronous cancellation 2. Deferred cancellation
- **Asynchronous cancellation :** Target thread is immediately terminated. With the help of another thread, target thread is cancelled. When a thread holds no locks and has no resources allocated, asynchronous cancellation is a valid option.
- **Deferred cancellation :** When a thread has enabled cancellation and is using deferred cancellation, time can elapse between the time it's asked to cancel itself and the time it's actually terminated.

Q.33 What is signal ? How signals are handled ?

Ans. : • Signal is used to notify a process that a particular event has occurred. Signal may be synchronous or asynchronous. All types of signals are based on the following pattern :

1. At a specific event, signal is generated.
2. Generated signal is sent to a process/thread.
3. Signal handler is used for handling the delivered signal.

- **Synchronous signals :** An illegal memory access, division by zero is the example of synchronous signals. These signals are delivered to the same process which performed the operation for generating the signal.
- **Asynchronous signals :** Terminating a process with certain keystrokes are signals that are generated by an event external to the running process.
- Synchronous signal is sent to same process whether as asynchronous signal is sent to another process.
- Signals may be handled in different ways :
  1. Some signals may be ignored. For example changing the windows size.
  2. Other signals may be handled by terminating the program. For example illegal access of memory.
- Delivery of signals in multithreaded programs is more complex than single thread.
- Following are the condition where/ how should the signals be delivered to threads/process :
  a. Thread applies the signal are received the signal.
  b. Every thread in the process received the signal.
  c. Deliver the signal to limited threads in the process
  d. All the signals are received to particular thread for the process.
- The method for delivering a signal depends on the type of signals generated.
  1. Synchronous signals is sent to the thread which causes the signal.
  2. All the thread received asynchronous signals.

Q.34 What is thread pool ? What are the problem with multithread server ? List advantages of thread pool.

Ans. : • A thread pool offers a solution to both the problem of thread life-cycle overhead and the problem of resource thrashing. By reusing threads for multiple tasks, the thread-creation overhead is spread over many tasks.

- Thread pools group CPU resources, and contain threads used to execute tasks associated with that thread pool. Threads host engines that execute user tasks, run specific jobs such as signal handling and process requests from a work queue.
- Multithreaded server has potential problems and these problems are solved by using thread pool. Problems with multithreaded server :
  1. Time for creating thread
  2. Time for discarding thread
  3. Excess use of system resources.

Advantages of thread pools :

1. Servicing a request with an existing thread is usually faster than waiting to create a thread.
2. Thread pool size is fixed.

END...

Unit III

3 Process Coordination

3.1 : Synchronization : Principles of Concurrency

Q.1 Explain the principle of concurrency.

Ans. : • Concurrent access to shared data may result in data inconsistency. Maintaining data consistency requires mechanisms to ensure the orderly execution of cooperating processes.

- Concurrency arises in the same way at different levels of execution streams. Following are the example of concurrency in different types of operating systems :
  1. **Concurrency in multiprogramming :** An interaction between multiple processes running on one CPU.
  2. **Concurrency in multithreading :** An interaction between multiple threads running in one process
  3. **Concurrency in multiprocessors :** An interaction between multiple CPUs running multiple processes or threads.
  4. **Concurrency in multi-computers :** An interaction between multiple computers running distributed processes or threads.
- Java is a concurrent programming language.
- Process synchronization is required in uni-processor system, multiprocessor system and network. If more than one thread exists in a system at the same time, then the threads are said to be concurrent. Synchronization problems can occur whenever two or more concurrent processes use any shared resource.
- Cooperating process share the logical address space.

Q.2 Explain the following terms :
i) Critical section ii) Race condition

Ans. : i) Critical section :

- A critical section is a block of code that only one process at a time can execute; so, when one process is in its critical section, no other process may be in its critical section. The critical section problem is to ensure that **only one process at a time is allowed** to be operating in its critical section.

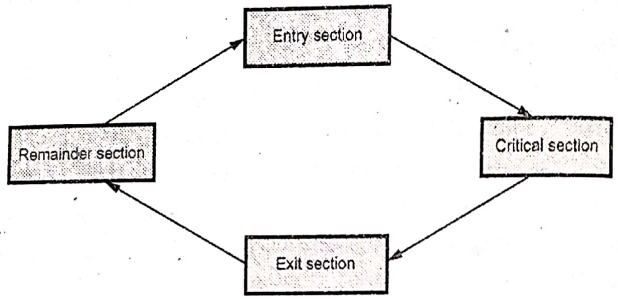


Fig. Q.2.1 Critical section

- Critical section means, process may change some common variable, writing files, updating memory location, updating a process table etc. When process is accessing shared modifiable data, it is said to be in a critical section.
- Each process takes permission from operating system to enter into the critical section. Structure of process is as follows :

  1. Entry section 2. Remainder section 3. Exit section

- **Entry section :** It is block of code executed in preparation for entering critical section.
- **Exit section :** The code executed upon leaving the critical section.
- **Remainder section :** Rest of the code is remainder section.
- Each process cycles through remainder, entry, critical, exit sections in this order.

ii) Race condition :

- **Race condition** occurs when two or more operations occur in an undefined manner. When two or more processes are reading or writing some shared data and the final result depends on who runs precisely when, are called **race condition.**
- There is a "race condition" if the outcome depends on the order of the execution. The outcome depends on the CPU scheduling or "interleaving" of the threads.
- **Race condition** should be avoided because they can cause fine errors in applications and are difficult to debug.
- In banking system **balance** is a shared variable. After depositing the amount, it update by bank employee **balance = balance + amount.** At the same time, some amount is transfer then it becomes **balance = balance – amount.** These two tasks are in a race to write variable **balance.** In this the process that updates last determines the final value of balance.
- The concurrent execution of two processes is not guaranteed to be determinate, since different executions of the same program on the same data may not produce the same result.

Q.3 List the requirements of mutual exclusion.

Ans. : Requirements of mutual exclusion

1. At any time, only one process is allowed to enter in its critical section.
2. Solution is implemented purely in software on a machine.
3. A process remains inside its critical section for a bounded time only.
4. No assumption can be made about relative speeds of asynchronous concurrent processes.
5. A process cannot prevent any other process for entering into critical section.
6. A process must not be indefinitely postponed from entering its critical section

Q.4 Explain the following terms :
i) Mutual exclusion ii) Synchronization iii) Race condition

Ans. : i) Mutual exclusion :

- Mutual exclusion methods are used in concurrent programming to avoid the simultaneous use of a common resource, such as a global variable, by pieces of computer code called critical sections
- Requirement of mutual exclusion is that, when process P1 is accessing a shared resource R1 then on other process should be able to access resource R1 until process P1 has finished its operation with resource R1.
- Examples of such resources include files, I/O devices such as printers and shared data structures.

ii) Synchronization : Synchronization means sharing system resources by processes in such way that, concurrent access to shared data is handled thereby minimizing the chance of inconsistent data. Maintaining data consistency demands mechanisms to ensure synchronized execution of cooperating processes. Process synchronization was introduced to handle problems that arose while multiple process executions.

iii) Race condition :
Refer Q.2.

3.2 : Mutual Exclusion : Hardware Support

Q.5 Explain interrupt disabling concept of mutual exclusion.

Ans. : • On uniprocessor system, parallel execution of process is not possible. But they can only be interleaved. To guarantee mutual exclusion, it is sufficient to prevent a process from being interrupted.

- A process can prevent the operating system from executing by not issuing any supervisor call instructions and by disabling interrupts. Access to a critical section can be controlled by calling those primitives.

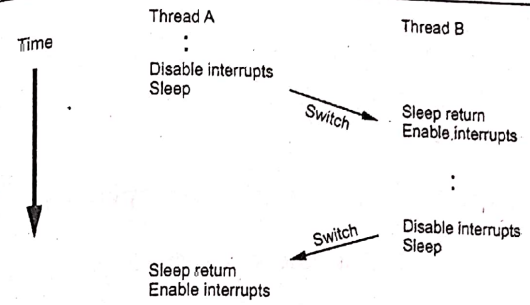


Fig. Q.5.1

- Use interrupts
  a) Implement preemptive CPU scheduling
  b) Internal events to relinquish the CPU
  c) External events to reschedule the CPU
- Disable interrupts
  a) Introduce uninterruptible code regions
  b) Think sequentially most of the time
  c) Delay handling of external events
- Mutual exclusion is guaranteed because the critical section cannot be interrupted

Disadvantages

1. It works only in a single processor environment.
2. Interrupts can be lost if not serviced promptly.
3. A process waiting to enter its critical section could suffer from starvation.

Q.6 What are the advantages and disadvantages of Test and Set operations?

Ans. : Advantages :

1. Simple to implement.
2. It works well for a small number of processes.
3. It can be used to support multiple critical sections.

Disadvantages :

1. It suffers from starvation.
2. There is possibility of busy waiting.
3. There may be deadlock.

3.3 : Operating System Support : Semaphores and Mutex

Q.7 Explain with definition, the concept of general and binary semaphore.

Ans. : Concept of general semaphore :

- Semaphore is described by Dijkstra. Semaphore is a nonnegative integer variable that is used as a flag. Semaphore is an operating system abstract data type. It takes only integer value. Its used to solve critical section problem.
- Dijkstra introduced two operations (P and V) to operate on semaphore to solve process synchronization problem. A process calls the P operation when it wants to enter its critical section and calls V operation when it wants to exit its critical section. The P operation is called as wait operation and V operation is called as signal operation.
- A wait operation on a semaphore decease its value by one.

```
waits : while S < 0
        do loops;
S:= S – 1;
```

- A signal operation increments its value :

```
signal:
S:= S + 1;
```

- A proper semaphore implementation requires that P and V be indivisible operations. A semaphore operation is atomic. This may be possible by taking hardware support. The operation P and V are executed by the operating system in response to calls issued by any one process naming a semaphore as parameter.
- There is no guarantee that no two processes can execute wait and signal operations on the same semaphore at the same time.

Binary semaphore :

- Binary semaphore is also known as mutex locks. It deals with the critical section for multiple processes. Binary semaphore value is only between 0 and 1.

1. The semaphore value is restricted to 0 and 1.
2. P succeeds only when the semaphore value is 1.
3. V does not change the semaphore value when it is 1.

- Binary semaphores are easier to implement than general semaphores

```
var
   mutex=1:binary-semaphore;
   delay=0:binary-semaphore;
   C={initvalue}:integer;
Procedure Wait()
      begin
   wait(mutex);
   C:=C-1;
   If C < 0 then begin
         signal(mutex);
         wait(delay);
         end
      else
               signal(mutex);
      end
Procedure Signal()
      begin
      wait(mutex);
      C:=C+1;
      If C <= 0 then
               signal(delay)
signal(mutex)
end
```

Q.8 List the properties of semaphore.

Ans. : Properties of semaphore :

1. Semaphores are machine independent.
2. Semaphores are simple to implement.
3. Correctness is easy to determine.
4. Semaphore acquire many resources simultaneously.
5. Can have many different critical sections with different semaphores.

Q.9 Explain the following functions (along with parameters passed) with reference to semaphore programming in 'C'.
i) sem_post( ) ii) sem_wait( )

Ans. : i) sem_post ( )

Syntax :

```
#include <semaphore.h>
int sem_post(sem_t *sem);
```

The sem_post( ) function unlocks the specified semaphore by performing a semaphore unlock operation on that semaphore. When this operation results in a positive semaphore value, no threads were blocked waiting for the semaphore to be unlocked; the semaphore value is simply incremented.

- When this operation results in a semaphore value of zero, one of the threads waiting for the semaphore is allowed to return successfully from its invocation of the sem_wait( ) function. The sem_post( ) interface is reentrant with respect to signals and may be invoked from a signal-catching function.

ii) Sem_wait ( )

Syntax :

```
int sem_wait(sem_t *s) {
                    wait until value of semaphore s is
                   greater than 0
                    decrement the value of semaphore s by 1
                  }
```


- The sem_wait( ) function locks the semaphore referenced by sem by performing a semaphore lock operation on that semaphore.
- If the semaphore value is currently zero, then the calling thread will not return from the call to sem_wait( ) until it either locks the semaphore or the call is interrupted by a signal

Q.10 Explain busy waiting with appropriate example.

Ans. : • Busy waiting is a situation in which a process is blocked on a resource but does not yield the processor. A busy wait keeps the CPU busy in executing a process even as the process does nothing.

- Busy waiting is also called as spin waiting.
- Busy waiting waste CPU cycles that some other process might be able to use productively.
- To avoid busy waiting, a process waiting for critical section, they put into the blocked state. When process allowed entering into critical section, then process is kept in ready state.
- To overcome the need for bust waiting :

In wait( ) operation :

a. When process reads semaphore value and it is not positive value then process blocked itself for some time.
b. Semaphore keeps all blocked process in the waiting queue so process is kept in this queue.
c. CPU scheduler selects another process for execution.

In signal ( ) operation :

a. The wakup ( ) operation is used for restating the blocked processes. When any other process executes signal ( ) operation, then blocked process state changed from blocked state to ready state.
b. This process is kept in the ready queue.

3.4 : Programming Language Support : Monitors

Q.11 Explain monitor in brief.

Ans. : • Monitor is an object that contains both data and procedures needed to perform allocation of a shared resource. It is a programming language construct that support both data access synchronization and control synchronization.

• Fig. Q.11.1 shows structure of monitor.

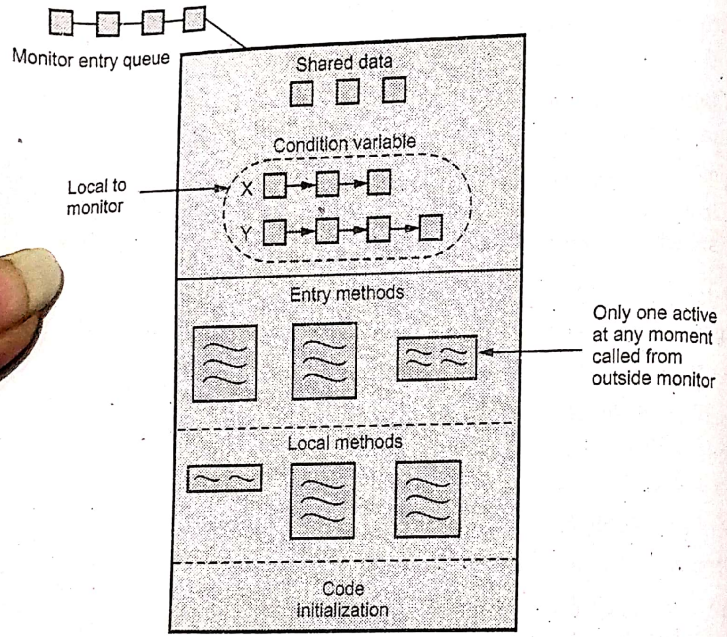


Fig. Q.11.1 Monitor structure

• Monitor is implemented in programming languages like Pascal, java and C++. Java makes extensive use of monitors to implement mutual exclusion.

• Monitor is an abstract data type for which only one process may be executing a procedure at any given time. Monitor is a collection of

procedure, variables and data structure. Data inside the monitor may be either global to all routines within the monitor or local to a specific routine.

• Monitor data is accessible only within the monitor. A shared data structure can be protected by placing it in a monitor. If the data in a monitor represents some resource, then the monitor provides a mutual exclusion facility for accessing the resource. A procedure defined within a monitor can access only those variable declared locally within the monitor and its formal parameters.

• When a process calls a monitor procedure, the first few instruction of the procedure will check to see if any other process is currently active within the monitor. If process is active then calling process will be suspended until the other process has left the monitor. If no other process is using the monitor, the calling process may enter.

• Monitor supports synchronization by the use of condition variables that are contained within the monitor and accessible only within the monitor. Every conditional variable has an associated queue. Condition variables operates on two functions :

cwait(condition variable)

csignal (condition variable)

• **cwait :** It suspend execution of the calling process on condition.

• **csignal :** Resume execution of some process blocked after a cwait on the same condition.

• The cwait must come before the csignal.

Q.12 List the drawback of monitor.

Ans. :

1. Major weakness of monitors is the absence of concurrency if a monitor encapsulates the resource, since only one process can be active within a monitor at a time.
2. There is the possibility of deadlocks in the case of nested monitors calls.

3. Monitor concept is its lack of implementation most commonly used programming languages.
4. Monitors cannot easily be added if they are not natively supported by the language.

3.5 : Classical Synchronization Problems

Q.13 Explain reader writer problem.

Ans. : • When two types of processes need to access a shared resource such as a file or database. They called these processes reader and writers.

- Multiple readers can concurrently read from the data base. But when updating the database, there can only be one writer (i.e., no other writers and no readers either)
- More than one reader may read shared data (no writers). When a writer uses shared data, all other writers and readers must be excluded.
- Fig. Q.13.1 shows reader-writer problem.

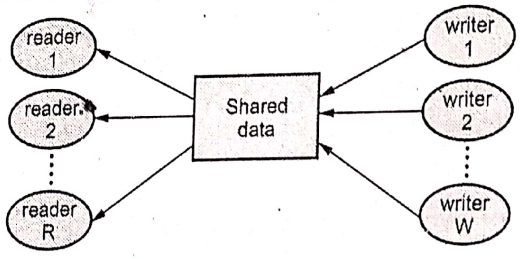


Fig. Q.13.1 Reader-writer problem

- For example in railway reservation system, readers are those who want train schedule information. They are called reader because readers do not change the content of the database. Many readers may access the database at once. There is no need to enforce mutual exclusion among readers.
- The writers are those who making reservations on a particular train. Writer can modify the database, so it must have exclusive access.

- When writer is active, no other readers or writers may be active. Therefore, it must enforce mutual exclusion if there are groups of readers and a writer.
- Reader writer problem is similar to one which a file is to be shared among a set of processes. If a process want only to read the file, then it may share the file with any other process that also wants to read the file.
- We apply rules for access order :
  1. If a writer is waiting for readers to be finished, do not allow any more readers.
  2. If a reader is waiting for a writer to finish, give reader priority.
- If the writer wants to append the file, then no other process should have access to the file when the writer has access to it.
- If reader having higher priority than the writer, then there will be starvation with writers. For writer having higher priority than reader then starvation with readers.

Q.14 Write a pseudo code for producer-consumer problem using semaphore.

Ans. : • Solution to producer consumer problem by using binary semaphore :

```
void main ( )
{
   count = 0;
}
 int i;
binarysemaphore s = 1;
int producer ( )
{
  While (true)
{
       produce_data_item ( );
     semwaitB(s);
```

```
    append ( );
    count = count + 1;
    if ( count == 1)
    semSignalB(delay);
    semSignalB(s);
    }
}
 int consumer ( )
{
    int p;
semWaitB(delay);
while (true)
{
   semWaitB(s);
consume( );
count = count - 1;
p = count;
semSignalB(s);
consumedata( );
if ( p == 0)
semWaitB(delay);
}
}
```

Q.15 Write a semaphore solution for readers - writer's problem.

Ans. : • When two types of processes need to access a shared resource such as a file or database. They called these processes reader and writers.

- The writers are those who making reservations on a particular train. Writer can modify the database, so it must have exclusive access. When writer is active, no other readers or writers may be active. Therefore, it must enforce mutual exclusion if there are groups of readers and a writer.
- Reader writer problem is similar to one which a file is to be shared among a set of processes. If a process want only to read the file, then it

may share the file with any other process that also wants to read the file. If the writer wants to append the file, then no other process should have access to the file when the writer has access to it.

```
int readcount = 0;
semaphore wsem = 1;
semaphore x = 1;
void main( )
        {
            int p = fork( );
            if(p) reader;          // assume multiple instances
            else writer;           // assume multiple instances
        }
```

```
void reader( )
    {
        while(1)
            {
                wait(x);
                readcount ++;
                if (readcount = = 1)
                wait(wsem);
                signal(x);
                doReading( );
        wait(x);
            readcount--;
        if (readcount==0)
            signal(wsem);
        signal(x);
            }
        }
```

```
void writer( ) {
   while(1) {
      wait(wsem)
      doWriting();
      signal(wsem)
   }
}
```

Q.16 What is critical section ? Give semaphore solution for producer - consumer problem.

Ans. : • A critical section is a block of code that only one process at a time can execute; so, when one process is in its critical section, no other

process may be in its critical section. The critical section problem is to ensure that only one process at a time is allowed to be operating in its critical section.

- Producer - consumer problem is example classic problems of synchronization. Producer process produce data item that consumer process consumes later.
- Buffer is used between producer and consumer. Buffer size may be fixed or variable. The producer portion of the application generates data and stores it in a buffer and the consumer reads data from the buffer.
- In order to synchronize these processes, both procedure and consumer are blocked on some condition. The producer is blocked when the buffer is full and the consumer is blocked when the buffer is empty.

1. Code for producer :

```
producer (void)
{
   int item;
   while (TRUE)
   {
      produce item (& item);
      produce_if (counter = = N)
         sleep ( );
         enter_item (item);
         counter = counter + 1;
         if (counter = = 1)
         wakeup (consumer)
      }
}
```

2. Code for consumer process

```
consumer (void)
{
   int item;
   while (TRUE)
   {
      if (count = = 0)
      sleep ();
      remove_item (& item);
      counter = counter – 1;
      if (count = = N – 1)
      wakeup (producer);
      consume_item (item);
   }
}
```


3.6 : Inter-Process Communication

Q.17 Enlist and explain different IPC mechanisms.

Ans. : Interprocess Communication provides a mechanism to exchange data and information across multiple processes, which might be on single or multiple computers connected by a network.

- 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.

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.

2. Memory Mapped :

- A memory-mapped file contains the contents of a file in virtual memory. This mapping between a file and memory space enables an application, including multiple processes, to modify the file by reading and writing directly to the memory.

3. Pipes :

- A pipe is a unidirectional, first-in first-out, unstructured data stream of fixed maximum size. Writers add data to the end of the pipe; readers retrieve data from the front of the pipe.
- Once read, the data is removed from the pipe and is unavailable to other readers. A pipe provides a simple flow control mechanism.

4. FIFO :

- Used to communicate between two processes that are not related. Full-duplex method - Process P1 is able to communicate with Process P2 and vice versa.

5. Socket :

- A socket is defined as an endpoint for communication. A pair of processes communicating over a network employs a pair of sockets, one for each process. A socket is made up of an IP address concatenated with a port number. In general, sockets use a client-server architecture.

Q.18 What is purpose of IPC?

Ans. : • Purpose 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.
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 re 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.

Q.19 What is pipe ? How to create pipe ?

Ans. :

- A pipe is a unidirectional, first-in first-out, unstructured data stream of fixed maximum size. Writers add data to the end of the pipe; readers retrieve data from the front of the pipe.

- Once read, the data is removed from the pipe and is unavailable to other readers. A pipe provides a simple flow control mechanism.
- A process attempting to read from empty pipe blocks until more data is written to the pipe. A process trying to write to a full pipe lock until another process reads data from pipe.
- The pipe system call creates a pipe and returns two file descriptors: one for reading and one for writing. These descriptors are inherited by child processes, which thus share access to the file.
- Each pipe can have several readers and writers. A given process may be a reader or writer or both. Fig. Q.19.1 shows data flow through a pipe.

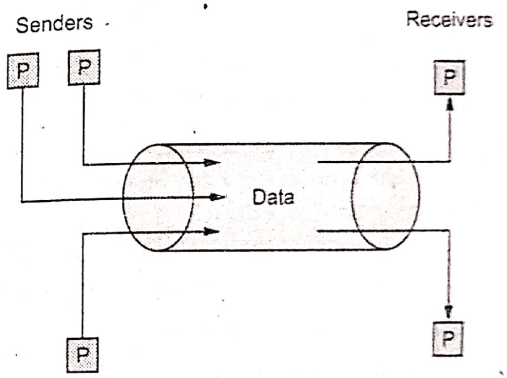


Fig. Q.19.1 Data flow through a pipe

- Pipes can be used only between processes that have a common ancestor. Normally, a pipe is created by a process, that process calls fork and the pipe is used between the parent and the child.
- Example to show how to create and use a pipe :

```
main( )
{
   int pipefd[2], n;
   char buff[100];

   if (pipe(pipefd) < 0 )
```

```
    err_sys("pipe error");
    printf("read fd = %d, write fd = %d\n", pipefd[0], pipefd[1]);

    if (write(pipefd[1], "hello world\n", 12) != 12)
     err_sys("write error");
    if ( (n=read(pipefd[0], buff, sizeof(buff))) < =0)
     err_sys("read error");
    write (1, buff, n); /*fd=1=stdout*/
    }
```

Q.20 Explain difference between named pipe and unnamed pipe.

Ans. : Difference between named pipe and unnamed pipe :

- A major difference between these IPC mechanism is the related processes are allowed to communicate using unnamed pipes but the unrelated processes are also communicate in named pipes.
- An unnamed pipe is only used for communication between a child and it's parent process, while a named pipe can be used for communication between two unnamed process as well.
- An unnamed pipe is a circular memory buffer that can be used to communicate between related processes. Named pipe is like FIFO.

Q.21 Explain the features of message passing system.

Ans. : 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.

3.7 : Deadlock

Q.22 What are deadlocks ? Explain the different methods for dealing with the deadlocks.

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

- 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.
- In resource deadlocks, processes can simultaneously wait for several resources and cannot proceed until they have acquired all those resources.

- In communication deadlocks, processes wait to communicate with other processes among a set of processes. A waiting process can unblock on receiving a communication from one of these processes.

Methods for Handling Deadlocks

1. Ensure that the system will never enter a deadlock state.
2. Allow the system to enter a deadlock state and then recover.
3. Ignore the problem and pretend that deadlocks never occur in the system; used by most operating systems, including UNIX.

Q.23 Explain the necessary and sufficient conditions for the occurrence of a deadlock.

Ans. : • 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.

Q.24 Differentiate between the following : deadlock and starvation.

Ans. : • **Deadlocks :** A group of processes are waiting for resources held by others in the group. None of them will ever make progress.

- **Starvation :** A process may wait indefinitely because other process keep coming in and getting the requested resources before this process does.
- In a deadlock situation, none of the involved processes can possibly make progress. In a starvation situation, a process is ready to execute, but it is not being allowed to execute.

Q.25 Explain with an appropriate example, how resource allocation graph determines a deadlock.

Ans. : 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. Q.25.1.

Fig. Q.25.1 Process

- A resource node is graphically represented by a rectangle and represents one type of resources. Fig. Q.25.2 shows resources node.
- The number of bullet symbols in a resource node indicates how many units of that resource class exist in the system.

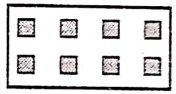


Fig. Q.25.2 Resource with 8 instances

- 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 → 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. Q.25.3 shows request and claim edge.

(a) Request edge (b) Claim edge

Fig. Q.25.3 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. Q.25.4 shows a resource allocation graph. System consists of process and resources.

Process : P_1, P_2

Resource : R_1, R_2

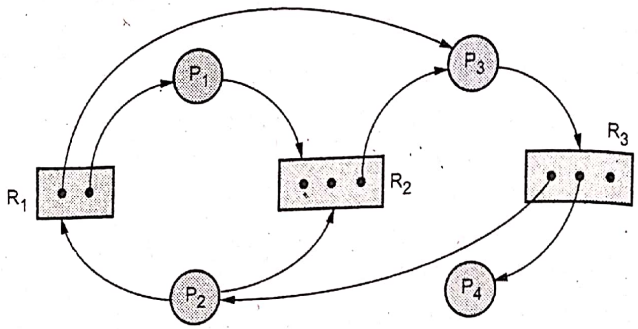


Fig. Q.25.4 Resource allocation graph

Resource	Number of instance
R_1	2
R_2	3
R_3	3

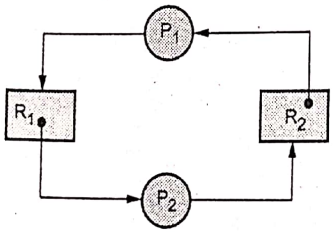


Fig. Q.25.5 Circular wait with deadlock

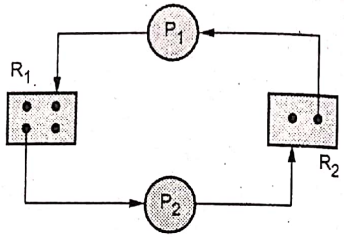


Fig. Q.25.6 Circular wait but no deadlock

Fig. Q.25.6 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

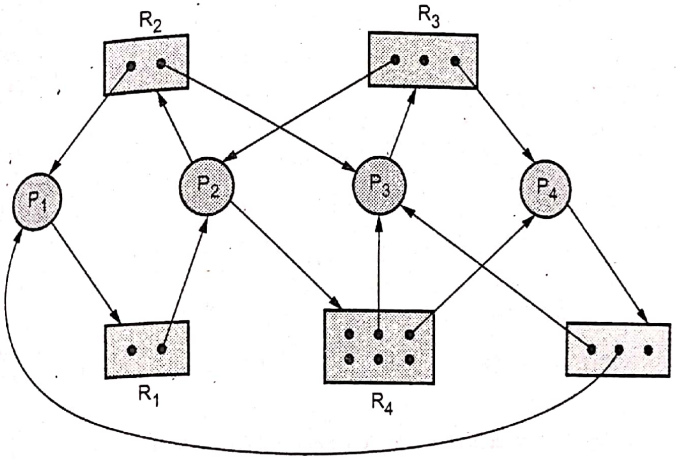


Fig. Q.25.7 Resource allocation graph

- 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.

3.8 : Deadlock Prevention

Q.26 Explain deadlock prevention techniques with example.

Ans. : • 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.

Q.27 How can the hold-and wait condition be prevented ?

Ans. : To prevent hold-and wait condition, processes must be prevented from holding one or more resources while simultaneously waiting for one or more others.

• There are several possibilities for this :

1. Require that all processes request all resources at one time. This can be wasteful of system resources if a process needs one resource early in its execution and doesn't need some other resource until much later.
2. Require that processes holding resources must release them before requesting new resources, and then re-acquire the released resources along with the new ones in a single new request. This can be a

problem if a process has partially completed an operation using a resource and then fails to get it re-allocated after releasing it.

Either of the methods described above can lead to starvation if a process requires one or more popular resources.

3.9 : Deadlock Avoidance

Q.28 Consider the following snapshot of a system. Answer the following questions using Banker's algorithm.

	Allocation	Maximum	Available
	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 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	

i) What are the contents of Need matrix ?

ii) Is the system in a safe state ?

Ans. : i) Content of need matrix is

Process	R_1	R_2	R_3	R_4
P_1	0	0	0	0
P_2	0	7	5	0
P_3	1	0	0	2
P_4	0	0	2	0
P_5	0	6	4	2

ii) Safe state

a) Need of P_1 = (0, 0, 0, 0)

Available = (1, 5, 2, 0)

Need < Available

(Required resources are allocated to P_1)

∴New available = Allocation + Available

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

= (1, 5, 3, 2)

b) For P_2 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_3 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_4

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

Available = (2, 8, 8, 6)

Need < Available

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

e) Process P_5

Need of P_5 = (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_2 process

Need of P_2 = (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_1, P_3, P_4, P_5, P_2

Q.29 Consider following snapshot of system.

Process	Allocation			Max			Available		
	R1	R2	R3	R1	R2	R3	R1	R2	R3
P1	0	1	0	7	5	3	3	3	2
P2	2	0	0	3	2	2			
P3	3	0	2	9	0	2			
P4	2	1	1	2	2	2			
P5	0	0	2	4	3	3			

Answer the following questions using bankers algorithm.

i) What is the content of matrix need ?

ii) Is the system in safe state ? What is safe sequence ?

Ans. : 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 ≥ available → 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 ≤ available (work)

(1, 2, 2) ≤ (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 >

Q.30 What is banker's algorithm? Explain data structure used by banker's algorithm.

Ans. : • 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.

Q.31 List weakness of Bankers algorithm.

Ans. : 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.

3.10 : Deadlock Detection and Recovery from Deadlock

Q.32 State and explain the deadlock recovery methods.

Ans. : • Once deadlock has been detected in the system, the deadlock must be broken by removing one or more of the four necessary conditions.

- Here one or more processes will have to be preempted, thus releasing their resources so that the other deadlocked processes can become unblocked.

1. Process termination

- Deadlock is removed by aborting a process. But aborting process is not easy. All deadlocked processes are aborted.
- Circular wait is eliminated by aborting one by one process. There will be lot of overhead. Deadlock detection algorithm must rerun after each process kill.
- Selection of process for aborting is difficult. Following parameters are considered :

1. Priority of the process.
2. What percentage the process finished its execution ?
3. Resource used by process.
4. Need of resources to complete process remaining operation.
5. How many processes will need to be terminated ?
6. Process type : Batch or interactive.

2. Resource preemption

- Some times, resource temporarily take away from its current process and allocate it to another process.
- For selecting victim, following factors are considered.

1. Priority of the process. Higher priority process are usally not seleced.
2. CPU time used by process. The process which is close to completion are usually not selected.
3. The number of other process that would be affected if this process were selected as the victim.

3. Recovery through rollback

- When a process in a system terminates, the system performs a rollback by undoing every operation related to the terminated process.
- Checkpointing a process means that its state is written to a file so that it can be restarted later.
- Risk in this method is that the original deadlock may recover but the nondeterminancy of concurrent processing may ensure that this does not happen.

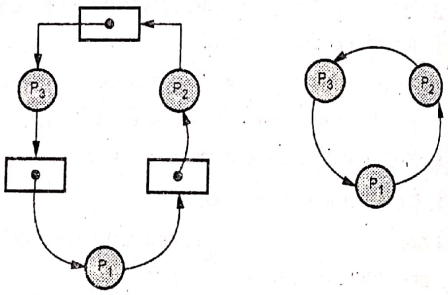
4. Starvation

- Starvation is one type situation in which a process waits for an event that might never occur in the system.
- Select the victim only for finite number of times. Use rollback method for selecting victim process.

Q.33 What is wait for graph ?

Ans. : • Any resource allocation graph with a single copy of resources can be transferred to a **wait for graph.**

- Fig. Q.33.1 shows resource allocation graph with corresponding wait for graph. The state of the system can be modeled by directed graph, called a wait for graph.
- Wait for graph is a graph, where each node represents a process. An



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

Fig. Q.33.1

edge $P_i \rightarrow P_j$ means that process P_i is blocked and waiting for process P_j to release a resource.

- The wait for graph of a system is always smaller than the resource allocation graph of that same system. There is a deadlock in a system if and only if there is a loop in the wait for graph of that system.
- Deadlock detection invloves two issues :
  1. Maintenance of the wait for graph.
  2. Searching of the wait for graph for the presence of cycles.
- Deadlock detection requires overhead for run time cost of maintaining necessary information and executing the detection algorithm.
- For multiple instances of a resource type use an algorithm similar to banker's algorithm.

END...

Unit IV

4 Memory Management

4.1 : Memory Management Requirements

Q.1 Explain primary memory requirement. Also list functions of memory management.

Ans. : Primary memory requirements

1. **Access time :** It should be as small as possible. This need influences both software and hardware design.
2. **Size :** Size must be as large as possible. It can accomodate many programs into memory.
3. **Cost :** Cost of the memory is less than the total cost of the computer.

Memory Management Function

1. Allocate primary memory space to processes.
2. Minimize access time.
3. Determining allocation policy for memory.
4. Deallocation technique and policy.

Q.2 Define the following :
Compile time, load time, execution time.

Ans. : • **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.

Q.3 What are requirement for memory management ?

Ans. : • Memory management requirements are Relocation, Protection, Sharing, Logical organization and Physical organization.

• **Relocation :** Programmer does not know where the program will be placed in memory when it is executed. While the program is executing, it may be swapped to disk and returned to main memory at a different location (relocated). Memory references must be translated in the code to actual physical memory address.

• The operating system need to know the location of :

1. Process control information
2. Execution stack
3. Entry point to begin the execution of a program. 'Processor' must deal with memory references within the program.

• **Protection :** 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.

• **Sharing :** Allow several processes to access the same portion of memory. Better to allow each process access to the same copy of the program rather than have their own separate copy.

- **Logical organization** : Main memory is organized as a linear or one-dimensional address space that consists of sequence of bytes or words. Secondary memory at its physical level is similarly organized. Most of the programs are organized into modules.
- **Physical organization** : Computer memory is organized into two levels :
  a) Main memory : Main memory is a volatile memory and it provides fast access at relatively high cost.
  b) Secondary memory : Secondary memory is a non-volatile memory and it is slower and cheaper than main memory

4.2 : Memory Partitioning : Fixed Partitioning and Dynamic Partitioning

Q.4 Explain fixed partition of memory.

Ans. : Contiguous Memory Allocation with Fixed Partitions

- 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. Q.4.1 shows an example of fixed partitioning of a memory.
- 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.

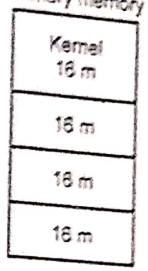


Fig. Q.4.1 Fixed size partition

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.

Q.5 Explain dynamic partition of memory.

Ans. : Contiguous Memory Allocation with Variable Partitions

- 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. Q.5.1 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.

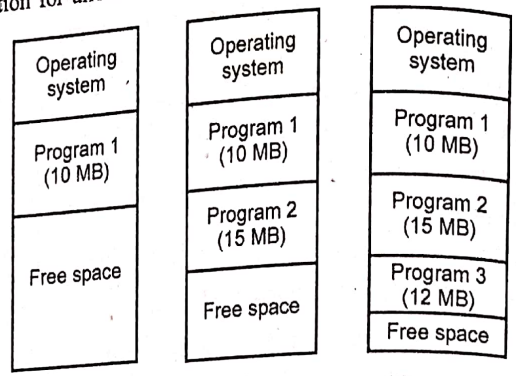


Fig. Q.5.1 (a) Variable size partition

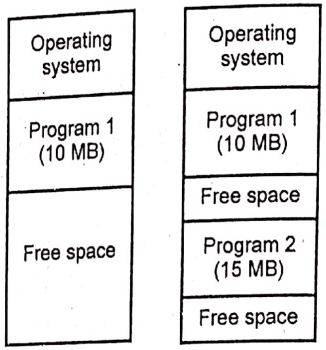


Fig. Q.5.1 (b) Noncontiguous memory allocation

- 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.

Q.6 Explain difference between contiguous and non-contiguous memory allocation.

Ans. : 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.

Q.7 What is fragmentation ? Explain types of fragmentation.

Ans. : • Fragmentation is a phenomenon in which storage space is used inefficiently, reducing capacity or performance and often both. The exact consequences of fragmentation depend on the specific system of storage allocation in use and the particular form of 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. Q.7.1 shows an internal fragmentation.

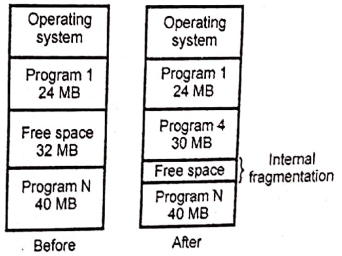


Fig. Q.7.1 Internal fragmentation

- 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. Q.7.2 shows external fragmentation.

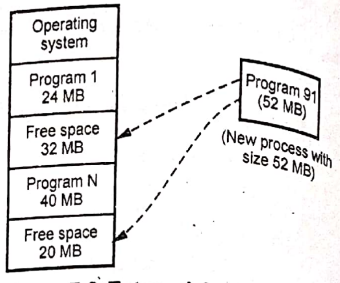


Fig. Q.7.2 External fragmentation

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

Q.8 Explain the difference between internal and external fragmentation.

Ans. :

Sr. No.	Internal fragmentation	External fragmentation
1.	Memory allocated to a process may be slightly larger than the requested memory. The difference between these two numbers is internal fragmentation.	External fragmentation exists when there is enough total memory space to satisfy a request, but available spaces are not contiguous.
2.	First-fit and best-fit memory allocation does not suffer from internal fragmentation.	First-fit and best-fit memory allocation suffers from external fragmentation.
3.	Systems with fixed sized allocation units, such as the single partitions scheme and paging suffer from internal fragmentation.	Systems with variable sized allocation units, such as the multiple partitions scheme and segmentation suffer from external fragmentation.

Q.9 Explain different ways to remove external fragmentation.

Ans. :

- Compaction solves problem of external fragmentation.
- 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.
- Fig. Q.9.1 shows compaction.

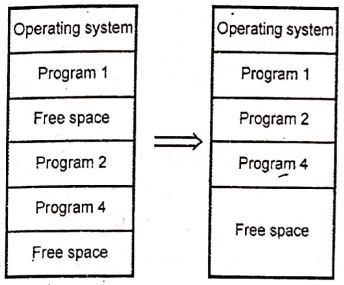


Fig. Q.9.1 Compaction

- 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.

Q.10 Free memory holes of sizes 15 K, 10 K, 5 K, 25 K, 30 K, 40 K are available. The processes of size 12 K, 2 K, 25 K, 20 K, are to be allocated. How processes are placed using first fit, best fit and worst fit partitioning algorithm. Calculate internal and external fragmentation.

Ans. : Free memory holes = 15 K, 10 K, 5 K, 25 K, 30 K, 40 K

Process size = 12 K, 2 K, 25 K, 20 K

First Fit : 12 K → 15 K, 2 K → 10 K, 25 K → 25 K, 20 K → 30 K

Best Fit : 12 K → 15 K, 2 K → 5 K, 25 K → 25 K, 20 K → 30 K

Worst Fit : 12 K → 40 K, 2 K → 30 K, 25 K → 28 K (40 K – 12 K), 20 K → 28 K (30 K – 2 K)

Q.11 Given memory partitions of 100 K, 500 K, 200 K, 300 K and 600 k (in-order), how would each of First-Fit, Best-Fit and Worst-Fit algorithms place processes of 212 K, 417 K, 112 K, 426 K ? Which algorithm makes the most efficient use of memory ?

Ans. : 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 to be the best.

4.3 : Buddy System

Q.12 Write a short note on : Buddy system.

Ans. : • The buddy system is a simple dynamic storage allocation method. The buddy system is a memory allocation and management algorithm that manages memory in power of two increments. Linux operating system manages memory using buddy algorithm.

• Fig. Q.12.1 shows buddy system.

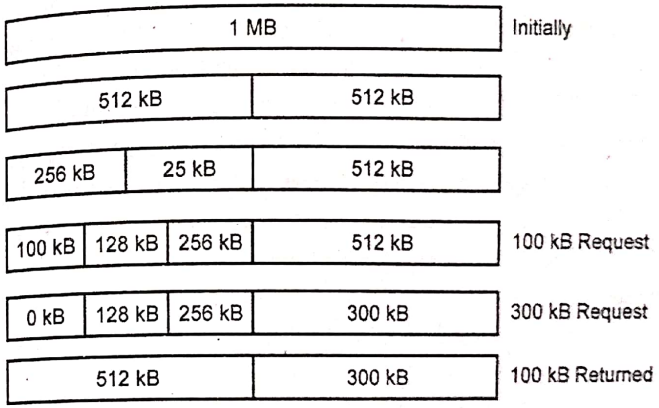


Fig. Q.12.1 : Buddy system

- The buddy system assumes that memory is of size 2^N for some integer N. Both free and reserved blocks will always be of size 2^k for $k < N$. At any given time, there might be both free and reserved blocks of various sizes.

- The buddy system keeps a separate list for free blocks of each size. There can be at most N such lists, because there can only be N distinct block sizes.
- Single allocation block to be split, to form two blocks half the size of the parent block. These two blocks are known as 'buddies'.
- Initially memory consists of a single contiguous area of 128 pages. When user send request for memory, it is first rounded up to a power of 2.
- All the lists are initially empty, except for the 1 MB list, which has one hole listed. All allocations are rounded up to a power of two. Suppose the request for memory is 75 K then allocations rounded up to 128 K, 13 K allocations rounded up to 16 K, etc.
- If a block of that size is free, it is taken; otherwise, the smallest free block larger than the desired size is found and split in two halves. This splitting continuous until the appropriate size is reached.
- When memory is deallocated, the buddy system groups contiguous free pages. When process free a block of memory, the memory manager checks the bitmap for checking block size. If adjacent block is also free, the memory manager combines the two blocks into a larger block.
- This method makes deallocation fast. If a block of size 2^K is free then the memory manager checks only the list of 2^K holes to merge them into a 2^{K+1} sized partition. Internal fragmentation is caused since memory requests are fitted in 2^K sized partitions.

Advantages

a) Easy to implement a buddy system
b) Allocates block of correct size
c) It is easy to merge adjacent holes
d) Fast to allocate memory and de-allocating memory

Disadvantages

a) It requires all allocation unit to be powers of two
b) It leads to internal fragmentation.
c) Merging of holes is recursive, so poor worst-case behavior.

4.4 : Paging

Q.13 What is difference between page and a frame ?

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.

- Fixed sized blocks are called frames and breaking of logical memory into blocks of same size called pages.
- Pages are not loaded continuously in main memory. Each page can be stored in any available page frame anywhere in main memory.
- Page frames on main memory are required in paging. No frames are required in segmentation.

Q.14 What is paging ? Explain in details.

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.

- 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. Q.14.1 shows the virtual address format for paging system.

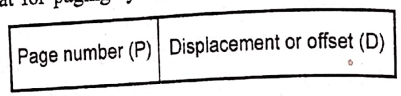


Fig. Q.14.1 Virtual address format

- 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. Q.14.2 shows address translation.

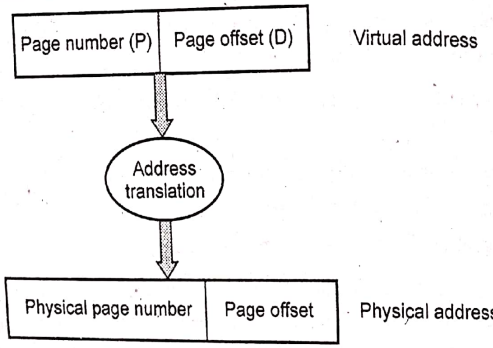


Fig. Q.14.2 Address translation

- 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.

Q.15 What are the common techniques for structuring the page table ? Explain at least three of these techniques.

Ans. : Page Table Structure :

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.

$$\text{Virtual address (v)} = (p, t, d)$$

where (p, t) = Page number

d = Displacement

- Here p is an index into the outer page table.
- Fig. Q.15.1 shows hierarchical page table.

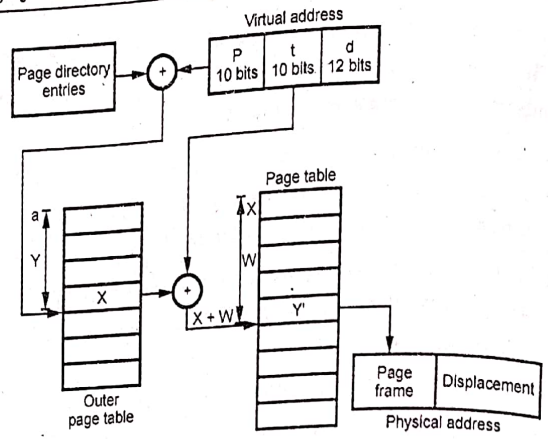


Fig. Q.15.1 Two level page table method

- Intel IA - 32 architecture support two levels of page tables.

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.

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. Q.15.2 shows block diagram of hashed page table.

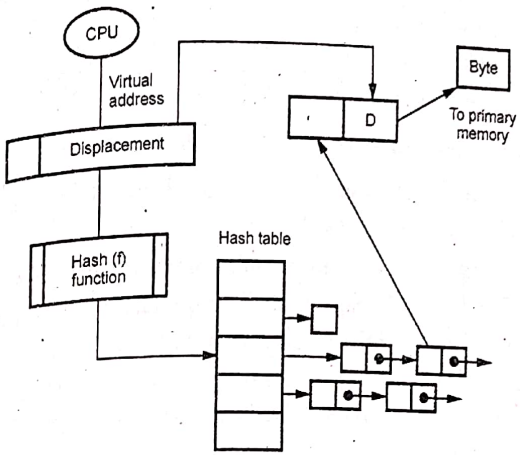


Fig. Q.15.2 Hashed page table

Working

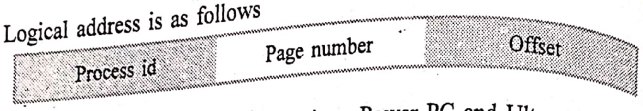
1. Virtual page number is taken from virtual address.
2. Virtual page number is hashed into the hash table.
3. Virtual page number is compared with the first element of linked list.
4. Both the value is matched, that value is (i.e. page frame) used for calculating physical address.
5. If the both value 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.

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.

- Logical address is as follows



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

Q.16 Explain with the help of a neat diagram how TLB can be used to improve effective access time ?

Ans. : • 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. Q.16.1 shows paging with TLB.

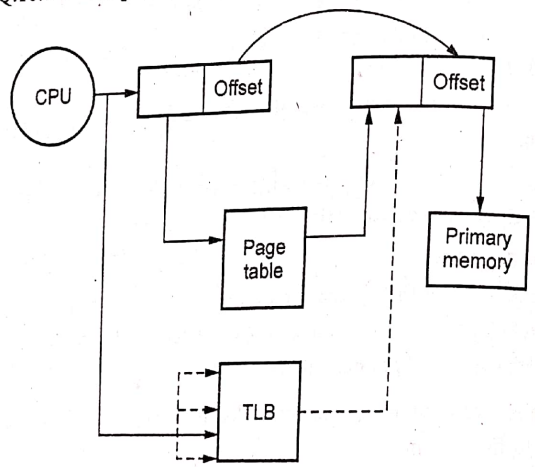


Fig. Q.16.1 Paging with TLB

- 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.

Q.17 List advantages and disadvantages of Paging.

Ans. : 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.

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.

Q.18 What elements are typically found in a page table entry ? Briefly define each element.

Ans. : The region table entry contains a pointer to a table of physical page numbers called a page table.

- Page table entries may also contain machine dependent information such as permission bits to allow reading or writing of the page. The Kernel stores page tables in memory and accesses them like all other Kernel data structures.
- Fig. Q.18.1 shows Page table entry.

Page frame number	Age	Copy on Write	Modify	Reference	Protect

1. Page frame number is the real number in the main memory.
2. Age indicates total time period of frame without being referred in the memory.
3. Copy on write : Set when more than one process shares a page.
4. Modify field indicates page has been modified.
5. Reference : Page is referred or not.
6. Valid : Page is valid if it is in main memory.
7. Protect : Whether write operation is allowed or not.

4.5 : Segmentation

Q.19 Describe the address translation mechanism in segmentation with suitable diagram.

Ans. : • In segmentation, a program's data and instructions are divided into blocks called segments. A segment is a logical entity in a program.

- 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.
- 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.
- 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. Q.19.1 shows an address translation in segmentation.

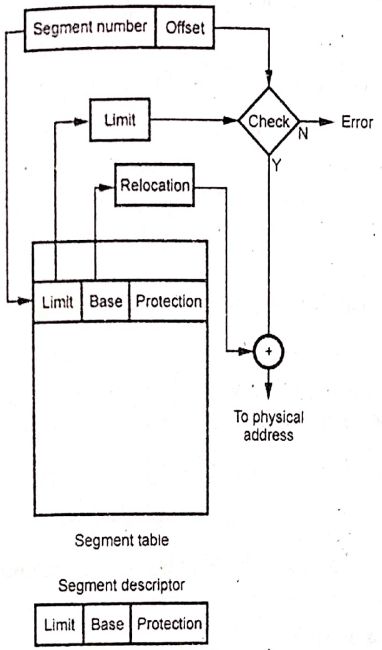


Fig. Q.19.1 Address translation in segmentation

- 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.

Q.20 Compare segmentation and paging.

Ans. :

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.

Q.21 Explain advantages and disadvantages of segmentation.

Ans. : Advantages

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

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.

Q.22 How paging is combined with segmentation ?

Ans. : Segmentation with Paging

- Most of the architecture support paging and segmentation. All the pages of segment need not be in main memory.
- It simplify the memory allocation and speed increases. It requires a high speed register to store the base address of the segment map table.
- For each segment, page table is created by OS. A pointer to the page table is kept in the segment's entry in the segment table.
- Segments are typically larger than pages.
- The base address get from the segment descriptor table is concatenated with the offset. This new address is referred to as a **linear address.** Linear address is generated by paging hardware.

Advantages

Combines all advantages of paging and segmentation.

Disadvantages

1. It increases hardware cost.
2. It increases processor overheads.
3. Dangers of thrashing.

4.6 : Virtual Memory : Hardware and Control Structures

Q.23 Explain the concept of virtual memory with suitable diagram.

Ans. : • 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.

- 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.
- 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
- Fig. Q.23.1 shows concept of virtual memory.
- 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.

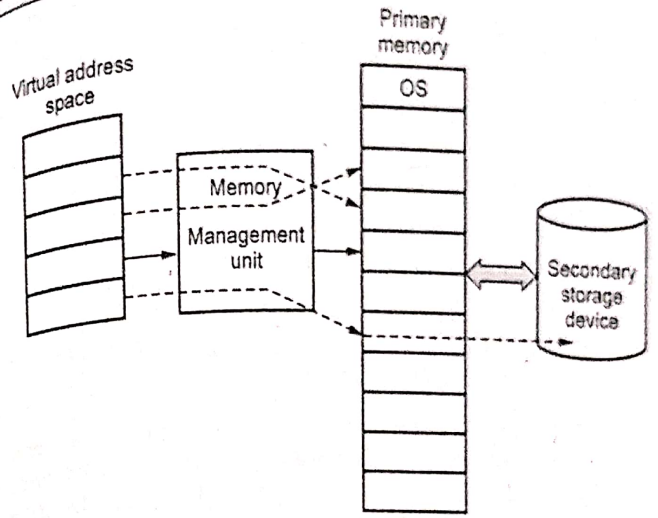


Fig. Q.23.1 Concept of virtual memory

- 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.

Q.24 Explain thrashing.

Ans. : • Thrashing occurs when a process does not have "enough" frames allocated to store the pages it uses repeatedly, the page fault rate will be very high.

- A process is thrashing if it is spending more time for paging in/out than executing. Since thrashing leads to low CPU utilization, the operating system's medium or long term scheduler may step in, detect this and increase the degree of multiprogramming.
- Local replacement algorithms can limit the effects of thrashing. If the degree of multiprogramming is increased over a limit, processor utilization falls down considerably because of thrashing. Fig. Q.24.1 shows thrashing.

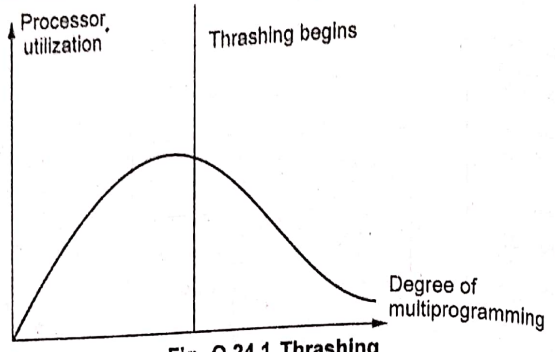


Fig. Q.24.1 Thrashing

- The selection of a replacement policy to implement virtual memory plays an important part in the elimination of the potential for thrashing. A policy based on the local mode will tend to limit the effect of thrashing. Replacement policy based on the global mode is more likely to cause thrashing. Since all pages of memory are available to all transactions, a memory-intensive transaction may occupy a large portion of memory, making other transactions susceptible to page faults and resulting in a system that thrashes.
- Thrashing is solved by using working set model and page fault frequency.

Q.25 Write short note on locality of reference.

Ans. : • Locality of reference is also known as the principle of locality. Many applications continually reference large amounts of data.

- Locality of reference is observes that an application does not access all of its data at once with equal probability. Instead, it accesses only a small portion of it at any given time.
- There are two basic types of reference locality.
  1. Temporal locality 2. Spatial locality

1. **Temporal locality :** Temporal locality is locality over time. If at one point in time a particular memory location is referenced, then it is likely that the same location will be referenced again in the near future.
2. **Spatial locality :** If a particular memory location is referenced at a particular time, then it is likely that nearby memory locations will be referenced in the near future.

4.7 : Operating System Software

Q.26 Explain fetch policy, placement policy and replacement policy.

Ans. : 1) Fetch Policy :

- When to bring a page into memory.
- **Demand paging :** Pages are fetched when needed. Normally when a page fault occurs, new pages are loaded into memory. Tends to produce a large number of page faults when the process starts, then the page fault ratio settles down.
- **Prepaging :** It bring in pages that are likely to be used in the near future. Pages other than the one needed are brought in. It makes use of disk storage characteristics. If pages are stored contiguously it may be more efficient to fetch them in one go. The policy is ineffective if the extra pages are not referenced.

2) Placement Policy

- This policy is related with determining where in real main memory a process piece is to reside.
- In **pure segmentation** systems, it is an important design issue; since most of the modern operating systems are not based on pure segmentation VM techniques, this discussion is not within our goal.
- In **pure paging or segmentation** combined with paging systems, placement is irrelevant, since the address translation hardware and main memory access hardware can perform their functions for any page-frame combinations with equal efficiency.
- It is an issue in pure segmentation systems because the memory allocated for a segment should be contiguous. The OS needs to find a memory block big enough to accommodate segments.

3) Replacement Policy

- All page frames are used. A page fault has occurred. New page must go into a frame. Which page to be replaced when a new one is brought in the main memory ? Several inter-related concepts are involved :

1. Resident set management :
   - For each active process, how many page frames are to be allocated.
   - For page replacement, pages should be considered limited per process. It caused the page fault.
2. Replacement Policy : For considered set pages, which page should be considered for page replacement.

Replacement Algorithm Objectives

- The page being replaced should be the page least likely to be referenced in the near future.
- There is a link between past history and the future because of locality.
- Thus most algorithms base their decision on past history.

Scope of Replacements

- The set of frames from which these algorithms choose is based on the scope
- **Local scope :** Only frames belonging to the faulting process can be replaced.
- **Global scope :** All frames can be replaced.
- Some frames will be locked (e.g. Kernel, system buffers etc.)

Q.27 What is page replacement ? List the name of page replacement algorithm.

Ans. : • Whenever a process requests data from a page which is not in the memory then the system uses a page replacement algorithm to swap out an existing page to make space for the new page.

- Page replacement algorithm is used to decide which page will be replaced to allocate memory to the current referenced page.
- There are three page replacement algorithms :
  1. First In First Out (FIFO)
  2. Least Recently Used First (LRU)
  3. Optimal Page Replacement
- In FIFO algorithm the system keeps track of the time when each page gets into the memory. For page replacement, the page which entered

first into the memory is selected for replacement. This technique suffers from Beladay's Anomaly.

- LRU page replacement algorithm associates with each page the time of that page's last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time.
- Optimal page replacement looks in future requests. Optimal page replacement algorithm replaces the page that will not be used for the longest period of time. Unfortunately, there is no way to determine which page will be last, so in practice this algorithm cannot be use.

Q.28 For the following reference string. 1,2,3,4,2,1,5,6,2,1,2,3,3,6. Count the number of page faults that occur with 3 frames using FIFO and LRU page replacement methods. Discuss the result.

Ans. : FIFO :

	1	2	3	4	2	1	5	6	2	1	2	3	3	6
1	1	1	1	4	4	4	4	6	6	6	6	3	3	3
2	-	2	2	2	2	1	1	1	2	2	2	2	2	6
3	-	-	3	3	3	3	5	5	5	1	1	1	1	1
Page Fault	PF	PF	PF	PF		PF	PF	PF	PF	PF		PF		PF

Total number of page fault = 11

LRU :

	1	2	3	4	2	1	5	6	2	1	2	3	3	6
1	1	1	1	4	4	4	5	5	5	1	1	1	1	6
2		2	2	2	2	2	2	6	6	6	6	3	3	3
3			3	3	3	1	1	1	2	2	2	2	2	2
Page Fault	PF	PF	PF	PF		PF	PF	PF	PF	PF		PF		PF

Total number of page fault = 11

Q.29 For the following reference string.
5,6,7,8,5,6, 9,5,6,7,8,9,6,7,4,9,8
Count the number of page fault occur with 3 frames using FIFO, optimal and LRU page replacement methods. Discuss the result.

Ans. : FIFO :

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

Total number of page fault = 13

LRU :

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

Total number of page fault = 15

Optimal :

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

Total number of page fault = 9

Q.30 For the following reference string. 6, 5, 1, 2, 5, 3, 5, 4, 2, 3, 6, 3, 2, 1, 2 Count the number of page faults that occur with 3 frames using FIFO, Optimal and LRU page replacement methods. Discuss the result.

Ans. : FIFO (Page frame = 3)

	6	5	1	2	5	3	5	4	2	3	6	3	2	1	2
1	6	6	6	2	2	2	2	4	4	4	6	6	6	1	1
2	-	5	5	5	5	3	3	3	2	2	2	2	2	2	2
3	-	-	1	1	1	1	5	5	5	3	3	3	3	3	3
Page Fault	P F	P F	P F	P F	P F	P F	P F	P F	P F	P F	P F			P F	

Total number of page fault = 12

Optimal (Page frame = 3)

	6	5	1	2	5	3	5	4	2	3	6	3	2	1	2
1	6	6	6	6	6	3	3	3	3	3	3	3	3	1	1
2	-	5	5	5	5	5	5	4	4	4	6	6	6	6	6
3	-	-	1	2	2	2	2	2	2	2	2	2	2	2	2
Page Fault	P F	P F	P F	P F		P F		P F			P F			P F	

Total number of page fault = 8

LRU (page frame = 3)

	6	5	1	2	5	3	5	4	2	3	6	3	2	1	2
1	6	6	6	2	2	2	2	4	4	4	6	6	6	1	1
2	-	5	5	5	5	5	5	5	5	3	3	3	3	3	3
3	-	-	1	1	1	3	3	3	2	2	2	2	2	2	2
Page Fault	P F	P F	P F	P F		P F		P F	P F		P F			P F	

Total number of page fault = 9

Q.31 For the following reference string 0, 1, 3, 6, 2, 4, 5, 2, 5, 0, 3, 1, 2, 5, 4, 1, 0 count the number of page faults that occur with 3 frames using FIFO, Optimal and LRU page replacement methods. Discuss the results.

Ans. : FIFO (page frame = 3)

	0	1	3	6	2	4	5	2	5	0	3	1	2	5	4	1	0
1	0	0	0	6	6	6	5	5	5	5	5	1	1	1	4	4	4
2	-	1	1	1	2	2	2	2	2	0	0	0	2	2	2	1	1
3	-	-	3	3	3	4	4	4	4	4	3	3	3	5	5	5	0
Page fault	PF	PF	PF	PF	PF	PF	PF			PF	PF	PF	PF	PF	PF	PF	PF

Total number of page fault = 15

Optimal (Page frame = 3)

	0	1	3	6	2	4	5	2	5	0	3	1	2	5	4	1	0
1	0	0	0	0	0	0	0	0	0	0	3	1	1	1	1	1	1
2	-	1	1	6	2	2	2	2	2	2	2	2	2	2	4	4	4
3	-	-	3	3	3	4	5	5	5	5	5	5	5	5	5	5	0
Page fault	PF	PF	PF	PF	PF	PF	PF				PF	PF			PF		PF

Total number of page fault = 11

LRU (Page frame = 3)

	0	1	3	6	2	4	5	2	5	0	3	1	2	5	4	1	0
1	0	0	0	6	6	6	5	5	5	5	5	1	1	1	4	4	4
2	-	1	1	1	2	2	2	2	2	2	3	3	3	5	5	5	0
3	-	-	3	3	3	4	4	4	4	0	0	0	2	2	2	1	1
Page fault	PF	PF	PF	PF	PF	PF	PF			PF	PF	PF	PF	PF	PF	PF	PF

Total number of page fault = 15

END...

Unit V

5 I/O and File Management

5.1 : I/O Devices, Organization of I/O Function

Q.1 Enlist the characteristics of block and character devices. Explain each with suitable example.

Ans. : Characteristics of block devices :

- Driver communicates by sending entire blocks of data.
- Examples for Block Devices : hard disks, USB cameras, Disk-On-Key.
- Block devices are accessed a block at a time.
- Operations supported include read( ), write( ), and seek( ).

Character Devices :

- Character devices are accessed one byte at a time.
- Supported operations include get( ) and put( ).
- Examples for Character Devices : Serial ports, parallel ports, sounds cards.

Q.2 Explain parameters used for differentiating I/O devices.

Ans. :

1. **Data rate** will change according to the input output devices. There may be differences of several orders of magnitude between the data transfer rates. Data transfer rate of keyboard is the lowest among the entire I/O device.
2. **Application :** Different devices have different use in the system.
3. **Complexity of control :** It will change according to the input-output devices. Printer requires a relatively simple control interface. Disk is much more complex interface.

4. **Unit of transfer :** Data may be transferred as a stream of bytes or characters or in larger blocks.
5. **Data representation :** Different data encoding schemes are used for different devices.
6. **Error conditions :** The nature of errors differ widely from one device to another.

Q.3 List and explain in brief I/O performing techniques (at least three).

Ans. : • I/O performing techniques are DMA, Direct I/O with polling and Interrupt driven I/O.

1. DMA :

- Direct Memory Access (DMA) is an I/O technique that allows a control unit to directly access main memory. This means that once reading or writing has begun the remainder of the data can be transferred to and from memory without processor (CPU) intervention.
- Fig. Q.3.1 shows Operation of the DMA.

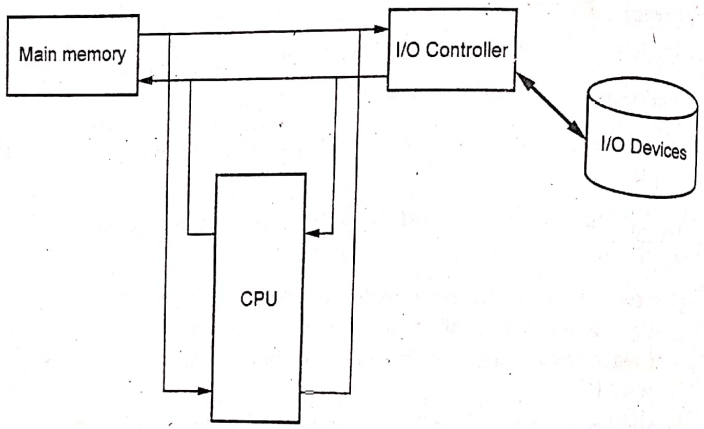


Fig. Q.3.1

- Without DMA, the processor is responsible for the physical movement of data between main memory and a device.
- A special control unit may be provided to allow transfer of a block of data directly between an external device and the main memory, without continuous intervention by the processor. This approach is called Direct Memory Access.
- DMA controller is programmed by CPU and its registers are set to perform required operations. It issues a command to a DMA controller for reading data from the secondary storage disk and stores it in the internal buffer. It verifies the checksum. If disk controller buffer contains valid data then DMA operation starts.

2. Direct I/O with polling : CPU is responsible for data transfer. It transfers data between primary memory and device controller. Following are the steps for polling :

a) Application process issues a read operation.
b) Device driver check the status of device. If device is busy, the driver waits for it to become idle.
c) Driver stores an input command into the controller command register. It starts the device.
d) Driver continuously read the status register while waiting for the device to complete its operations.
e) Driver copies the content of the controller data register into the user process space.

3. Interrupt driven I/O : When interrupt I/O is used, the CPU is free to ignore the I/O module until the interrupt signal a received. The only added overhead is processing a single interrupt when the I/O operation completes.

5.2 : I/O Buffering

Q.4 What is I/O buffering ? Explain input and output buffering.

Ans. : • Buffering is the technique by which the device manager can keep slower I/O devices busy during times when a process is not requiring I/O operations. Input/output buffering is a mechanism that

improves the throughput of input and output operations. It is implemented directly in hardware and the corresponding drivers.

- Input buffering is the technique of having the input device read information into the primary memory before the process requests it.
- Output buffering is the technique of saving information in memory and then writing it to the device while the process continues execution.

Q.5 Explain in brief different I/O buffering techniques.

Ans. : Types of I/O buffering schemes are,

1. Single buffering
2. Double buffering
3. Circular buffering
4. No buffering

1. Single Buffering

- Operating system assigns a buffer in the system portion of main memory. Fig. Q.5.1 shows single buffering.

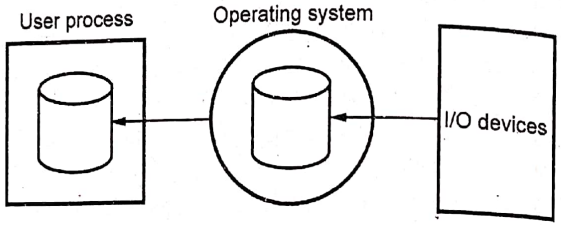


Fig. Q.5.1

2. Double Buffering

- There are two buffers in the system.
- One buffer is for the driver or controller to store data while waiting for it to be retrieved by higher level of the hierarchy.
- Other buffer is to store data from the lower level module.
- Double buffering is also called buffer swapping.
- Fig. Q.5.2 shows the hardware and software double buffering for bytes.

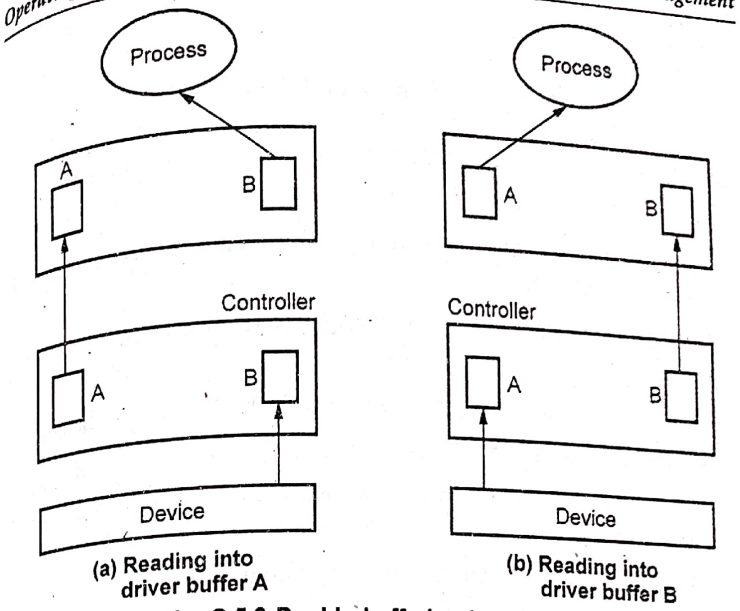


Fig. Q.5.2 Double buffering in the driver

Circular Buffer

- When more than two buffers are used, the collection of buffers is itself referred to as a circular buffer.
- In this, the producer cannot pass the consumer because it would overwrite buffers before they had been consumed.
- The producer can only fill up to buffer j−1 while data in buffer j is waiting to be consumed.
- Fig. Q.5.3 shows circular buffering.

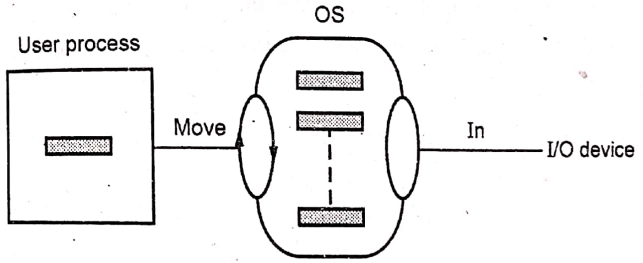


Fig. Q.5.3 Circular buffering

No Buffering : There is a buffer between user process, operating system and I/O device.

5.3 : Disk Scheduling

Q.6 Define following terms : i) Seek time ii) Rotational latency.

Ans. :

1. **Seek time :** The time it takes to position the head at the track is known as **seek time.**

   Seek time = Number of track traversed × Disk drive constant + Startup time

   Seek time do not apply to device with fixed read/write heads.

2. **Rotational latency :** The time it takes for the beginning of the sector to reach the head is known as rotational delay. It is known as search time.

Q.7 Explain SSTF and LOOK disk scheduling algorithm.

Ans. : Shortest Seek Time First

- SSTF select the next request at the one requiring the minimum seek time from the current position.
- In Shortest-Seek-Time-First (SSTF) scheduling priority is given to those processes which need the shortest seek, even if these requests are not the first ones in the queue. It means that all requests nearer to the current head positions are serviced together before moving head to distant tracks.
- Select the disk I/O request that requires the least movement of the disk arm from its current position and always choose the minimum seek time.

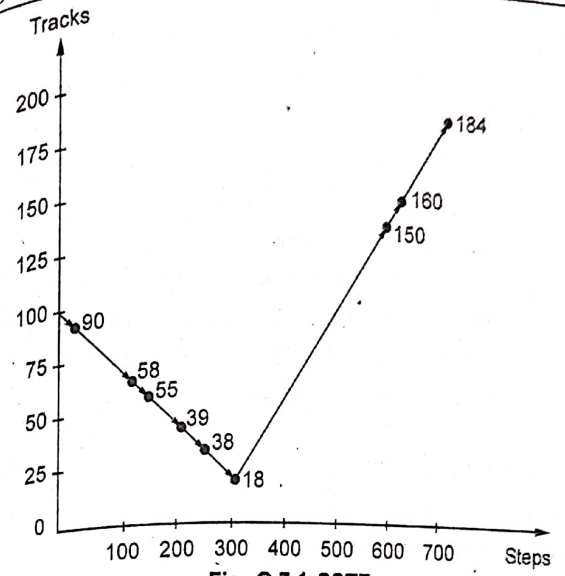


Fig. Q.7.1 SSTF

Next track accessed	Number of track traversed
90	10
58	32
55	03
39	16
38	01
18	20
150	132
160	10
184	24

$$\text{Average seek length} = \frac{10+32+3+16+1+20+132+10+24}{9} = \frac{248}{9}$$

$$= \mathbf{27.55}$$

- The only tricky part is if there are two jobs with the same distance. In this case, some kind of tie-breaking needs to be employed to pick one. For instance, you could just use a random number to effectively flip a coin and pick one.
- SSTF does not ensure fairness and can lead to indefinite postponement because its seek pattern tends to be highly localized.
- Under heavy load, SSTF can prevent distant requests from ever being serviced. This phenomenon is known as starvation. SSTF scheduling is essentially a form of shortest job first scheduling. SSTF scheduling algorithm are not very popular because of two reasons.

  1. Starvation possibly exists. 2. It increases higher overheads.

Advantages :

1. Throughput is better than FCFS.
2. SSTF minimize the response time.
3. Less number of head movements.

Disadvantages :

1. One major issue is STARVATION.
2. Localize under heavy load.

LOOK Scheduling

- Start the head moving in one direction. Satisfy the request for the closest track in that direction when there is no more request in the direction, the head is traveling, reverse direction and repeat. This algorithm is similar to SCAN, but unlike SCAN, the head does not unnecessarily travel to the innermost and outermost track on each circuit.
- Fig. Q.7.2 shows the look scheduling algorithm. For example, the disk request queue contains a set of references for blocks on tracks 76, 124, 17, 269, 201, 29, 137 and 12.

Current head position = 76

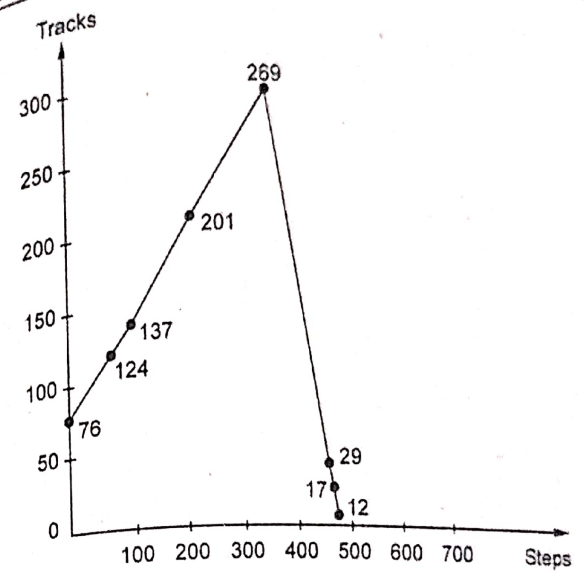


Fig. Q.7.2 LOOK scheduling

Next track accessed	Number of track traversed
124	48
137	13
201	64
269	68
29	240
17	12
12	5

Average seek length $= \frac{48+13+64+68+240+12+5}{10} = \frac{450}{7} = \mathbf{64.28}$

Q.8 Explain FIFO disk scheduling with example.

Ans. : • FIFO is also called first come first served method. Requests are processed in queue order. Fig. Q.8.1 shows FCFS algorithm.

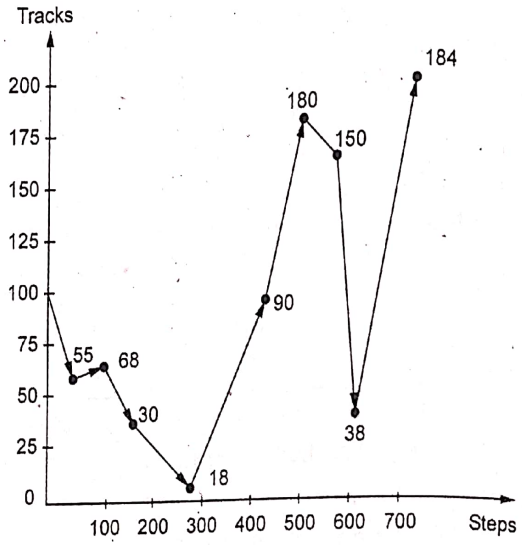


Fig. Q.8.1 FCFS

- FCFS has a fair policy in the sense that once a request has arrived, its place in the schedule is fixed irrespective of arrival of a higher priority request.
- The requested tracks in the order received are : 55, 58, 39, 18, 90, 160, 150, 38 and 184. Starting track at 100 and total number of track is 200. FCFS process the entire request in sequential order.
- FCFS would begin at track 100, move 45 tracks to 55, and move 3 tracks to 58 and so on. The Y-axis corresponds to the tracks on the disk. The X-axis corresponds to time or the number of tracks traversed.
- Starting at track 100.

Next accessed track is as below.

Next track accessed	Number of track traversed
55	45
58	03
39	19
18	21
90	72
160	70
150	10
38	112
184	146

$$\text{Average seek length} = \frac{45+3+19+21+72+70+10+112+146}{9}$$

$$= \frac{498}{9} = 55.333$$

- FCFS is simple to implement. But it does not provide fast services. It can not take special action to minimize the overall seek time.

Advantages :

1. Simple and easy to implement.
2. Suitable for light loads.

Disadvantages :

1. FCFS does not provide fast services.
2. Do not maximize throughput.
3. May involve lots of unnecessary seek distance

Q.9 Briefly describe SCAN disk scheduling algorithm.

Ans. : • SCAN is also called elevator algorithm.

- The next request scheduled is closest to current request but in one particular direction. All requests in other direction are put at the end of the list. SCAN services tracks in only one direction i.e. either increasing or decreasing track number.
- When SCAN reaches the edge of the disk (or track 0), it reverses direction.

- If any request comes on its way it will be serviced immediately, while request arriving just behind the head will have to wait until disk head moves to the end of the disk, reverses direction and returns before being serviced.

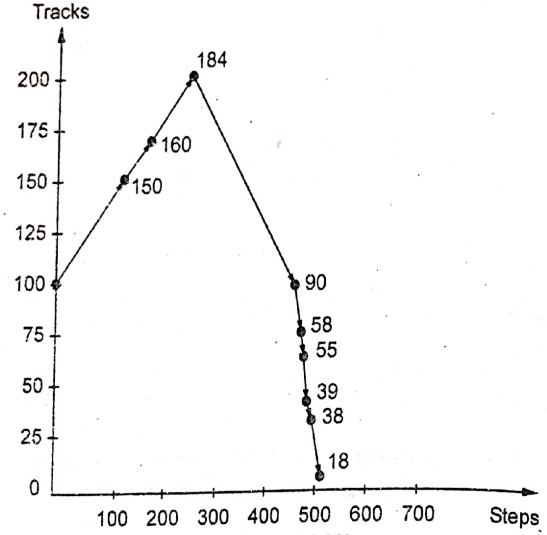


Fig. Q.9.1 SCAN

- SCAN is called elevator because an elevator continues in one direction servicing requests before reversing direction.
- Consider the example in which the requested tracks in the order received are : 55, 58, 39, 18, 90, 160, 150, 38, and 184. Starting track at 100 and total number of track is 200. Head is moving towards increasing track number.

Current head position = 100

Head is moving towards increasing number of track.

Next track accessed	Number of track traversed
150	50
160	10

184	24
90	94
58	32
55	03
39	16
38	01
18	20

$$\text{Average seek length} = \frac{50+10+24+94+32+3+16+01+20}{9} = \frac{250}{9}$$

$$= \mathbf{27.77}$$

Advantages :

1. SCAN eliminates starvation.
2. Throughput similar to SSTF.

Disadvantages :

1. Increase overhead.
2. Needs directional bit.

Q.10 A disk drive has 200 cylinders, numbered 0-199. The drive is currently serving the request at cylinder 53. The queue of pending requests in FIFO order is 98, 183, 37, 122, 14, 124, 65, 67. Starting from the current head position what is the total distance that disk arm moves to satisfy all the pending requests for the following disk scheduling algorithms. i) FCFS ii) SCAN iii) C-LOOK iv) SSTF

Ans. : i) FCFS :

Head movement = 53 → 98 → 183 → 37 → 122 → 14 → 124 → 65 → 67

Total head movement = 640

ii) SCAN :

Head movement = 53 → 37 → 14 → 0 → 65 → 67 → 98 → 122 → 124 → 183

Total head movement = 208

iii) C-LOOK :

Head movement = 53 → 65 → 67 → 98 → 122 → 124 → 183 → 14 → 37

Total head movement = 236

iv) SSTF :

Head movement = 53 → 65 → 67 → 37 → 14 → 98 → 122 → 124 → 183

Total head movement = 236

Q.11 For the given sequence of disk request, determine the total distance travelled by disk head in satisfying the entire request for FCFS, C-SCAN and SSTF algorithms. Initial head position is 120 and total number of cylinders in the disk is 200.
120, 130, 180, 150, 25, 10, 105, 90

Ans. : i) FCFS :

Head movement = 120 → 130 → 180 → 150 → 25 → 10 → 105 → 90

Total head movement = 340

ii) C-SCAN :

Head movement = 120 → 130 → 150 → 180 → 200 → 1 → 10 → 25 → 90 → 105

Total head movement = 383

iii) SSTF :

Head movement = 120 → 130 → 150 → 180 → 105 → 90 → 25 → 10

Total head movement = 230

Q.12 Is disk scheduling, other than FCFS useful in a single user environment. Explain your answer.

Ans. : • Yes, disk scheduling other than FCFS scheduling can be useful in a single-user environment.

• SSTF is useful when the data sets are near each other. In the elevator scheduling, we saw that the effects are optimal because it is more efficient than FCFS but unlike SSTF, there are no starvation.

• In a single-user environment, the I/O queue usually is empty. Requests generally arrive from a single process for one block or for a sequence of consecutive blocks.

• In these cases, FCFS is an economical method of disk scheduling. But LOOK is nearly as easy to program and will give much better performance when multiple processes are performing concurrent I/O, such as when a Web browser retrieves data in the background while the operating system is paging and another application is active in the foreground.

Q.13 A disk drive has 200 cylinders, numbered 0 - 199. The drive is currently serving the request at cylinder 63. The queue of pending requests in FIFO order is 27, 129, 110, 186, 147, 41, 10, 64, 120. Starting from the current head position what is the total distance that disk arm moves to satisfy all the pending requests for the following disk scheduling algorithms. i) FCFS ii) C-SCAN iii) C-LOOK iv) SSTF

Ans. : Currently serving the request at cylinder 63.

FCFS		C-SCAN		C-LOOK		SSTF	
Nest Request	Difference in head movement	Nest Request	Difference in head movement	Nest Request	Difference in head movement	Nest Request	Difference in head movement
27	36	64	1	64	1	64	1
129	102	110	46	110	46	41	23
110	19	120	10	120	10	27	14
186	76	129	9	129	9	10	17
147	39	147	18	147	18	110	100
41	106	186	39	186	39	120	10
10	31	0	186	10	176	129	9
64	54	10	10	27	17	147	18

120	56	27	17	41	13	186	39
		41	14				

Total head movement in FCFS = 519

Total head movement in C-SCAN = 349

Total head movement in C- LOOK = 329

Total head movement in SSTF = 231

5.4 : File Management Concept

Q.14 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.15 List the objectives for a file management system.

Ans. : Objectives are :

1. Meet the data management needs and requirements of the user
2. Guarantee that the data in the file are valid
3. Optimize performance
4. Provide I/O support for a variety of storage device types
5. Provide I/O support for multiple users.

Q.16 Discuss the following with respect to file system : file attributes.

Ans. : File Attributes

- One of the characteristics of file is a set of file attributes that give the operating system more information about the file and how it is intended to be used. For human users convenience bane is given to the file. File attributes are varies from system to system. File attributes are as follows :

1. Name	2. Identifier	3. Type	4. Location/place
5. Size	6. Protection	7. Date and time	

- **File name** is in human readable. User can perform any operation on file using its name.

- When file is created by user, operating system assigns unique identification number to each file. OS uses this **identifier** for its operation.
- **Type** information is required for systems that support different types of files.
- **Location/place** information is pointer to a device and actual location of files on the device. The information needed to locate the file on disk is kept in memory.
- **Size** is measured in bits or bytes. It gives idea about current size of the file.
- **Protection :** This information is stored on the per-process table so the operating system can allow or deny subsequent requests. Attributes like protection, password, creator and owner provides protection to a file.
- **Date and time :** This information is related to creation and modification of files. Protection, security and monitoring purposes this data is used by operating system.

Q.17 Discuss the following with respect to file system : File operation.

Ans. : • File operations are simply those things that user can perform on a file. For example, user can create a file, save a file, open a file, read a file and modify a file. There are many different types of file operations supported by operating system.

- Operating system can provides system call for performing various operations on the file. Six basic file operations are creating, write, read, delete, repositioning and truncating.
- Following are the some list of file operation :

1. Create	2. Write
3. Close	4. Read
5. Delete	6. Repositioning

- All these operations require that the directory structure be first searched for the target file.

- **File creation :** Any time user can create a file. File is also created without data. The steps for creating a file.
  a. Space : File system must provide a space for the file.
  b. New file entry is made in the directory.
- **Write :** To perform write operation on the file, file name and data is required. It updates a file by adding new data and changes some content of the file. For writing into a file, operating system searches the directory for particular file. Write pointer is kept at the location where the writing starts. Write pointer position is at middle of the file or end of the file. Write pointer is updated when writing to file is completed.
- **Close :** Process is not used a file after closing. Process can not perform any operation on file. Operating system remove it's entry from open file table.
- **Read :** A process reads all or some portion of the data in a file. Read system call is used for reading a file. For reading a file, name and position of the memory is specified. Read pointer is kept into the file for next read operation. Read pointer is updated after completion of the read system call. Information about read pointer is stored in *per process current file position pointer.*
- **Delete :** Operating system remove the file from file structure. For deleting a file, directory is searched with particular name and file is deleted. Deleting means making space free from memory and disk. Operating system also removes the directory entry. This free space is used by another file.
- **Repositioning :** The directory is searched for the proper entry, and the current-file-position pointer is repositioned to a given value. Repositioning within a file need not involve any actual I/O. This file operation is also known as a file seeks.
- Other file operations are open, append, seek, get and set attributes and rename.
- **Open :** Before performing any operation on file, it is necessary to open file in required mode.

- **Append :** Append is related to write operation of file. Append only add data to the end of the file.
- **Get and set attributes :** Get attribute is used before performing operation on file. When user request a file for any operation, get attributes is executed by the process. Depending upon the attributes, user can allow or deny the operation. User can set the attributes and it changed after creation of file.
- **Rename :** User can change the name of existing file.

Q.18 What is the objective of file management system ?

Ans. : Objective of file management system

1. Provide I/O support the multiple users.
2. Provide I/O support for a variety of secondary storage device.
3. Support for data management
4. It minimizes the data lost.
5. Optimize the system performance.

Q.19 Draw and explain file system architecture.

Ans. : File System Architecture

- Fig. Q.19.1 shows file system architecture.

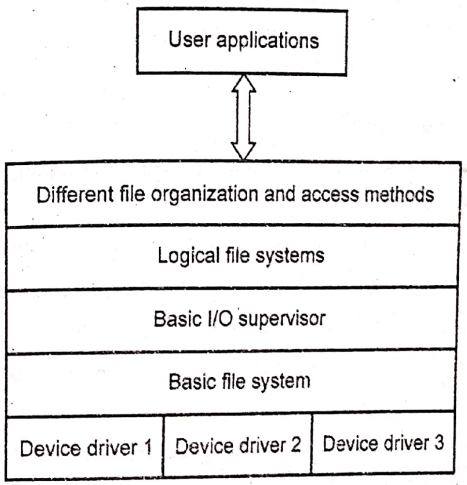


Fig. Q.19.1 File system architecture

Device driver

- Device driver directly communicate with peripheral device and their controllers.
- It is responsible for starting physical I/O operations on the device.

Basic file system

- It uses specific device driver.
- It is primary interface used by system for communicating with outside world.
- It exchange block of data with secondary storage device.
- It is part of the operating system.

Basic I/O Supervisor

- It is responsible for initialize all file I/O and termination.
- It is also part of operating systems.
- It select the device on which file I/O is to be performed.
- It allocates I/O buffer.

5.5 : Access Methods

Q.20 Explain different file organization techniques.

Ans. : • Different file organization techniques are Sequential Access, direct access, Indexed sequential access.

Sequential Access Method

- Sequential access method is simple method. The information in a file is accessed sequentially one record after another.
- In this method, a process could read all the records in a file in order, starting at the beginning. It can not skip any records and can not read out of order. Fig. Q.20.1 shows sequential access method.
- Batch application uses sequential files. A byte stream file uses sequential access method.
- **Example :** Compiler and editor usually access files in this method. Transaction file is also example of sequential access method.

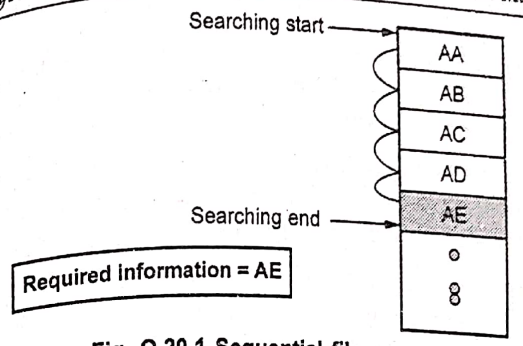


Fig. Q.20.1 Sequential file access

- Sequential file organization only method easily stored on tape and hard disk. Sequential access is best suited where most of the records in a file are to be processed.

Disadvantages :

- It provides poor performances.
- More efficient search technique is required.

Direct Access Method

- It is also called random access method. This access allows a user to position the read/write mark before reading or writing. Fig. Q.20.2 shows direct access method.

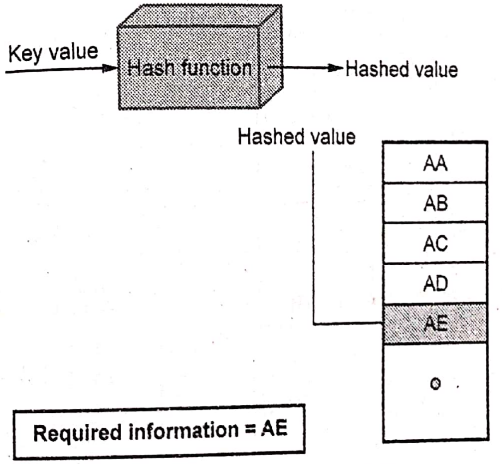


Fig. Q.20.2 Direct access method

- Direct access file method provides, accessing the records directly. Direct access method is based on the hard disk that is a direct access device. It allows random access of any file block.
- Each record has its own address on the file with by the help of which it can be directly accessed for reading or writing. This feature is used by editors. An editors need to randomly access the contents of the file.
- There is no restriction on the order of reading or writing for a direct access file. Operating system support is not required for direct access file.
- At the time of file creation, access method is defined. According to defined access method, file is accessed. Sequential access of a direct access file is possible but direct access of a sequential file is not.

Disadvantages :

1. Poor utilization of input-output device.
2. Consumes CPU time for record address calculation.

Indexed Sequential Access

- It is modified method of direct access method. This method is combination of direct and sequential access method.
- In simple indexed sequential method, simple single level of indexing is used. Sequential file is used as index.
- Fig. Q.20.3 shows indexed sequential file. Indexed sequential access contains key field and a pointer to the main file.

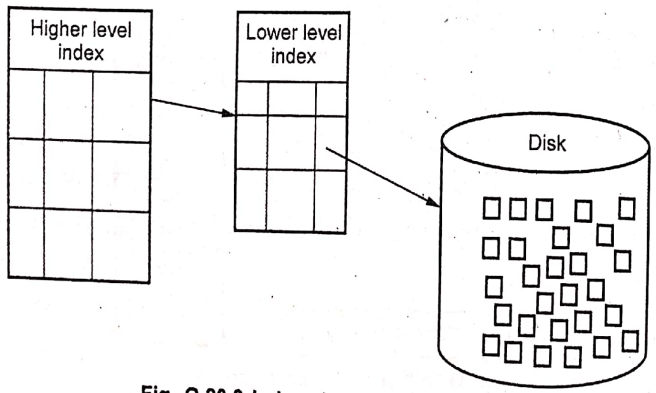


Fig. Q.20.3 Indexed sequential access

- Indexed sequential files are store records in the order that they are written to the disk. Records may be retrieved in sequential order or in random order using an index to represent the record number in the file.
- If file size is large, larger index is required and it contains large number of entries. CPU takes time to search into the index. Higher level index is used to reduce the search time. An entry in the higher level index points to a section of the index. This higher index contains the records. This index is searched to find the section of the disk that may contains a required record.
- File system maintains an overflow file. It adds new records to an overflow file. The record in the main file that immediately precedes the new record in logical sequence is updated to contain pointer to the new record in the overflow file.
- The overflow file is merged with the main file during a batch update. The multiple indexes for the same key field can be set up to increase efficiency.
- The lowest level of index file is treated as a sequential file and a higher level index file is created for that file. Higher index file would contain pointers to lower index files, which would point to the actual data items.

5.6 : Directory Structure

Q.21 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.

Q.22 Explain various operation performed on directory.

Ans. : Operation on directory :

1. **File searching :** Directory structure is searched for particular file entry. File uses symbolic names and similar names may indicate a relationship between files.

2. **Create a file :** User creates new files and added to the directory.
3. **Delete a file :** User remove the file after completing its work on files.
4. **Rename a file :** User change the file name if file content is change.
5. **Listing of directory :** Listing of files in the directory for some use. MS-DOS and windows uses "dir" command and Linux/UNIX uses "ls" command for this purposes.

Q.23 Write a short note on : Directory structure.

Ans. : • Directories are basically symbol tables of files. A single flat directory can contain a list of all files in a system. A directory contains information about the files, including attributes, location and ownership.

- All information about files is kept in the directory structure. It is stored on the secondary storage device.
- **Single level directory :** Single level directory is a simple structure and there are no sub-directories. It is also called flat directory structure.
- In single level directory structure, no two files can have the same name.
- **Two level directory :** Two-level directory structure contains master file directory at the top level then user file directory at the second level. Actual user files are at the third level.
- File system maintains a master block for each user. Master block has one entry for each user. User directory address is stored in the master block. Each user has a private directory.
- **Tree structured directory :** In this structure, directory itself is a file. A directory and sub directory contains a set of files. Internal format is same for all directories. Commonly used directory structure is tree structure. Tree has a root directory. All files in disk have a unique path name.

Q.24 Explain acyclic graph tree.

Ans. : • Acyclic graph directory solve the problem of tree structure directory. It allows sharing the directory in between two users. At a time more than one places, shared directory or file will exist in the file system. Fig. Q.24.1 shows acyclic graph directory. Links can be used to construct acyclic graph structure. Acyclic graph is graph with no cycles.

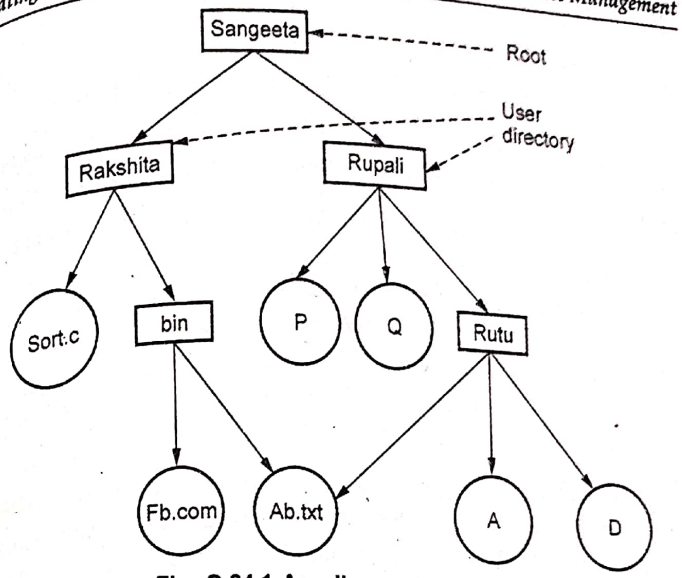


Fig. Q.24.1 Acyclic graph directory

- It is very interesting to note that a shared directory or file is not the same as two copies of the file. When there are two copies of files, each user can view the copy rather than the original, but if one user changes the file content, the changes will not appear in the other's copy.
- Only one original file exists for shared copy. Any changes made by one user are immediately visible to the other user. When user create file in shared directory, it automatically appear in all the shared subdirectories.

5.7 : Protection

Q.25 Which types of access is provided by protection mechanism ?

Ans. : • Access is limited or full access of data. User having full access can perform read, write and modify operation on the data and information. In limited access, user only read and executes the information.

- Protection mechanism provides controlled access. Following are some of the operation performed on the files.

1. Read : User can read a file.

2. Write : User can rewrite a file.
3. Delete : User can delete a file whenever space is required.
4. Execute : User execute a file after loading into the main memory.
5. List : User check attributes of the file and file/directory names.

Q.26 What is access control and access determination ? Explain.

Ans. : • Traditionally, a file object in Linux is associated with three sets of permissions. These sets assign read (r), write (w), and execute (x) permissions for the three user groups file owner group, and other.

- Nine bits are used to determine the characteristics of all objects in a Linux le system. Additionally, the set user id, set group id and sticky bits can be set for special cases.
- ACLs can be used for situations where the traditional file permission concept does not suffice. They allow the assignment of permissions to individual users or groups even if these do not correspond to the owner or the owning group. Access Control Lists are a feature of the Linux kernel.
- There are two types of ACLs : Access ACLs and default ACLs. An access ACL is the access control list for a specific file or directory. A default ACL can only be associated with a directory; if a file within the directory does not have an access ACL, it uses the rules of the default ACL for the directory. Default ACL's are optional.
- ACL's can be configured :
    1. Per user 2. Per group
    3. Via the effective rights mask
    4. For users not in the user group for the file.

Access determination

- When a process attempts to access a file, its effective UID is compared to the UID that owns the file. If they are the same, access is determined by the ACL's user permissions. Otherwise, if a matching user specific ACL entry exists, permissions are determined by that entry in combination with the ACL mask.

- If no specific entry is available, the file system tries to locate a valid group related entry that provides the required access; these entries are processed in conjunction with the ACL mask. If no matching entry can be found, the other entry prevails.

5.8 : File System Implementation

Q.27 Write short note on file system implementation.

Ans. : • File system is implemented on disk and memory. How to implement the file system, it varies according to operating system and file system. But all the operating system follow some general rules. If the file system is implemented on the disk, it contains the following information :

1. **Boot block control :** Boot block is maintained per volume. The boot block contains the initial bootstrap program used to load the operating system. Operating system requires some information at the time of booting. If the disk is divided into number of partitions, the operating system is stored in the first partition of the disk. If the operating system is not installed on the disk, then this block can be empty. In UNIX file system, this is called the book block and in NTFS, it is the partition boot sector.
2. **Volume control block :** It consists of volume or partitions detail information. The information like block size, number of blocks in the partition, free block count, free block pointer, free FCB count and FCB pointers. In UNIX operating system, each partition is a standalone file system. Superblock is name in UNIX file system. A super block describes the state of the file system : The total size of the partition, the block size, pointers to a list of free blocks, the inode number of the root directory, magic number, etc. In network file system, it is stored in the master file table.
3. **Directory structure :** It is used to organize the files. Directory structure is maintained per file system.
4. **Per-file PCB :** It contains information about files such as file size, file ownership, file permission and location of data blocks. In NTFS, master file table stored this information. Master table file uses a relational database structure.

- In-memory information is used for caching and files system management. Caching improve the performance.

1. **In-memory mount table :** It contains information about each mounted volume.
2. **In-memory directory structure cache :** It contains recently accessed directories information.
3. **System wide open table :** Open files FCB information is stored.
4. **Per-process open file table :** It maintains the pointer to the appropriate entry in the system wide open file tables.

- UNIX operating system treats a directory exactly as a file. Windows NT implements separate system calls for files and directories. It treats directory as entities separate from files. Fig. Q.27.1 shows a file control block.
- FCB specify the information that the system needs to manage a file. Sometimes it is called file attributes. These are highly system dependent structures. For creating new file, an application program calls the logical file system.

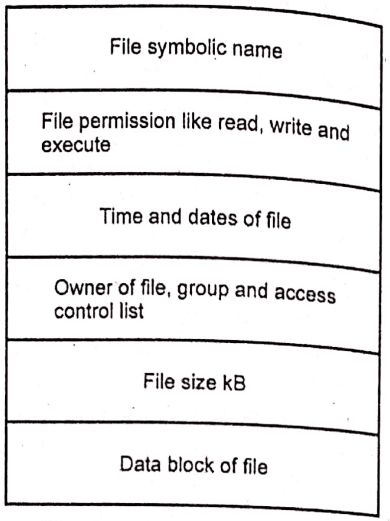


Fig. Q.27.1 File control block

Optimizations for file system :

- Delayed updates of data and metadata are main difficulty. Updates could be delayed in the hope that the same data might be updated in the future or that the updated data might be temporary and might be deleted in the near future.
- If computer were to crash without having committed the delayed updates, then the consistency of the file system is destroyed.

Metadata updates :

- For recoverable file system after crash, it must be consistent or must be able to be made consistent. So it is necessary to prove that logging metadata updates keeps the file system in a consistent.
- For inconsistent file system, the metadata must be written incompletely. Sometime, file system data structures is in wrong order. With metadata logging, the writes are made to a sequential log.
- The complete transaction is written there before it is moved to the file system structures. While updating data, file system crashed, the updates can be completed based on the information in the log.
- The logging ensures that file system changes are made completely. The order of the changes is guaranteed to be correct because of the sequential writes to the log. If a change was made incompletely to the log, it is discarded, with no changes made to the file system structures.

5.9 : Directory Implementation

Q.28 How directory is implemented by using linear list and hash table ?

Ans. : • Directory is implemented by using linear list and hash table.

Linear list

- Linear list is simple method of directory implementation. It uses file names with pointers to data block. It is time consuming because of searching overheads.
- To create a new file, file name is searched in the directory to avoid the file name duplication. Then it adds new entry at the end of the directory. To delete a file, again file name is searched in the directory. Space is released after deleting a file. Directory entry is also removed. Unused space is marked as free entry. Last entry of directory is copied into the free space list.
- For deleting a file, some OS uses linked list. Linked list decreases time required.

- Searching the directory entry is the major disadvantage. It is slow process. Because of this reason, many operating system uses software cache memory. It stores the most recently used directory information.

Hash table

- Hash table is one more data structure used for directory implementation. Hash table takes a value computed from the file name and returns a pointer to the file name in the liner list.
- Hash table reduces the searching time. It uses fixed size block.

5.10 : Allocation Methods

Q.29 Explain contiguous file allocation.

Ans. : • When user creates a file, system allocates a set of contiguous blocks on disk. This is a pre-allocation method that uses portion of variable size. Contiguous allocation is simple method to implement. It only search free list of correct number of consecutive blocks and mark them as used block.

- Disk address is a linear ordering on the disk. Because of linear ordering, accessing block b + 1 after block b normally requires no head movement. Here head movement is only one track. Fig. Q.30.1 shows contiguous allocation.
- Contiguous allocation of a file is defined by the disk address and the length of the first block.
- If the file size is "n" blocks and starting location is "L", then it occupies blocks L, L+1, L+2, L+3, ..., L+(n–1). The directory entry for each file indicates the address of the starting block and the length of the area allocated for this file.
- Sequential and random access can be supported by contiguous allocation.
- It is easy to retrieve single block. To improve the I/O performance of sequential processing, multiple blocks can be brought in one at a time. It supports sequential access very well because files entire data is stored in adjacent block. This method also supports random access.

File allocation table

File name	Start block number	Block length
File 1	1	3
File 2	18	2
File 4	6	6
File 5	20	1
File 6	25	3

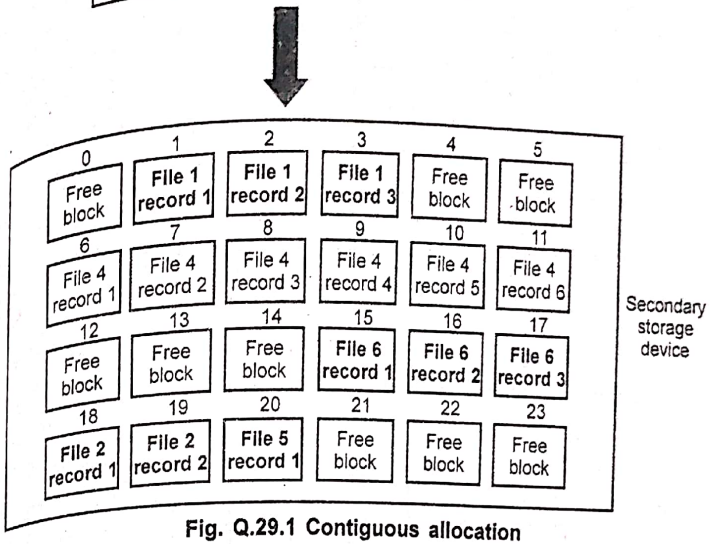


Fig. Q.29.1 Contiguous allocation

- Contiguous allocation also suffers from **external fragmentation.** Small free disk spaces are created after allocation free block and deleting files. External fragmentation means there will require free space available but that is not contiguous. To solve the problem of external fragmentation, compaction method is used.
- One more problem is that how to calculate the space needed for a file. It is difficult to estimate the required space.

Q.30 Describe indexed allocation with its advantages and disadvantages.

Ans. : • Indexed allocation method solves the problem of both contiguous and linked allocation. It uses concept of index block. Index block stores the entire pointer in one location. But the index block will occupy some space and thus could be considered as an overhead of the method.

- OS keeps a list of free blocks. It allocates an array to hold pointers to all the blocks used by the file. It allocates blocks only on demand. Fig. Q.30.1 shows indexed allocation.

File name	Index block number
File 1	10
File 4	24

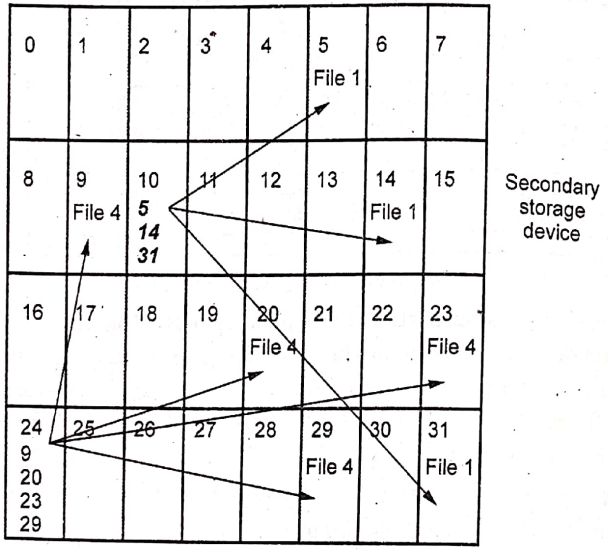


Fig. Q.30.1 Indexed allocation

- In indexed allocation, each file maintains its own index block. It contains an array of disk sector of addresses. For example : The n^{th} entry in the index block points to the n^{th} sector of the file. The directory contains the address of the index block of a file. To read the

n^{th} sector of the file, the pointer in the n^{th} index block entry is read to find the desired sector.

- It supports direct access and without suffering from external fragmentation. Any free block anywhere on the disk may satisfy a request for more space.
- Indexed allocation does suffer from wasted space. The pointer overhead of the index block is generally greater than the pointer overhead of linked allocation.

Advantages :

1. It supports sequential and direct access.
2. No external fragmentation.
3. Faster than other two methods.
4. It supports fixed and variable size blocks.

Disadvantages :

1. Indexed allocation does suffer wasted space.
2. Pointer overhead is generally greater.

Q.31 Compare contiguous, linked and indexed file allocation methods.

Ans. :

Sr. No.	Contiguous	Linked	Indexed
1.	Allocate each file to contiguous blocks on disk.	Allocate linked-list of fixed-sized blocks.	Allocate fixed-sized blocks for each file.
2.	It suffers from external fragmentation.	No external fragmentation.	No external fragmentation.
3.	Pre-allocation is required.	Pre-allocation is possible.	Pre-allocation is possible.

4.	It support variable size portions.	It support fixed size portions.	It supports fixed and variable size portions.
5.	Simple to calculate random addresses.	Cannot calculate random addresses without reading previous blocks.	Supports random access.
6.	Allocation frequency is once.	Allocation frequency is low to high.	Allocation frequency is medium.
7.	No pointer overhead.	Space required for pointer.	Pointer overhead is generally greater.

Q.32 Explain advantages and disadvantages of contiguous and linked allocation methods.

Ans. : Advantages of contiguous allocation methods

1. It supports variable size portion.
2. Easy to retrieve single block.
3. Accessing a file is easy.
4. It provides good performance.

Disadvantages of contiguous allocation methods

1. It suffers from external fragmentation.
2. Pre-allocation is required.

Advantages of linked allocation methods

1. There is no external fragmentation.
2. It is never necessary to compact disk space.
3. Pre-allocation is not required.

Disadvantages of linked allocation methods

1. Files are accessed only sequentially.
2. Space required for pointers.
3. Reliability is not good.
4. Can not support direct access.

Q.33 Describe linked allocation with its advantages and disadvantages.

Ans. : • Linked allocation is also called chained allocation. Operating system keeps an ordered list of free blocks. File descriptor stores pointers to the first block and each block stores pointer to the nest block.

• Fig. Q.33.1 shows linked allocation. The disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of the file. No space is lost to disk fragmentation.

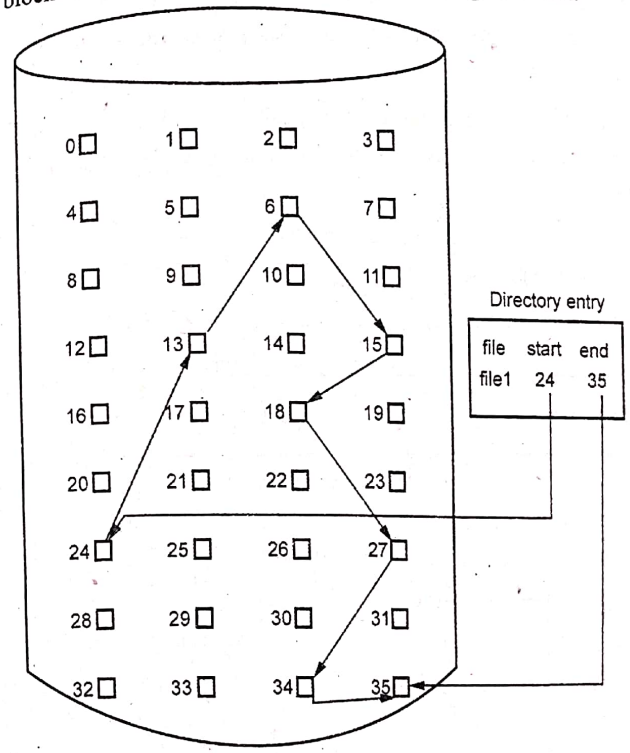
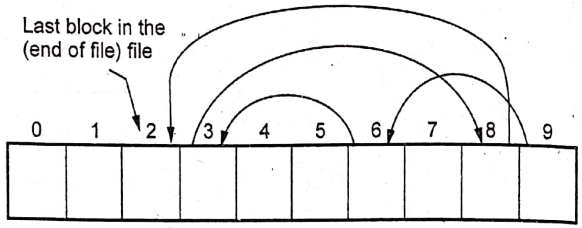


Fig. Q.33.1 Linked allocation

• Creation of new file is easy. For new file, simply create new entry in the directory. Reading a file is straightforward. User simple read blocks by following pointers from block to block. There is no external fragmentation with linked allocation.

- To write to file, system finds a free block, and this new block is written to and linked to the end of the file.
- While creating a new file, it is not necessary to declare the size of the file. A file can contiguous to grow as long as free blocks are available.
- Compaction can be used so that blocks of one file are located continuously on the disk. It optimizes disk access.
- File allocation table is an extension of the linked allocation method. Instead of putting the pointers in the file, keep a table of the pointers around. This pointer table can be quickly searched to find ant random block in the file.
- Fig. Q.33.2 shows the file allocation table. All blocks on the disk must be included in the table. This method is used by windows operating system.



Current disk allocation

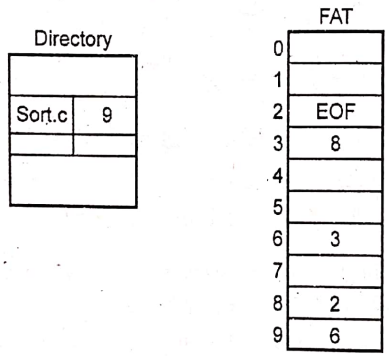


Fig. Q.33.2 FAT

Characteristics

1. It supports fixed size portions.
2. Pre-allocation is possible.
3. File allocation table is one entry for a file.
4. Allocation frequency is low to high.

Advantages

1. There is no external fragmentation.
2. It is never necessary to compact disk space.
3. Pre-allocation is not required.

Disadvantages

1. Files are accessed only sequentially.
2. Space required for pointers.
3. Reliability is not good.
4. Can not support direct access.

5.11 : Free Space Management

Q.34 What is meant by free space management ?

Ans. : To keep track of free disk space, the system maintains a free space list. Free space list. Free space list records all free disks blocks i.e. those not allocated to some file or directory.

Q.35 Explain bit vector with its advantages and disadvantages.

Ans. :

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.

Advantages

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.

Q.36 Explain link list and grouping of free space management.

Ans. : Link list :

- In the 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.

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.

1. Consider the free blocks : 3, 4, 5, 7, 9, 14, 15, 19, 30, 31, 32, 35, 36, 37, 38. Then,

   When all the blocks in the group have been allocated, then use the block that held the pointer.

2. Because of grouping method, address of a large number of free blocks can be found quickly.

END...

Unit VI

6 Linux

6.1 : History of Unix and Linux

Q.1 Discuss history of Linux.

Ans. :

- Linux is a modern, free operating system based on UNIX standards. First developed as a small but self-contained kernel in 1991 by Linus Torvalds, with the major design goal of UNIX compatibility, released as open source.
- Its history has been one of collaboration by many users from all around the world, corresponding almost exclusively over the Internet.
- It has been designed to run efficiently and reliably on common PC hardware, but also runs on a variety of other platforms.
- The core Linux operating system kernel is entirely original, but it can run much existing free UNIX software, resulting in an entire UNIX-compatible operating system free from proprietary code.
- Linux system has many, varying Linux distributions including the kernel, applications, and management tools

Q.2 List and explain components of Linux.

Ans. : Components of Linux System

- Linux operating system consists of three components :

  1. Kernel 2. System library 3. System utility

- Fig. Q.2.1 (see on next page) shows Linux operating system components.

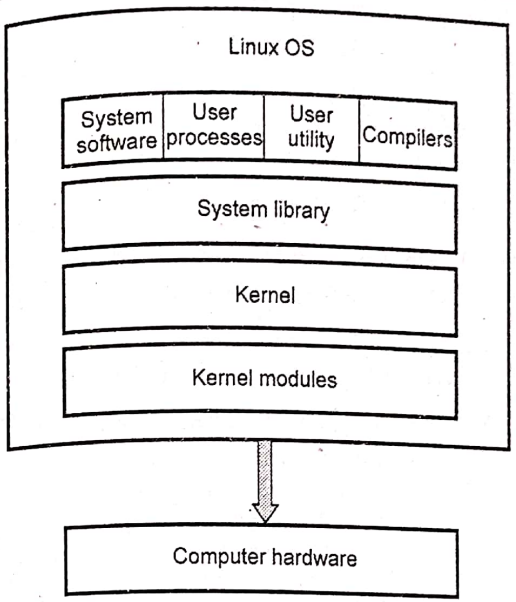


Fig. Q.2.1 Linux OS components

1. **Kernel :** Kernel is the heart of the Linux operating system. It is responsible for all major activities of Linux OS. It provides important abstraction to hide low level hardware information to user or application program. Kernel is combination of various modules and it interacts directly with the underlying hardware. The Linux kernel consists of several important parts : Process management, memory management, hardware device drivers, file system drivers, network management etc.
2. **System library :** System libraries are special functions or programs. Libraries are used for accessing kernel's features. These libraries implements most of the functionalities of the operating system and do not requires kernel module's code access rights.
3. **System utility :** System utility programs are responsible to do specialized, individual level tasks.

- Privileged mode is called kernel mode in Linux system. All the kernel code executes in the privileged mode with full access to the hardware resource of the computer. In Linux operating system, no user mode code is built into the kernel.

Q.3 Explain design principles of Linux.

Ans. : • Linux is a multiuser, multitasking system with a full set of UNIX-compatible tools

- Its file system adheres to traditional UNIX semantics, and it fully implements the standard UNIX networking model
- Main design goals are speed, efficiency, and standardization
- Linux is designed to be compliant with the relevant POSIX documents; at least two Linux distributions have achieved official POSIX certification.
- It supports Pthreads and a subset of POSIX real-time process control.
- The Linux programming interface adheres to the SVR4 UNIX semantics, rather than to BSD behavior

6.2 : Shell

Q.4 What is shell ?

Ans. : • A shell is special user program which provide an interface to user to use operating system services. Shell accept human readable commands from user and convert them into something which kernel can understand.

- It is a command language interpreter that execute commands read from input devices such as keyboards or from files. The shell gets started when the user logs in or start the terminal.

Q.5 Write short note on shell.

Ans. :

- A shell is a program that acts as the interface between you and the UNIX system, allowing you to enter commands for the operating system to execute.
- A UNIX shell is both a command interpreter and a programming language. As a command interpreter, the shell provides the user interface to the rich set of GNU utilities. The programming language features allow these utilities to be combined.

- Files containing commands can be created and become commands themselves. These new commands have the same status as system commands in directories such as/bin, allowing users to groups to establish custom environments to automate their common tasks.
- Shells may be used interactively or non-interactively. In interactive mode, they accept input typed from the keyboard. When executing non-interactively, shells execute commands read from a file.
- A shell allows execution of gnu commands, both synchronously and asynchronously. The shell waits for synchronous commands to complete before accepting more input; asynchronous commands continue to execute in parallel with the shell while it reads and executes additional commands.
- The redirection constructs permit fine-grained control of the input and output of those commands. Moreover, the shell allows control over the contents of commands environments.

6.3 : Linux Utility Programs and Kernel Structure

Q.6 What is a utility in Linux ?

Ans. : • A utility (program), sometimes referred to as a command, performs a task that is frequently related to the operating system. A utility is simpler than an application program, although there is no clear line separating the two. Linux distributions include many utilities.

Q.7 Explain principle kernel component of Linux.

Ans. : • Principle kernel component of Linux are signals, system call, virtual memory, processor and scheduler.

- Signals : Kernel uses signal to call into a process.
- System calls : Process request a specific kernel service.
- Virtual memory : Allocates and manages virtual memory for process.
- Processor and scheduler : Creates, manages and schedule the process.

Q.8 What is kernel module ? Explain module management system.

Ans. : • Modules are pieces of code that can be loaded and unloaded into the kernel upon demand. These loadable kernel modules run in privileged kernel mode.

- Kernel module can implement a device driver, a file system or a networking protocol.
- The module interface allows third parties to write and distribute, on their own terms, device drivers or file systems that could not be distributed under the GPL.
- Kernel modules allow a Linux system to be set up with a standard, minimal kernel, without any extra device drivers built in.
- The Linux kernel is modular, which means it can extend its capabilities through the use of dynamically-loaded kernel modules. A kernel module can provide a device driver which adds support for new hardware; or support for a file system.
- On modern systems, kernel modules are automatically loaded by various mechanisms when the conditions call for it. However, there are occasions when it is necessary to load and/or unload modules manually, such as when a module provides optional functionality, one module should be preferred over another although either could provide basic functionality or when a module is misbehaving, among other situations.
- Kernel modules allow a Linux system to be set up with a standard minimal kernel, without any extra device drivers built in.
- The module support following Linux components :

1. Module-management system
2. Module loader and unloader
3. Driver-registration system
4. Conflict-resolution mechanism

Module-management system

- It allows modules to be loaded into main memory and to communicate with the rest of the kernel.
- Internal symbol table is stored in the kernel by Linux. This symbol table contains only explicitly exported entries. The set of exported

symbols constitutes a well-defined interface by which a module can interact with the kernel.

- User can use these symbols by using an explicit request. Although exporting symbols from a kernel function requires an explicit request by the programmer.
- The loading of the module is performed in two steps :

1. The kernel must reserve a continuous area of virtual kernel memory for the module. Address of the allocated memory is return by kernel. Loader uses this address for loading machine code.
2. Module passes the system call and symbol table to kernel.

Q.9 Explain driver-registration system and conflict-resolution mechanism.

Ans. : Driver registration

- Kernel also maintains dynamic tables of all known drivers. At any time, these drivers are removed or added by using routines. These routines are responsible for registering the module's functionality.
- Allows modules to tell the rest of the kernel that a new driver has become available.
- Registration tables include following :

1. Device drivers : These drivers include character devices, block devices and network interface devices.
2. File systems : It contains format for storing files on a disk and other required information.
3. Network protocols : Module may implement all required networking protocols.
4. Binary format : This format specifies a way of recognizing loading and executing a new type of executed file.

Conflict resolution

- A mechanism that allows different device drivers to reserve hardware resources and to protect those resources from accidental use by another driver.

- The conflict resolution module aims to :
  1. Prevent modules from clashing over access to hardware resources.
  2. Prevent auto probes from interferring with existing device drivers.
  3. Resolve conflicts with multiple drivers trying to access the same hardware.

Q.10 What the kernel does ?

Ans. : The kernel has 4 jobs :

- **Memory management :** Keep track of how much memory is used to store what and where.
- **Process management :** Determine which processes can use the CPU, when and for how long.
- **Device drivers :** Act as mediator/interpreter between the hardware and processes.
- **System calls and security :** Receive requests for service from the processes.

6.4 : Processes in Linux

Q.11 What is process management ?

Ans. : • Process management separates the creation of processes and the running of a new program into two distinct operations.

- The fork() system call creates a new process.
- A new program is run after a call to exec()
- Under Linux, process properties fall into three groups: the process's identity, environment, and context.

Q.12 Explain process environment.

Ans. : • The process's environment is inherited from its parent, and is composed of two null - terminated vectors:

a) The argument vector lists the command-line arguments used to invoke the running program; conventionally starts with the name of the program itself.

b) The environment vector is a list of "NAME=VALUE" pairs that associates named environment variables with arbitrary textual values.

- Passing environment variables among processes and inheriting variables by a process's children are flexible means of passing information to components of the user-mode system software.
- The environment - variable mechanism provides a customization of the operating system that can be set on a per-process basis, rather than being configured for the system as a whole.

Q.13 Explain process state transition diagram of Linux.

Ans. : • Fig. Q.13.1 shows a process state transition diagram.

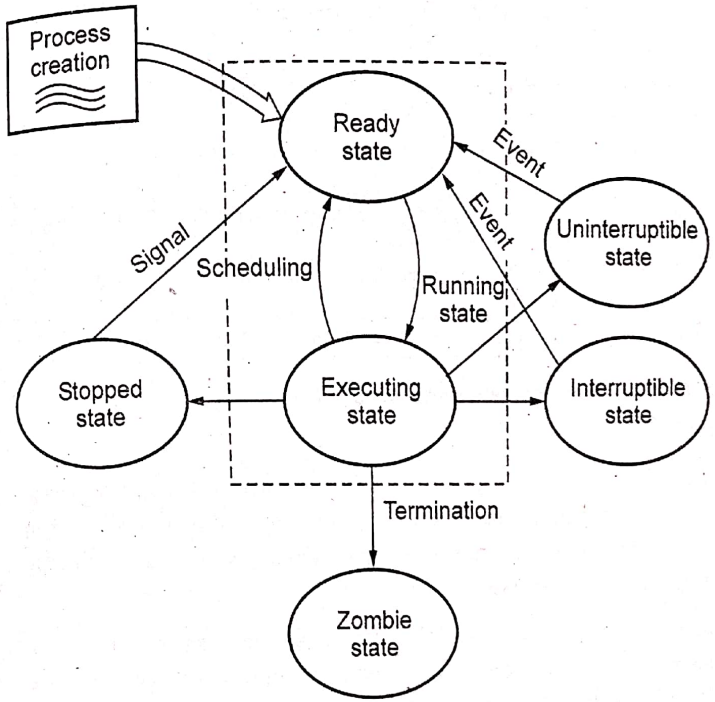


Fig. Q.13.1 Process/Thread state model

1. **Running :** Running state is two type : Executing or ready to execute.

2. **Interruptible :** It is blocked state of process. The process is waiting for an event or signal from another process.
3. **Uninterruptible :** It is also blocked state. Here process is waiting directly on hardware condition. Process can not handle any signals in this state.
4. **Stopped :** Process has been halted and can only resume by positive action from another process.
5. **Zombie :** Process has been terminated for some reason.

Q.14 Explain data structure used by task_struct.

Ans. : • Process in Linux is represented by task_struct data structure. It contain various information.

State	Various state of process execution : Ready, executing, suspended, zombie and stopped.
IPC	Linux support IPC mechanism.
Scheduling information	Information needed by Linux to schedule processes. Process can be real time and a priority.
Links	Each process contains link to its parent and to its all children process.
Identifier	Each process has a unique process identifier (PID). It also user and group ID.
Times	It includes process creation time and time consumed by process.
File system	Includes pointers to any files opened by this process.

Q.15 Discuss Linux process scheduling policy.

Ans. :

- Linux uses a timesharing technique. We know that this means that each process is assigned a small quantum or time slice that it is allowed to execute. This relies on hardware timer interrupts and is completely transparent to the processes.
- Linux schedule process according to a priority ranking this is a "goodness" ranking. Linux uses dynamic priorities, i.e. priorities are adjusted over time to eliminate starvation.

- Processes that have not received the CPU for a long time get their priorities increased, processes that have received the CPU often get their priorities decreased.
- Linux uses process preemption, a process is preempted when its time quantum has expired and new process enters TASK_RUNNING state and its priority is greater than the priority of the currently running process.
- The preempted process is not suspended, it is still in the ready queue, it simply no longer has the CPU.
- Consider a text editor and a complier, since the text editor is an interactive program, its dynamic priority is higher than the complier.
- The text editor will be block often since it is waiting for I/O. When the I/O interrupt receives a key-press for the editor is put on the ready queue and the scheduler is called since the editor's priority is higher than the complier. The editor getes the input and quickly blocks for more I/O.
- The Linux scheduling algorithm is not based on a continuous CPU time axis, instead it divides the CPU time into epochs. An epoch is a division of time or a period of time.
- In a single epoch, every process has a specified time quantum that is computed at the beginning of each epoch. This is the maximum CPU time that process can use during the current epoch.
- A process only uses its quantum when it is executing on the CPU, when the process is waiting for I/O its quantum is not used. As a result, a process, can get the CPU many times in one epoch, until its quantum is full used.
- A epoch ends when all runnable processes have used all of their quantum. A new epoch starts and all process get a new quantum.

2. **Interruptible :** It is blocked state of process. The process is waiting for an event or signal from another process.
3. **Uninterruptible :** It is also blocked state. Here process is waiting directly on hardware condition. Process can not handle any signals in this state.
4. **Stopped :** Process has been halted and can only resume by positive action from another process.
5. **Zombie :** Process has been terminated for some reason.

Q.14 Explain data structure used by task_struct.

Ans. : • Process in Linux is represented by task_struct data structure. It contain various information.

State	Various state of process execution : Ready, executing, suspended, zombie and stopped.
IPC	Linux support IPC mechanism.
Scheduling information	Information needed by Linux to schedule processes. Process can be real time and a priority.
Links	Each process contains link to its parent and to its all children process.
Identifier	Each process has a unique process identifier (PID). It also user and group ID.
Times	It includes process creation time and time consumed by process.
File system	Includes pointers to any files opened by this process.

Q.15 Discuss Linux process scheduling policy.

Ans. :

- Linux uses a timesharing technique. We know that this means that each process is assigned a small quantum or time slice that it is allowed to execute. This relies on hardware timer interrupts and is completely transparent to the processes.
- Linux schedule process according to a priority ranking this is a "goodness" ranking. Linux uses dynamic priorities, i.e. priorities are adjusted over time to eliminate starvation.

- Processes that have not received the CPU for a long time get their priorities increased, processes that have received the CPU often get their priorities decreased.
- Linux uses process preemption, a process is preempted when its time quantum has expired and new process enters TASK_RUNNING state and its priority is greater than the priority of the currently running process.
- The preempted process is not suspended, it is still in the ready queue, it simply no longer has the CPU.
- Consider a text editor and a complier, since the text editor is an interactive program, its dynamic priority is higher than the complier.
- The text editor will be block often since it is waiting for I/O. When the I/O interrupt receives a key-press for the editor is put on the ready queue and the scheduler is called since the editor's priority is higher than the complier. The editor getes the input and quickly blocks for more I/O.
- The Linux scheduling algorithm is not based on a continuous CPU time axis, instead it divides the CPU time into epochs. An epoch is a division of time or a period of time.
- In a single epoch, every process has a specified time quantum that is computed at the beginning of each epoch. This is the maximum CPU time that process can use during the current epoch.
- A process only uses its quantum when it is executing on the CPU, when the process is waiting for I/O its quantum is not used. As a result, a process, can get the CPU many times in one epoch, until its quantum is full used.
- A epoch ends when all runnable processes have used all of their quantum. A new epoch starts and all process get a new quantum.

Q.16 Explain completely fair scheduler of Linux.

Ans. : • Linux kernel version 2.6.23, a new approach has been taken to the scheduling to runnable processes. The Completely Fair Scheduler (CFS) becomes the default Linux kernel scheduler.

- Completely Fair Scheduler does allocation of CPU resources fairly without compromising interactivity performance. Also CFS offers user fair scheduling, group scheduling and modular scheduler framework.
- CFS tries to assure that each process obtains its fair share of the processor time.
- Model process scheduling as if the system had an ideal, perfectly multitasking processor. In such a system, each process would receive 1/n of the processor's time, where n is the number of runnable processes, and we should schedule them for infinitely small durations, so that in any measurable period we'd have run all n processes for the same amount of time.
- CFS will run each process for some amount of time, round-robin, selecting next the process that has run the least. Rather than assign each process a time slice, CFS calculates how long a process should run as a function of the total number of runnable processes. Instead of using the nice value to calculate a time slice, CFS uses the nice value to weight the proportion of processor a process is to receive.
- The CFS tries to keep track of the fair share of the CPU that would have been available to each process in the system. So, CFS runs a fair clock at a fraction of real CPU clock speed.
- The fair clock's rate of increase is calculated by dividing the wall time by the total number of processor waiting. The resulting value is the amount of CPU time to which each process is entitled.
- As a process waits for the CPU, the scheduler tracks the amount of time it would have used on the ideal processor.
- The wait time is represented by per-task wait_runtime. This is used to rank the processes for scheduling and to determine the amount of time the process is allowed to execute before being preempted.

- Scheduler selects the process with longest wait time and assigned to the CPU. Wait time decreases when process is running.
- At the same time, time of other waiting tasks increases. After some time, there will be another task with largest wait time and the currently running task will be preempted.
- Using this principle, CFS tries to be fair to all tasks and always tries to have a system with zero wait time for each process.

Q.17 Write short note on process management in Linux.

Ans. : • In Linux, tasks represent both processes and threads. Linux threads are really kernel threads.

- To manage multi-tasking, the OS needs to use a data structure which can keep track of every task's progress and usage of the computer's available resources. Such a data structure is called a 'process descriptor' and every active task needs one.
- Every task needs its own 'private' stack. So every task in addition to having its own code and data will also have a stack area that is located in user space plus another stack area that is located in kernel space.
- Each task also has a process descriptor which is accessible only in kernel space.
- The **task_struct** is used to represent a task.

1. **State :** Process execution states are executing, ready, suspended, stopped, zombie.
2. **Scheduling information :** Linux uses this information for scheduling processes.
3. **Identifiers :** Each process has a unique process identifier and also has user and group identifiers.
4. **IPC :** Linux supports the IPC mechanisms are pipes, shared memory, socket etc.
5. **Links :** In a Linux system no process is independent of any other process. Every process in the system, except the initial process has a parent process. New processes are not created, they are copied or rather cloned from previous processes. Every task_struct

representing a process keeps pointers to its parent process and to its siblings as well as to its own child processes.

6. **Times and timers :** The kernel keeps track of processes creation time as well as the CPU time that it consumes during its lifetime. The kernel updates the amount of time.
7. **File system :** Processes can open and close files as they wish and the processes task_struct contains pointers to descriptors for each open file.
8. **Virtual memory :** Most processes have some virtual memory and the Linux kernel must track how that virtual memory is mapped onto the system's physical memory.
9. **Processor specific context :** When a process is suspended, all of that CPU specific context must be saved in the task_struct for the process. When a process is restarted by the scheduler its context is restored from here.

Q.18 Write short note on scheduling in Linux.

Ans. : • Scheduling is based on the threads, not on the process. Linux system uses three classes of threads for scheduling :

1. Real time FIFO
2. Real time round robin
3. Timesharing

- Linux implements the **FIFO** and **round-robin real-time scheduling** classes; in both cases, each process has a priority in addition to its scheduling class. The scheduler runs the process with the highest priority; for equal-priority processes, it runs the process waiting the longest. FIFO processes continue to run until they either exit or block.
- Linux uses 140 priority levels. Threads are internally represented by priority levels from 0 to 99. These priority is also called static priority. The priority 0 is highest priority and 99 is the lowest real time priority level.
- Non-real time threads are associated with priority levels from 100 to 139. These are called dynamic priority. Sum of the base time quantum and of the number of ticks of CPU time left to the process before its quantum expires in the current **epoch.** The number of ticks left to the

parent is split in two halves, one for the parent, one for the child. This prevents processes from getting an unlimited amount of CPU time.

- Linux uses dynamically assigned process priorities for non real-time processes. Processes running for a long time have their priorities decreased while processes that are waiting have their priorities increased dynamically.
- Data structure used by Linux scheduler is **runqueue.** There are two sets of 140 queues, active and expired. The system only runs processes from active queues, and puts them on expired queues when they use up their quanta.
- When a priority level of the active queue is empty, the scheduler looks for the next-highest priority queue. After running all of the active queues, the active and expired queues are swapped. There are pointers to the current arrays; at the end of a cycle, the pointers are switched.
- A scheduler runqueue is a list of tasks that are runnable on a particular CPU. A rq structure maintains a linked list of those tasks. The runqueues are maintained as an array runqueues, indexed by the CPU number.

Avoiding starvation

- The system only runs processes from active queues, and puts them on expired queues when they use up their quanta. When a priority level of the active queue is empty, the scheduler looks for the next-highest priority queue.
- After running all of the active queues, the active and expired queues are swapped. There are pointers to the current arrays; at the end of a cycle, the pointers are switched.

Scheduling priority

- Tasks are assigned a static priority. It is also called nice value. Task priority affects the size of its time slice and the order in which it executes on a processor.
- A task's effective priority determines the level of the priority array in which a task is placed.

Q.19 What are the types of process management system call in Linux ?

Ans. : Process management system calls in Linux.

- fork : For creating a duplicate process from the parent process.
- wait : Processes are supposed to wait for other processes to complete their work.
- exec : Loads the selected program into the memory.
- exit : Terminates the process.

Q.20 Explain fork() system call.

Ans. : • The `fork()' used to create a new process from an existing process. The new process is called the child process, and the existing process is called the parent. The parent gets the child's pid returned to him, but the child gets 0 returned to him.

6.5 : Booting

Q.21 Explain Linux booting with master boot record.

Ans. : • PC's have simplistic ROM codes compared to firmware found on UNIX machines.

- UNIX firmware generally knows what devices are installed, how to configure them and use them on a basic level.
- When machine boots, it begins by executing code stored in ROMs. The exact location and nature of this code varies, depending on the type of machine you have. For UNIX type machine, the code is typically firmware that knows how to use devices connected to the machine, how to talk to the network on a basic level, and how to understand disk based file systems.
- BIOS (Basic Input / Output System) refer to the software code run by a computer when first powered on. The primary function of BIOS is code program embedded on a chip that recognizes and controls various devices that make up the computer.

- Actually PC has a several levels of BIOS : One for machine itself, one for video card, one for SCSI card if the system has one.

MBR (Master Boot Record)

- OS is booted from a hard disk, where the Master Boot Record (MBR) contains the primary boot loader. The MBR is a 512-byte sector, located in the first sector on the disk (sector 1 of cylinder 0, head 0).
- After the MBR is loaded into RAM, the BIOS yields control to it. Fig. Q.21.1 shows MBR.

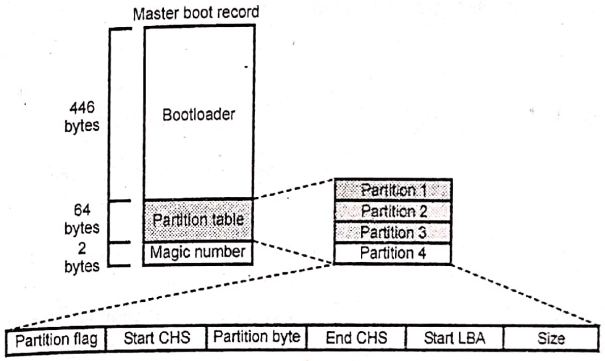


Fig. Q.21.1 MBR

- The first 446 bytes are the primary boot loader, which contains both executable code and error message text. The next sixty-four bytes are the partition table, which contains a record for each of four partitions.
- The MBR ends with two bytes that are defined as the magic number (0xAA55). The magic number serves as a validation check of the MBR.
- Boot loader could be more rightly called the kernel loader. The task at this stage is to load the Linux kernel. GRUB and LILO are the most popular Linux boot loader.
- A popular Linux boot loader is **LILO. LILO** can be installed on either the **MBR** or the boot record of the Linux root partition. Installing **LILO** on the boot record allows you to use a different boot loader for another operating system like Windows XP.

- **LILO** can be configured through the **"lilo"** command. Its contents are stored at **"/etc/lilo.conf"**. See the man pages for more details on how to configure **LILO.**
- The default MBR contains a simple program that tells the computer to get its boot loader from the first partition on the disk. Once the MBR has chosen a partition to boot from, it tries to load the boot loader specific to that partition. This loader is then responsible for loading the kernel.

Q.22 Explain different run level of Linux.

Ans. :

Run level	Name	Descriptions
0	Halt	Shut down all services when the system will not be restarted.
1	Single user mode	Used for system maintenance. Do not require logging in with a user name and password. It operates as root but do not provide networking capabilities.
2	Multi-user mode without networking enabled	Rarely used except for system maintenance or testing.
3	Regular multi-user networking mode	Standard multi-user text mode operation. Non-graphical systems use this mode for normal operation.
4	--	Not used
5	Graphical login	Identical to run level 3 except a graphical login is used.
6	Reboot	Shuts down all services so the system can be restarted.

Q.23 What is GRUB ? List the features of GRUB.

Ans. : • GRUB is a Multiboot boot loader. It was derived from GRUB, the GRand Unified Bootloader, which was originally designed and

implemented by *Erich Stefan Boleyn.* It is default boot loader for most UNIX and Linux system with Intel processors.

- GRUB will work well with Linux, DOS, Windows, or BSD. GRUB stands for GRand Unified Bootloader.
- Briefly, a boot loader is the first software program that runs when a computer starts. It is responsible for loading and transferring control to the operating system kernel software. The kernel, in turn, initializes the rest of the operating system (e.g. GNU).
- GRUB is an operating system independent boot loader. It is flexible command line interface and supports multiple executable formats. It also supports diskless system.
- GRUB is dynamically configurable. This means that the user can make changes during the boot time, which include altering existing boot entries, adding new, custom entries, selecting different kernels, or modifying *initrd.* GRUB also supports Logical Block Address mode. This means that if your computer has a fairly modern BIOS that can access more than 8 GB (first 1024 cylinders) of hard disk space, GRUB will automatically be able to access all of it.
- GRUB can be run from or be installed to any device *(floppy disk, hard disk, CD-ROM, USB drive, network drive)* and can load operating systems from just as many locations, including network drives. It can also decompress operating system images before booting them.
- GRUB has the following features :
  1. Recognize multiple executable formats
  2. Support non-multiboot kernels
  3. Load multiples modules
  4. Load a configuration file
  5. Provide a menu interface
  6. Have a flexible command-line interface
  7. Support multiple file system types
  8. Support logical block address mode

END...

SOLVED MODEL QUESTION PAPER (In Sem)
Operating Systems
S.E. (AI&DS) Semester - III [As Per 2020 Pattern]

Time : 1 Hour] **[Maximum Marks : 30**

N.B. : i) Attempt Q.1 or Q.2, Q.3 or Q.4.

ii) Neat diagrams must be drawn wherever necessary.

iii) Figures to the right side indicate full marks.

iv) Assume suitable data, if necessary.

Q.1 a) *Explain the following shell commands with example.*
i) Chmod ii) Grep iii) Cat iv) Sort. **(Refer Q.18 of Chapter - 1)** **[4]**

b) *What are the benefits of resource abstraction ?*
(Refer Q.3 of Chapter - 1) **[3]**

c) *What is system call ? Explain working of system call.*
(Refer Q.13 of Chapter - 1) **[8]**

OR

Q.2 a) *State command line arguments in shell with example.*
(Refer Q.21 of Chapter - 1) **[5]**

b) *What is real time OS ? Explain its types with suitable example.*
(Refer Q.11 of Chapter - 1) **[4]**

c) *Describe in brief the evolution of operating system.*
(Refer Q.5 of Chapter - 1) **[6]**

Q.3 a) *What is purpose of IPC ?* **(Refer Q.20 of Chapter - 2)** **[3]**

b) *What should be scheduling criteria for scheduling algorithm ?*
(Refer Q.11 of Chapter - 2) **[4]**

c) *For the table given below calculate average waiting time and average turnaround time and draw a Gantt Chart illustrating the process execution using following scheduling algorithms.*
i) RR (Time slice - 2 units) ii) SJF (non - preemptive)

Process	Arrival Time	Burst Time
P_1	0	8
P_2	1	5
P_3	3	3
P_4	4	1
P_5	6	4

(Refer Q.14 of Chapter - 2) **[8]**

OR

Q.4 a) *How PCB helps in process state management ? Explain the structure of PCB.* **(Refer Q.2 of Chapter - 2)** **[5]**

b) *Explain thread life cycle.* **(Refer Q.24 of Chapter - 2)** **[4]**

c) *Explain different types of schedulers in operating system.*
(Refer Q.8 of Chapter - 2) **[6]**

SOLVED MODEL QUESTION PAPER (End Sem)
Operating Systems
S.E. (AI&DS) Semester - III [As Per 2020 Pattern]

Time : 2 $\frac{1}{2}$ Hours] **[Maximum Marks : 70**

N.B. : i) Attempt Q.1 or Q.2, Q.3 or Q.4, Q.5 or Q.6, Q.7 or Q.8.

ii) Neat diagrams must be drawn wherever necessary.

iii) Figures to the right side indicate full marks.

iv) Assume suitable data, if necessary.

Q.1 a) *Explain with definition, the concept of general and binary semaphore.* **(Refer Q.7 of Chapter - 3)** **[6]**

b) *Explain with an appropriate example, how resource allocation graph determines a deadlock.* **(Refer Q.25 of Chapter - 3)** **[6]**

c) *Explain monitor in brief.* **(Refer Q.11 of Chapter - 3)** [6]

OR

Q.2 a) *Explain the following terms :*
i) Mutual exclusion ii) Synchronization iii) Race condition
(Refer Q.4 of Chapter - 3) [6]

b) *Consider the following snapshot of a system. Answer the following questions using Banker's algorithm.*

	Allocation	Maximum	Available
	A B C D	A B C D	A B C D
P_0	0 0 1 2	0 0 1 2	1 5 2 0
P_1	1 0 0 0	1 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	

i) What are the contents of Need matrix ?
ii) Is the system in a safe state ? **(Refer Q.28 of Chapter - 3)** [6]

c) *What is critical section ? Give semaphore solution for producer - consumer problem.* **(Refer Q.16 of Chapter - 3)** [6]

Q.3 a) *Explain different ways to remove external fragmentation.*
(Refer Q.9 of Chapter - 4) [6]

b) *What is paging ? Explain in details.*
(Refer Q.14 of Chapter - 4) [6]

c) *What is page replacement ? List the name of page replacement algorithm.* **(Refer Q.27 of Chapter - 4)** [5]

OR

Q.4 a) *Explain fixed partition of memory.*
(Refer Q.4 of Chapter - 4) [5]

b) *For the following reference string.*
1,2,3,4,2,1,5,6,2,1,2,3,3,6. Count the number of page faults that occur with 3 frames using FIFO and LRU page replacement methods. Discuss the result. **(Refer Q.28 of Chapter - 4)** [7]

c) *Describe the address translation mechanism in segmentation with suitable diagram.* **(Refer Q.19 of Chapter - 4)** [5]

Q.5 a) *Explain SSTF and LOOK disk scheduling algorithm.*
(Refer Q.7 of Chapter - 5) [8]

b) *Is disk scheduling, other than FCFS useful in a single user environment. Explain your answer.* **(Refer Q.12 of Chapter - 5)** [6]

c) *Enlist the characteristics of block and character devices. Explain each with suitable example.* **(Refer Q.1 of Chapter - 5)** [4]

OR

Q.6 a) *List and explain in brief I/O performing techniques (at least three).* **(Refer Q.3 of Chapter - 5)** [12]

b) *Briefly describe SCAN disk scheduling algorithm.*
(Refer Q.9 of Chapter - 5) [6]

Q.7 a) *Explain driver-registration system and conflict-resolution mechanism.* **(Refer Q.9 of Chapter - 6)** [6]

b) *Explain process state transition diagram of Linux.*
(Refer Q.13 of Chapter - 6) [6]

c) *Write short note on shell.* **(Refer Q.5 of Chapter - 6)** [5]

OR

Q.8 a) *List and explain components of Linux.*
(Refer Q.2 of Chapter - 6) [6]

b) *What is kernel module ? Explain module management system.*
(Refer Q.8 of Chapter - 6) [8]

c) *Explain design principles of Linux.*
(Refer Q.3 of Chapter - 6) [3]

END...