

EXAM: IV SEM **PAPER:** 20BCASD/NT/AI34
SUBJECT: OPERATING SYSTEM **CLASS:** BCA
MAXIMUM: 50 **TIME:** 2H

WEIGHTAGE TO OBJECTIVES TABLE

SL.N O	OBJECTIVES	MARK S	% MARKS
1.	Knowledge (Remembering)	5	10
2.	Understanding	25	50
3.	Application	15	30
4.	Skill	10	20
Total		50	100

EXAM: IV SEM **PAPER:** 20BCASD/AI/NT34
SUBJECT: OPERATING SYSTEM **CLASS:** BCA
MAXIMUM: 50 **TIME:** 2H

BLUE PRINT

Unit	REMEMBERING			UNDERSTAND			APPLICATION			SKILL			Total
	OT	SA	Unit wise Marks	OT	SA	Unit wise Marks	OT	SA	Unit wise Mark s	OT	SA	Unit wise Marks	
1	1(1)	1(4)	5	1(1)	1(4)	5							10
2				1(1)	1(4)	5	1(1)	1(4)	5				10
3				1(1)	1(4)	5	1(1)	1(4)	5				10
4				1(1)	1(4)	5		1(4)	4	1(1)		1	10
5							1(1)		1	1(1)	2(4)	9	10
	05			20				15		10			50

BLUE PRINT

EXAM: IV SEM **PAPER:** 20BCASD/AI/NT34
SUBJECT: OPERATING SYSTEM **CLASS:** BCA

MAXIMUM: 50 **TIME:** 2H

Unit	REMEMBERING		UNDERSTANDING		APPLICATIONS		SKILL		Total
	OT	SA	OT	SA	OT	SA	OT	S A	
1	1(1)	1(4)	1(1)	1(4)					10
2			1(1)	1(4)	1(1)	1(4)			10
3			1(1)	1(4)	1(1)	1(4)			10
4			1(1)	1(4)		1(4)	1(1)		10
5					1(1)		1(1)	2(4)	10
	05		20				15		10

UNIT 1

MULTIPLE CHOICE QUESTIONS REMEMBERING

1. Which among the following are the components of computer system
 - a. Hardware
 - b. Operating System
 - c. Application
 - d. **All the above**
2. Give the full form of CPU
 - a. **Central Processing Unit**
 - b. Central Processing User
 - c. Central Program Unit
 - d. Central Program User
3. Which system is the logical extension of multi-programming
 - a. Batch Processing System
 - b. Multi Programming Systems
 - c. **Time Sharing System**
 - d. Real Time System
4. Which system is less restrictive
 - a. **Soft Real Time**
 - b. Hard Real Time
 - c. Both of them
 - d. None of the two
5. Which system is used when there is rigid time requirement
 - a. Batch Processing System
 - b. Multi Programming Systems
 - c. Time Sharing System
 - d. **Real Time System**
6. Which of the following is passive entity
 - a. **Program**
 - b. Process
 - c. User process
 - d. All the above
7. Which of the following is active entity
 - a. Program
 - b. **Process**
 - c. File
 - d. All the above
8. The process being created is called as
 - a. **New**
 - b. Running
 - c. Waiting

d. Ready

9. Instruction being executed is called as

- a. New
- b. **Running**
- c. Waiting
- d. Ready

10. The process waiting for some event to occur is called as

- a. New
- b. Running
- c. **Waiting**
- d. Ready

11. The process waiting to be assigned to a processor is called as

- a. New
- b. Running
- c. Waiting
- d. **Ready**

12. Give the Full form of PCB

- a. **Process Control Block**
- b. Program Control Block
- c. Process Care Block
- d. Program Control Block

UNDERSTANDING

13. The process that enters into a system are put in

- a. Ready Queue
- b. **Job Queue**
- c. Device Queue
- d. None of the above

14. The process that are residing in main memory and are ready and waiting to execute are kept on a list called

- a. **Ready Queue**
- b. Job Queue
- c. Device Queue
- d. None of the above

15. The list of process waiting for I/O device is called

- a. Ready Queue
- b. Job Queue
- c. **Device Queue**
- d. None of the above

16. How do we represent “queue” in queuing diagram

- a. **Rectangle**
- b. Square

- c. Circle
- d. Arrow

17. How do we represent the resources that serve the queues in queuing diagram

- a. Rectangle
- b. Square
- c. Circle**
- d. Arrow

18. How the flow of process is been indicated

- a. Rectangle
- b. Square
- c. Circle
- d. Arrow**

19. Long term Scheduler is also called as

- a. Job Scheduler**
- b. Short-term scheduler
- c. CPU Scheduler
- d. None of the above

20. Short term Scheduler is also called as

- a. Job Scheduler
- b. Short-term scheduler
- c. CPU Scheduler**
- d. None of the above

21. The process swapped out and is later swapped in is done by which scheduler

- a. Long-term scheduler
- b. Short-term scheduler
- c. Medium-term scheduler**
- d. None of the above

22. Dividing the system function into separate processor or thread is called as

- a. Information Sharing
- b. Computation Speedup
- c. Modularity**
- d. Convenience

23. Give the full form of LWP

- a. Light Weight Process**
- b. Light Weight Program
- c. Lengthy Weight Process
- d. Lengthy Weight Program

24. A Traditional Process has how many threads in control

- a. 1**
- b. 2
- c. 3

d. 4

25. Traditional process is also called as

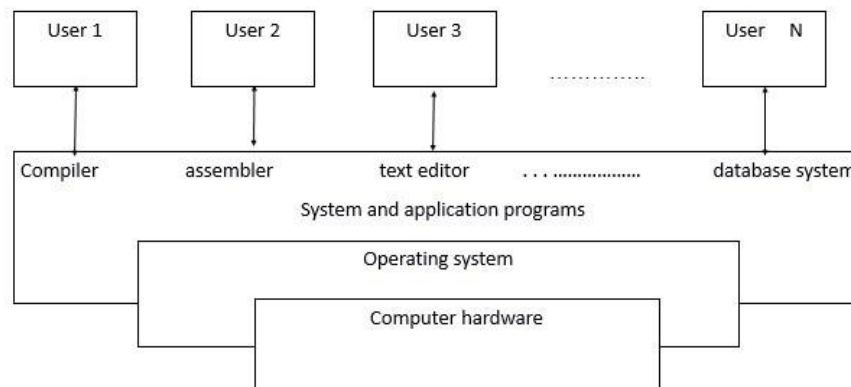
- a. **Heavy weight Process**
- b. Light weight Process
- c. Multi Process
- d. None of the above

FOUR MARKS QUESTIONS

REMEMBERING

1. **Give the brief introduction about operating system with neat diagram.**

An operating system is a program that manages the computer hardware. It also provides a basis for application programs and acts as an intermediary between the computer user and the computer hardware. An amazing aspect of operating systems is how varied they are in accomplishing these tasks. Mainframe operating systems are designed primarily to optimize utilization of hardware. Personal computer (PC) operating systems support complex games, business applications, and everything in between. Operating systems for hand held computers are designed to provide an environment in which a user can easily interface with the computer to execute programs.

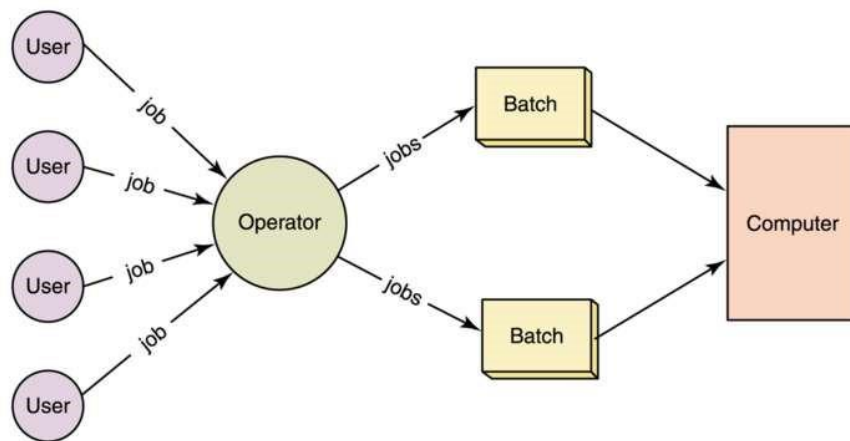


A computer system can be divided roughly into four components: the hardware, the operating system, the application programs, and the users. The hardware - the central processing unit (CPU), the memory, and the input/output (I/O) devices - provides the basic computing resources for the system. The application programs - such as word processors, spreadsheets, compilers, and Web browsers - define the ways in which these resources are used to solve users' computing problems.

The operating system controls the hardware and coordinates its use among the various application programs for the various users. We can also view a computer system as consisting of hardware, software, and data. The operating system provides the means for proper use of these resources in the operation of the computer system.

2. **Explain Batch processing system.**

Jobs that can run without end user interaction, or can be scheduled to run as resources permit, are called batch jobs. Batch processing is basically for those frequently used programs that can be run with minimal human interaction. A program that reads a large file and generates a report, for example, is considered to be a batch job.



Batch operating system users do not interact with the computer directly. Each job is prepared by each user on an off-line device like punch cards and submits it to the computer operator. To increase the processing speed, jobs with similar needs are batched together and run as a group. After completing the coding for the programs the programmers hand over their programs to the operator and the operator then analyze the programs having similar requirements and then divide them into batches.

3. Briefly explain Multi Programming system.

The most important aspect of job scheduling is the ability to multi program. A single user cannot keep either the CPU or I/O devices busy all the time. Multiprogramming increases CPU utilization by organizing jobs so that CPU always busy with executing one job. The idea is as follows - the OS keeps several jobs in memory simultaneously. The set of jobs is a subset of jobs in the job pool. The OS picks and executes one of the jobs in the memory. The job may have to wait for some task such as an I/O operation to complete. In non-multi programmed environment, the CPU would sit idle. In multi-programmed environment the OS switches to another job. When that job needs to wait, the CPU switches to another job and so on. Eventually the first job finishes waiting and gets the CPU. As long as there is at least one job to execute the CPU never sits idle. Multi-programming is the first instance where the OS has to make decisions for the user.

All the jobs that enter the system are kept in a job pool. A pool is a set of all processes residing on the disk awaiting allocation of main memory. If several jobs are ready to be brought into memory and if there is not enough space for all of them, then the system must choose from among them. This decision making capability is called job scheduling. When the OS selects a job from job pool, it loads that job into memory for execution. If several jobs are ready to run at the same time, the system must choose among them. This decision making capability of OS is called CPU scheduling.

Operating System

Job1

Job2

Job3

Job4

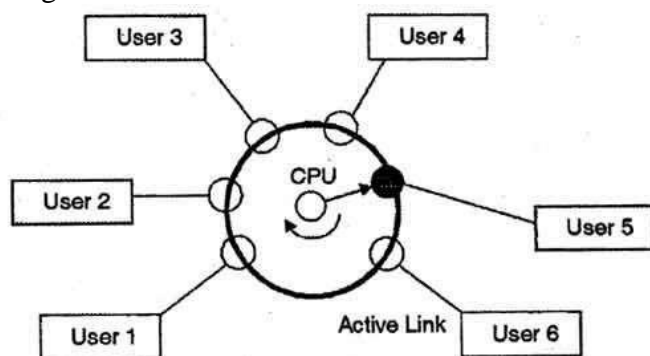
4. Write a note on time sharing system.

A time-sharing system allows many users to share the computer resources simultaneously. In other words, time-sharing refers to the allocation of computer resources in time slots to several programs simultaneously. For example a mainframe computer that has many users logged on

to it. Each user uses the resources of the mainframe -i.e. memory, CPU etc. The users feel that they are exclusive user of the CPU, even though this is not possible with one CPU i.e. shared among different users.

The time sharing systems were developed to provide an interactive use of the computer system. A time shared system uses CPU scheduling and multiprogramming to provide each user with a small portion of a time-shared computer. It allows many users to share the computer resources simultaneously. As the system switches rapidly from one user to the other, a short time slot is given to each user for their executions.

The time-sharing system provides the direct access to a large number of users where CPU time is divided among all the users on scheduled basis. The OS allocates a set of time to each user. When this time is expired, it passes control to the next user on the system. The time allowed is extremely small and the users are given the impression that they each have their own CPU and they are the sole owner of the CPU. This short period of time during that a user gets attention of the CPU; is known as a time slice or a quantum. The concept of time-sharing system is shown in figure.



In above figure the user 5 is active but user 1, user 2, user 3, and user 4 are in waiting state whereas user 6 is in ready status. As soon as the time slice of user 5 is completed, the control moves on to the next ready user i.e. user 6. In this state user 2, user 3, user 4, and user 5 are in waiting state and user 1 is in ready state. The process continues in the same way and so on.

5. Explain the following system components.

a. Process management

A process can be thought of a program in execution. Example: word processing application runs by an individual, a system task of sending output to a printer. A process needs certain resources - CPU time, memory, files and I/O devices to complete a task. These resources are given to it when the process is created or while it is running. A program is a passive entity such as contents of a file stored on a disk. A process is an active entity with a program counter specifying the next instruction to be executed.

The operating system is responsible for the following activities in connection with processes managed.

- The creation and deletion of both user and system processes
- The suspension and resumption of processes.
- The provision of mechanisms for process synchronization
- The provision of mechanisms for deadlock handling.
- Providing mechanism for process communication

b. Main Memory Management

Memory is central to the operation of a modern computer system. Memory is a large array of words or bytes, each with its own address. Interaction is achieved through a sequence of reads

or writes of specific memory address. The CPU fetches from and stores in memory. In order for a program to be executed it must be mapped to absolute addresses and loaded in to memory. As the program executes, it accesses program instructions and data from memory by generating these absolute is declared available, and the next program may be loaded and executed. In order to improve both the utilization of CPU and the speed of the computer's response to its users, several processes must be kept in memory.

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

- Keep track of which parts of memory are currently being used and by whom.
- Decide which processes are to be loaded into memory when memory space becomes available.
- Allocate and deallocate memory space as needed.

6. Explain two viewpoints of operating system.

An OS can be explored in two viewpoints: User View and System View.

USER VIEW

The user view of a computer varies by the interfaces being used. In the view where the user sits in front of the PC, performance is important to the user but does not matter about the system being idle or waiting for the slow I/O speed of the user. Here, the OS is designed mostly for ease of use and performance. Some users may be at terminals connected to mainframes. Others may access the same computer through other terminals. The users here share resources and exchange information. The OS in this case is designed to maximize resource utilization - to ensure that CPU time, memory and I/O are used efficiently. Some users may be at workstations connected to networks of other work stations and servers. These users have dedicated resources and at the same time share resources. The OS in this case is designed to compromise between individual usability and resource utilization. Some computers have little or no user view. For example, embedded computers in home devices and automobiles may have numeric keypads and may turn indicator lights on or off to show status, but they and their operating systems are designed primarily to run without user intervention.

SYSTEM VIEW

From computers point of view, an OS is the most intimate with the hardware. We can view OS as a resource allocator. A computer has many resources that may be required to solve a problem - CPU time, memory space and I/O devices. The OS acts as a manager of these resources. It decides on how to allocate these resources to specific programs and users so that it can operate the computer system efficiently. A slightly different view is the need to control the various I/O devices and user programs. An OS acts as a control program. A control program manages the execution of various user programs to prevent errors and improper use of computer. Thus, the common function of controlling and allocating the resources are brought together into one piece of software called Operating Systems.

7. Explain the following system components.

a. File management

A file is a collection of related information defined by its creator. Commonly, files represent programs (both source and object forms) and data. Data files may be numeric, alphabetic or alphanumeric. Files may be free-form, such as text files, or may be rigidly formatted. In general a files is a sequence of bits, bytes, lines or records whose meaning is defined by its creator and user. It is a very general concept. The operating system implements the abstract concept of the file by managing mass storage device, such as tapes and disks. Also files are normally

organized into directories to ease their use. Finally, when multiple users have access to files, it may be desirable to control by whom and in what ways files may be accessed.

The operating system is responsible for the following activities in connection with file management:

- The creation and deletion of files
- The creation and deletion of directory
- The support of primitives for manipulating files and directories
- The mapping of files onto disk storage.
- Backup of files on stable (non volatile) storage

b. I/O System Management

One of the purposes of an operating system is to hide the peculiarities of specific hardware devices from the user. For example, in Unix, the peculiarities of I/O devices are hidden from the bulk of the operating system itself by the I/O system. The I/O system consists of:

- A buffer caching system
- A general device driver code
- Drivers for specific hardware devices.

8. Explain the following system components.

a. Secondary storage management

The main purpose of a computer system is to execute programs. These programs, together with the data they access, must be in main memory during execution. Since the main memory is too small to permanently accommodate all data and program, the computer system must provide secondary storage to backup main memory. Most modern computer systems use disks as the primary on-line storage of information, of both programs and data. Most programs, like compilers, assemblers, sort routines, editors, formatters, and so on, are stored on the disk until loaded into memory, and then use the disk as both the source and destination of their processing.

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

- Free space management
- Storage allocation
- Disk scheduling.

b. Protection

The various processes in an operating system must be protected from each others activities. For that purpose, various mechanisms which can be used to ensure that the files, memory segment, CPU and other resources can be operated on only by those processes that have gained proper authorization from the operating system. Protection refers to a mechanism for controlling the access of programs, processes, or users to the resources defined by a computer controls to be imposed, together with some means of enforcement. Protection can improve reliability by detecting latent errors at the interfaces between component subsystems. Early detection of interface errors can often prevent contamination of a healthy subsystem by a subsystem that is malfunctioning. An unprotected resource cannot defend against use (or misuse) by an unauthorized or incompetent user.

UNDERSTANDING

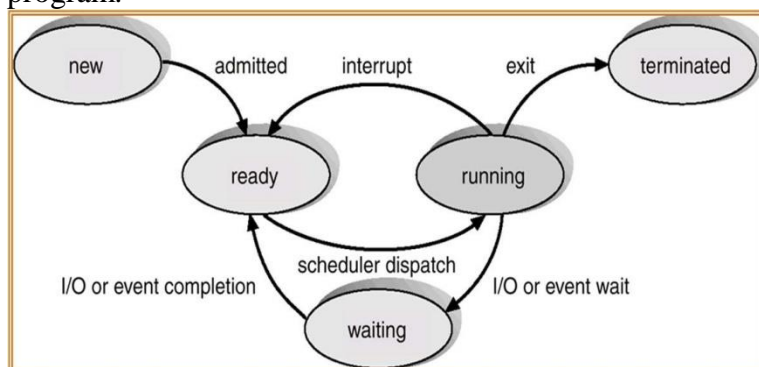
9. Briefly explain Operating System services.

An OS provides an environment for the execution of program. It provides certain services to the program and the user of the program. The services are provided for the convenience of the programmer to make programming task easier. The services provided by the OS are

- **Program execution:** The system must be able to load a program into memory and to run the program. The program must be able to end its execution either normally or abnormally (by indicating an error)
- **I/O operations:** A running program may require I/O. This I/O may be a file or an I/O device. Users cannot control I/O devices directly. Hence, an OS must provide a means for I/O operation.
- **File system manipulation:** The OS must provide means to manage and work with file.
- **Communication:** One process needs to exchange information with another process. This could take place between the processes within the same computer or between different computers tied through a network. Communication can be implemented by shared memory or by message passing, in which packets of information are moved between processes by OS
- **Error detection:** Errors may occur in CPU, memory hardware, I/O devices or user programs. For each type of error, the OS should take appropriate action to ensure correct and consistent computing.

10. **Briefly explain Process concepts. Explain different Process states with neat diagram.**

For a program to be executed, a process, or task, is created for that program. From the processor's point of view, it executes instructions from its repertoire in some sequence dictated by the changing values in the program counter register. From the point of view of an individual program, its execution involves a sequence of instructions within that program.



The five states in this new diagram are:

- **Running:** The process that is currently being executed. For this chapter, we will assume a computer with a single processor, so at most one process at a time can be in this state.
- **Ready:** A process that is prepared to execute when given the opportunity.
- **Blocked/Waiting:** A process that cannot execute until some event occurs, such as the completion of an I/O operation.
- **New:** A process that has just been created but has not yet been admitted to the pool of executable processes by the OS. Typically, a new process has not yet been loaded into main memory, although its process control block has been created.
- **Terminate:** A process that has been released from the pool of executable processes by the OS, either because it halted or because it aborted for some reason.

11. **Explain Process control block with neat diagram.**

Two essential elements of a process are program code (which may be shared with other processes that are executing the same program) and a set of data associated with that code. Let us suppose that the processor begins to execute this program code, and we refer to this executing entity as a process. At any given point in time, while the program is executing, this process can be uniquely characterized by a number of elements, including the following:

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

- **Process State:** State may enter into new, ready, running, waiting, halted.
- **Process Number (PID):** A unique identification number for each process in the operating system (also known as Process ID).
- **Program Counter (PC):** A pointer to the address of the next instruction to be executed for this process.
- **CPU Registers:** Indicates various register set of CPU where process need to be stored for execution for running state.
- **CPU Scheduling Information:** Indicates the information of a process like process priority, pointers to scheduling queue and any other scheduling parameters.
- **Memory Limit:** It specifies the memory limits
- **List of Open File:** Provides information about files that are open

12. **Write a note on Process scheduling with neat diagram.**

The Operating System maintains the following important process scheduling queues –

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

A common representation of process scheduling is a queuing diagram. Each rectangular box represents a queue. Two types of queues are present: the ready queue and a set of device queues. The circles represent the resources that serve the queues, and the arrows indicate the flow of processes in the system.

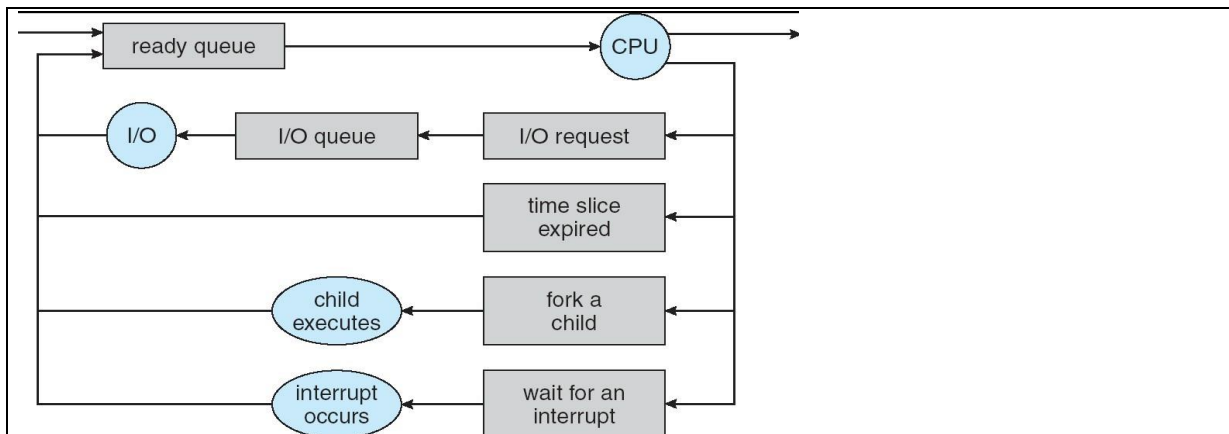


Figure: Queuing-diagram representation of process scheduling

A new process is initially put in the ready queue. It waits in the ready queue until it is selected for execution (or dispatched). Once the process is assigned to the CPU and is executing, one of several events could occur:

- The process could issue an I/O request, and then be placed in an I/O queue.
- The process could create a new sub process and wait for its termination.
- The process could be removed forcibly from the CPU, as a result of an interrupt, and be put back in the ready queue.

13. Write a note on different types of scheduler.

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

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

Long Term Scheduler

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

Short Term Scheduler

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

Medium Term Scheduler

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

be swapped out or rolled out. Swapping may be necessary to improve the process mix.

14. Write a note on Co-operating process.

The concurrent processes executing in the operating system may be either independent processes or cooperating processes. A process is independent if it cannot affect or be affected by the other processes executing in the system. Clearly, any process that does not share any data (temporary or persistent) with any other process is independent. On the other hand, a process is cooperating if it can affect or be affected by the other processes executing in the system.

We may want to provide an environment that allows process cooperation for several reasons:

- **Information sharing:** Since several users may be interested in the same piece of information. We must provide an environment to allow concurrent access to these types of resources.
- **Computation speedup:** If we want a particular task to run faster, we must break it into subtasks, each of which will be executing in parallel with the others. Such a speedup can be achieved only if the computer has multiple processing elements.
- **Modularity:** We may want to construct the system in a modular fashion, dividing the system functions into separate processes or threads.
- **Convenience:** Even an individual user may have many tasks on which to work at one time. For instance, a user may be editing, printing, and compiling in parallel.

15. Briefly explain thread concepts and the benefits of single and multi-threads.

A thread, sometimes called a lightweight process (LWP), is a basic unit of CPU utilization; A traditional (or heavyweight) process has a single thread of control. If the process has multiple threads of control, it can do more than one task at a time. A web browser might have one thread display images or text while another thread retrieves data from the network. A word processor may have a thread for displaying graphics, another thread for reading keystrokes from the user, and a third thread for performing spelling and grammar checking in the background.

Single and multiple threads benefits

- **Responsiveness:** Multithreading an interactive application may allow a program to continue running even if part of it is blocked or is performing a lengthy operation thereby increasing responsiveness to the user.
- **Resource sharing:** by default threads share the memory and the resources of the process to which they belong, The benefit of code sharing is that it allows an application to have several different of activities all within the same address space.
- **Economy:** Allocating memory and resources for process creation is costly alternatively, because threads share resources of the process to which they belong it is more economical to create and context switch threads.
- **Utilization of multiprocessor architectures:** the benefits of multithreading can be greatly increased in a multithreading architecture, where each thread may be running in parallel on a processor.

UNIT II

MULTIPLE CHOICE QUESTIONS **UNDERSTANDING**

1. Process execution cycle consist of which of the following
 - a. Only CPU Execution
 - b. Only I/O Wait
 - c. **Both CPU Execution and I/O Wait**
 - d. None of the above
2. Process execution starts with
 - a. **CPU burst**
 - b. Input burst
 - c. Output burst
 - d. I/O wait
3. Process execution ends with
 - a. **CPU burst**
 - b. Input burst
 - c. Output burst
 - d. I/O wait
4. Which Scheduling does this “Once a CPU is allocated to process, the process keeps the CPU until it releases the CPU either by terminating or by switching to waiting state”.
 - a. Preemptive
 - b. **Non- Preemptive**
 - c. Both The above
 - d. None of the above
5. Which Scheduling does this “Once a CPU is allocates to process, the process can be forcibly be taken away from the CPU can be allocated to the new process”.
 - a. **Preemptive**
 - b. Non- Preemptive
 - c. Both The above
 - d. None of the above
6. Which of the following gives control of CPU to process selected by short term scheduler
 - a. **Dispatcher**
 - b. Switching Context
 - c. User mode
 - d. None of the above
7. Which of the following are functions of dispatcher
 - a. Switching context
 - b. Switching to user mode
 - c. Jumping to the proper location in the user program to restart that program
 - d. **All the above**
8. The time taken by the dispatcher to stop one process and start another process is called as

- a. **Latency**
 - b. CPU Utilization
 - c. Throughput
 - d. None of the above
9. Which criteria says we want to keep the CPU as busy as possible.
- a. **CPU utilization**
 - b. Throughput
 - c. Turnaround time
 - d. Waiting time
10. The number of processes completed per time unit is called as
- a. CPU utilization
 - b. **Throughput**
 - c. Turnaround time
 - d. Waiting time
11. The interval from the time of submission of a process to the time of completion is
- a. CPU utilization
 - b. Throughput
 - c. **Turnaround time**
 - d. Waiting time
12. The amount of time that a process spends waiting in the ready queue is called
- a. CPU utilization
 - b. Throughput
 - c. Turnaround time
 - d. **Waiting time**
13. The time interval between submission of a request and the first response is called
- a. CPU utilization
 - b. Throughput
 - c. Turnaround time
 - d. **Response time**

APPLICATIONS

14. Give the Full form of FCFS
- a. **FIRST-COME, FIRST-SERVED**
 - b. FIRST-COME, FIRST-Scan
 - c. FIRST-Case, FIRST-SERVED
 - d. FIRST-Case, FIRST-Scan
15. Full form of SJF
- a. **Shortest Job First**
 - b. Simple Job First
 - c. Shortest Job Fine
 - d. Simple Job Fine
16. Equal Priority process are processed in which order
- a. **FCFS**

- b. SJF- preemptive
 - c. SJF Non-Preemptive
 - d. None of the above
17. Which Scheduling algorithm takes time slice
- a. FIRST-COME, FIRST-SERVED
 - b. Shortest Job First
 - c. Round Robin**
 - d. Multi-level Queue Scheduling
18. A small Unit of time is called
- a. Time gap
 - b. Time Slice**
 - c. Nano Time
 - d. None of the above
19. Which Scheduling algorithm allows the process to move between queue
- a. FIRST-COME, FIRST-SERVED
 - b. Shortest Job First
 - c. Round Robin
 - d. Multi-level feedback Queue Scheduling**
20. Which among the following partitions the ready queue into several separate Queue
- a. FIRST-COME, FIRST-SERVED
 - b. Shortest Job First
 - c. Round Robin
 - d. Multi-level Queue Scheduling**
21. Which is the technique used for gradually increasing the priority of process that waits for a long time
- a. Starvation
 - b. Aging**
 - c. Fragmentation
 - d. None of the above
22. What is the problem called when a process is ready to run but lacks CPU and is Blocked-waiting for the CPU
- a. Starvation**
 - b. Aging
 - c. Fragmentation
 - d. None of the above
23. Round Robin Scheduling algorithm is especially design for which system
- a. Batch Processing System
 - b. Multi Programming Systems
 - c. Time Sharing System**
 - d. Real Time System
24. Which of the following is correct

- a. **foreground or interactive**
- b. background or interactive
- c. foreground or batch
- d. None of the above

25. Which scheduling algorithm moves process to lower priority queue if it uses too much
- a. CPU time
 - b. FIRST-COME, FIRST-SERVED
 - c. Shortest Job First
 - d. **Multi-level feedback Queue Scheduling**

LONG QUESTIONS UNDERSTANDING

1. Give a brief outline about CPU Schedulers.

Whenever the CPU becomes idle, the operating system must select one of the processes in the ready queue to be executed. The selection process is carried out by the short-term scheduler (or CPU scheduler). The scheduler selects from among the processes in memory that are ready to execute, and allocates the CPU to one of them. The ready queue is not necessarily a first-in, first-out (FIFO) queue. A ready queue may be implemented as a FIFO queue, a priority queue, a tree, or simply an unordered linked list. Conceptually, however, all the processes in the ready queue are lined up waiting for a chance to run on the CPU. The records in the queues are generally process control blocks (PCBs) of the processes.

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

1. When a process switches from the running state to the waiting state (for I/O request or invocation of wait for the termination of one of the child processes).
2. When a process switches from the running state to the ready state (for example, when an interrupt occurs).
3. When a process switches from the waiting state to the ready state (for example, completion of I/O).
4. When a process terminates.

2. Briefly explain different scheduling criteria

Different CPU-scheduling algorithms have different properties and may favour one class of processes over another. In choosing which algorithm to use in a particular situation, we must consider the properties of the various algorithms. There are many different criteria's to check when considering the "best" scheduling algorithm, they are:

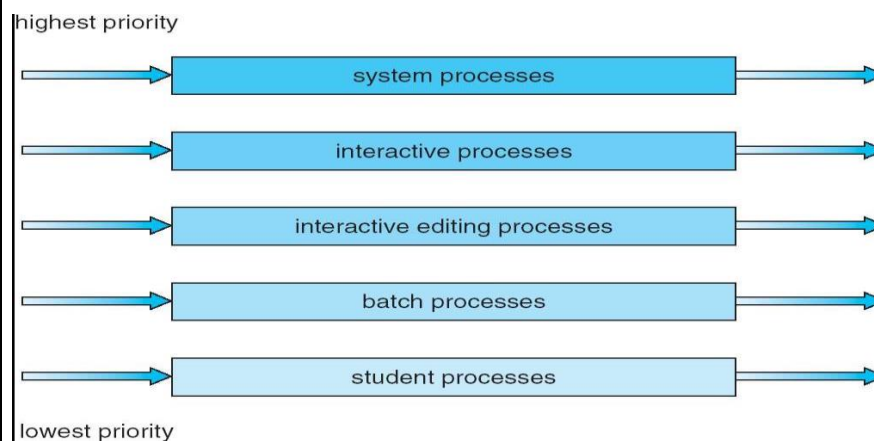
- **CPU Utilization** : To make out the best use of CPU and not to waste any CPU cycle, CPU would be working most of the time (Ideally 100% of the time). Considering a real system, CPU usage should range from 40% (lightly loaded) to 90% (heavily loaded.)
- **Throughput** : It is the total number of processes completed per unit time or rather say total amount of work done in a unit of time. This may range from 10/second to 1/hour depending on the specific processes.
- **Turnaround Time** : It is the amount of time taken to execute a particular process, i.e. The interval from time of submission of the process to the time of completion of the process (Wall clock time).
- **Waiting Time** : The sum of the periods spent waiting in the ready queue amount of time a process has been waiting in the ready queue to acquire get control on the CPU.
- **Load Average** : It is the average number of processes residing in the ready queue waiting for their turn to get into the CPU.

- **Response Time** : Amount of time it takes from when a request was submitted until the first response is produced. Remember, it is the time till the first response and not the completion of process execution (final response). In general CPU utilization and Throughput are maximized and other factors are reduced for proper optimization.

3. Explain multi-level Queue Scheduling.

A multilevel queue-scheduling algorithm partitions the ready queue into several separate queues. The processes are permanently assigned to one queue, generally based on some property of the process, such as memory size, process priority, or process type. Each queue has its own scheduling algorithm. For example, separate queues might be used for foreground and background processes. The foreground queue might be scheduled by an RR algorithm, while the background queue is scheduled by an FCFS algorithm.

In addition, there must be scheduling among the queues, which is commonly implemented as fixed-priority preemptive scheduling. For example, the foreground queue may have absolute priority over the background queue.



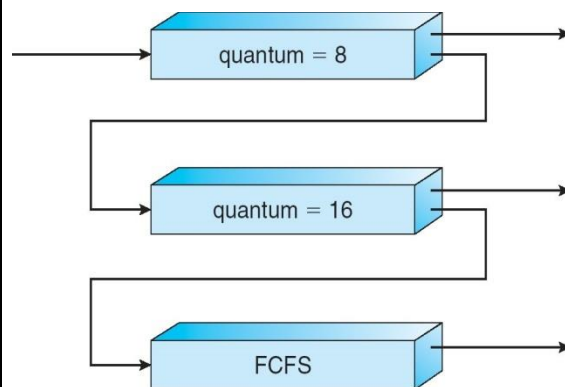
Each queue has absolute priority over lower-priority queues. No process in the batch queue, for example, could run unless the queues for system processes, interactive processes, and interactive editing processes were all empty. If an interactive editing process entered the ready queue while a batch process was running, the batch process would be pre-empted.

4. Explain multi-level feedback Queue Scheduling.

Multilevel feedback queue scheduling, however, allows a process to move between queues. The idea is to separate processes with different CPU-burst characteristics. If a process uses too much CPU time, it will be moved to a lower-priority queue. This scheme leaves I/O bound and interactive processes in the higher-priority queues. Similarly, a process that waits too long in a lower priority queue may be moved to a higher-priority queue. This form of aging prevents starvation.

For example, consider a multilevel feedback queue scheduler with three queues, numbered from 0 to 2. The scheduler first executes all processes in queue 0. Only when queue 0 is empty will it execute processes in queue 1. Similarly, processes in queue 2 will be executed only if queues 0 and 1 are empty. A process that arrives for queue 1 will preempt a process in queue 2. A process that arrives for queue 0 will, in turn, preempt a process in queue 1. A process entering the ready queue is put in queue 0. A process in queue 0 is given a time quantum of 8 milliseconds. If it does not finish within this time, it is moved to the tail of queue 1. If queue 0 is empty, the process at the head of queue 1 is given a quantum of 16 milliseconds. If it does not complete, it is pre-empted and is put into queue 2. Processes in queue 2 are run on an FCFS

basis, only when queues 0 and 1 are empty.



5. Explain the working of First-come First Served scheduling with example

By far the simplest CPU-scheduling algorithm is the first-come, first-served (FCFS) scheduling algorithm. With this scheme, the process that requests the CPU first is allocated the CPU first. The implementation of the FCFS policy is easily managed with a FIFO queue. When a process enters the ready queue, its PCB is linked onto the tail of the queue. When the CPU is free, it is allocated to the process at the head of the queue. The running process is then removed from the queue. The average waiting time under the FCFS policy, however, is often quite long.

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

Process	Burst Time
P1	24
P2	3
P3	3

If the processes arrive in the order P₁, P₂, P₃, and are served in FCFS order, we get the result shown in the following Gantt chart:

P1	P2	P3
0	24	27 30

The waiting time is 0 milliseconds for process P₁, 24 milliseconds for process P₂, and 27 milliseconds for process P₃. Thus, the average waiting time is $(0 + 24 + 27)/3 = 17$ milliseconds. The turn around time is 24 milliseconds for process P₁, 27 milliseconds for process P₂, and 30 milliseconds for process P₃. Thus, the average turn around time is $(24 + 27 + 30)/3 = 27$ milliseconds.

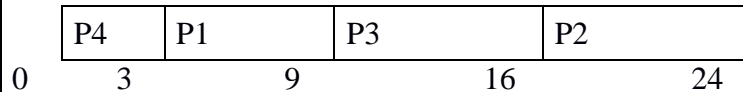
6. Explain the working of Shortest job first scheduling with example

A different approach to CPU scheduling is the shortest-job-first (SJF) scheduling algorithm. This algorithm associates with each process the length of the latter's next CPU burst. When the CPU is available, it is assigned to the process that has the smallest next CPU burst. If two processes have the same length next CPU burst, FCFS scheduling is used to break the tie. Note that a more appropriate term would be the shortest next CPU burst, because the scheduling is done by examining the length of the next CPU burst of a process, rather than its total length. As an example, consider the following set of processes, with the length of the burst time given

in milliseconds:

Process	Burst Time
P1	6
P2	8
P3	7
P4	3

Using SJF scheduling, we would schedule these processes according to the following Gantt chart:



The waiting time is 3 milliseconds for process P₁, 16 milliseconds for process P₂, 9 milliseconds for process P₃, and 0 milliseconds for process P₄. Thus, the average waiting time is $(3 + 16 + 9 + 0)/4 = 7$ milliseconds. If we were using the FCFS scheduling scheme, then the average waiting time could be 10.25 milliseconds.

APPLICATIONS

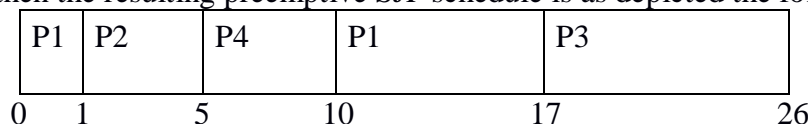
7. Explain the working of pre-emptive Shortest job first scheduling with example

The SJF algorithm may be either preemptive or non-pre-emptive. The choice arises when a new process arrives at the ready queue while a previous process is executing. The new process may have a shorter next CPU burst than what is left of the currently executing process. A preemptive SJF algorithm will preempt the currently executing process, whereas a non-preemptive SJF algorithm will allow the currently running process to finish its CPU burst. Preemptive SJF scheduling is sometimes called shortest-remaining-time-first scheduling.

As an example, consider the following four processes, with the length of the CPU-burst time given in milliseconds.

Process	Burst Time
P1	8
P2	4
P3	9
P4	5

If the processes arrive at the ready queue at the times shown and need the indicated burst times, then the resulting preemptive SJF schedule is as depicted the following Gantt chart:



Process P₁ is started at time 0, since it is the only process in the queue. Process P₂ arrives at time 1. The remaining time for process P₁ (7 milliseconds) is larger than the time required by process P₂ (4 milliseconds), so process is preempted, and process P₂ is scheduled. The average waiting time for this example is $((10 - 1) + (1 - 1) + (17 - 2) + (5 - 3))/4 = 26/4 = 6.5$ milliseconds.

8. Explain the working of non-pre-emptive Shortest job first scheduling with example

A non pre-emptive SJF scheduling would result in an average waiting time of 7.75

milliseconds.

Example: Non-Preemptive SJF (varied arrival times)

Process	Arrival Time	Burst Time
P1	2	6
P2	1	8
P3	0	7
P4	3	3

P3	P4	P1	P2	
0	7	10	16	24

Average waiting time = $((10-2)+(16-1)+(0-0)+(7-3))/4 = (8 + 15 + 0 + 4)/4 = 6.75\text{ms}$

9. Explain the working of Priority Scheduling with example

A priority is associated with each process, and the CPU is allocated to the process with the highest priority. Equal-priority processes are scheduled in FCFS order. An SJF algorithm is simply a priority algorithm where the priority (p) is the inverse of the (predicted) next CPU burst. The larger the CPU burst, the lower the priority, and vice versa.

As an example, consider the following set of processes, assumed to have arrived at time 0, in the order P₁, P₂.....P₅ with the length of the CPU-burst time given in milliseconds:

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Using priority scheduling, we would schedule these processes according to the following Gantt chart:

P2	P5	P1	P3	P4	
0	1	6	16	18	19

Average waiting time is 8.2 milliseconds (i.e. $(6+0+16+18+1)/5$).

10. Explain the working of Round Robin scheduling with example

The Round-Robin (RR) scheduling algorithm is designed especially for time sharing systems. It is similar to FCFS scheduling, but preemption is added to switch between processes. A small unit of time, called a time quantum (or time slice), is defined. A time quantum is generally from 10 to 100 milliseconds. The ready queue is treated as a circular queue. The CPU scheduler goes around the ready queue, allocating the CPU to each process for a time interval of up to 1 time quantum. To implement RR scheduling, we keep the ready queue as a FIFO queue of processes. New processes are added to the tail of the ready queue. The CPU scheduler picks the first process from the ready queue, sets a timer to interrupt after 1 time quantum, and dispatches the process.

The average waiting time under the RR policy, however, is often quite long. Consider the following set of processes that arrive at time 0, with the length of the CPU-burst time given in milliseconds:

Process	Burst Time
P1	24
P2	3
P3	3

If we use a time quantum of 4 milliseconds, then process P1 gets the first 4 milliseconds. Since it requires another 20 milliseconds, it is preempted after the first time quantum, and the CPU is given to the next process in the queue, process P2. Since process P2 does not need 4 milliseconds, it quits before its time quantum expires. The CPU is then given to the next process, process P3. Once each process has received 1 time quantum, the CPU is returned to process P1 for an additional time quantum. The resulting RR schedule is

P1	P2	P3	P1	P1	P1	P1	P1	
0	4	7	10	14	18	22	26	30

The average waiting time is $17/3 = 5.66$ milliseconds.

11. Consider the following set of process, with the length of CPU-burst time given in milliseconds

Process	Burst Time
P1	24
P2	3
P3	3

Draw Gantt chart and find average waiting time using FCFS scheduling.

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

Process	Burst Time
P1	24
P2	3
P3	3

If the processes arrive in the order P1, P2, P3, and are served in FCFS order, we get the result shown in the following Gantt chart:

P1	P2	P3	
0	24	27	30

The waiting time is 0 milliseconds for process P1, 24 milliseconds for process P2, and 27 milliseconds for process P3. Thus, the average waiting time is $(0 + 24 + 27)/3 = 17$ milliseconds. The turn around time is 24 milliseconds for process P1, 27 milliseconds for process P2, and 30 milliseconds for process P3. Thus, the average turn around time is $(24 + 27 + 30)/3 = 27$ milliseconds.

12. Consider the following set of process, with the length of CPU-burst time given in milliseconds

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

Draw Gantt chart and find average waiting time using SJF Non-Pre-emptive scheduling.

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

P1	P2	P4	P3
----	----	----	----

0 8 12 17 26

Average waiting time = $((0-0)+(8-1)+(17-2)+(12-3))/4 = (0+7+15+9)/4 = 7.75\text{ms}$

13. Consider the following set of process, with the length of CPU-burst time given in milliseconds

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

Draw Gantt chart and find average waiting time using SJF Pre-emptive scheduling.

consider the following four processes, with the length of the CPU-burst time given in milliseconds.

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

If the processes arrive at the ready queue at the times shown and need the indicated burst times, then the resulting preemptive SJF schedule is as depicted the following Gantt chart:

P1	P2	P4	P1	P3	
0	1	5	10	17	26

Process P₁ is started at time 0, since it is the only process in the queue. Process P₂ arrives at time 1. The remaining time for process P₁ (7 milliseconds) is larger than the time required by process P₂ (4 milliseconds), so process P₁ is preempted, and process P₂ is scheduled. The average waiting time for this example is $((10 - 0) + (1 - 1) + (17 - 2) + (5 - 3))/4 = 26/4 = 6.5$ milliseconds.

14. Consider the following set of process, with the length of CPU-burst time given in milliseconds. Quantum Time is 4ms

Process	Burst Time
P1	24
P2	3
P3	3

Draw Gantt chart and find average waiting time using Round Robin scheduling.

The average waiting time under the RR policy, however, is often quite long. Consider the following set of processes that arrive at time 0, with the length of the CPU-burst time given in milliseconds:

Process	Burst Time
P1	24
P2	3
P3	3

P1	P2	P3	P1	P1	P1	P1	P1	
0	4	7	10	14	18	22	26	30

The average waiting time is $17/3 = 5.66$ milliseconds.

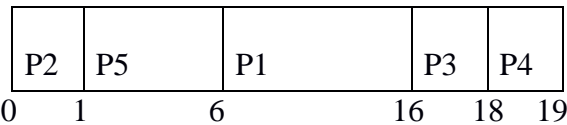
15. Consider the following set of process, with the length of CPU-burst time given in milliseconds

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Draw Gantt chart and find average waiting time using Priority scheduling.

Process	Burst Time	Priority
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Using priority scheduling, we would schedule these processes according to the following Gantt chart:



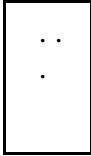
Average waiting time is 8.2 milliseconds (i.e. $(6+0+16+18+1)/5$).

UNIT III

MULTIPLE CHOICE QUESTIONS

UNDERSTANDING

1. Which among the following is not physical resource
 - a. Printer
 - b. Tape drive
 - c. Files**
 - d. None of the above
2. Which among the following are necessary and sufficient condition for deadlock state
 - a. Mutual exclusion
 - b. Hold and wait
 - c. No preemption
 - d. All the above**
3. Which necessary any sufficient condition for a deadlock states that “At least one resource must be held in a non-sharable model”.
 - a. Mutual exclusion**
 - b. Hold and wait
 - c. No preemption
 - d. Circular wait
4. Which necessary any sufficient condition for a deadlock state says that “A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes”
 - a. Mutual exclusion
 - b. Hold and wait**
 - c. No preemption
 - d. Circular wait
5. Which necessary any sufficient condition for a deadlock state says that “Resources cannot be preempted”
 - a. Mutual exclusion
 - b. Hold and wait
 - c. No preemption**
 - d. Circular wait
6. In resource allocation graph how do we represent a process
 - a. Circle**
 - b. Square
 - c. Rectangle
 - d. Arrow
7. In resource allocation graph how do we represent a Resource type
 - a. Circle
 - b. Square**
 - c. Rectangle

- d. Arrow
8. In resource allocation graph how do we represent more than 1 instance of resource type
- Triangle within a square
 - Dot within a square**
 - Star within a square
 - Arrow
9. A directed edge from Process P_i to resource $R_j(P_i \rightarrow R_j)$ is called as
- Request edge**
 - Assignment edge
 - Process edge
 - Resource edge
10. A directed edge from Resource R_j to Process $P_i(R_j \rightarrow P_i)$ is called as
- Request edge
 - Assignment edge**
 - Process edge
 - Resource edge
11. Give the number of instance present here
- 1
 - 2
 - 3**
 - 0
- 
12. How is claimed edge represented
- Dashed lines**
 - Solid lines
 - Double lines
 - None of the above
13. Full form of MMU
- Memory Management Unit**
 - Main Management Unit
 - Main Memory Unit
 - None of the above

APPLICATION

14. Which among the following allocates the hole that is large enough
- First fit**
 - Best fit
 - Worst fit
 - None of the above
15. Which among the following allocates the hole that is Smallest hole that is big enough
- First fit
 - Best fit**

- c. Worst fit
 - d. None of the above
16. Which among the following allocates the largest hole
- a. First fit
 - b. Best fit
 - c. Worst fit**
 - d. None of the above
17. The size of the page is typically a power of
- a. 2**
 - b. 3
 - c. 4
 - d. 6
18. Full form of PTBR
- a. Page table base registers**
 - b. Page time base registers
 - c. Page table base reference
 - d. Page time base reference
19. Full form of TLB
- a. Translation look aside buffer**
 - b. Transfer look aside buffer
 - c. Translation late aside buffer
 - d. Transfer late aside buffer
20. The percentage of times particular page is found in TLB is called
- a. Page ratio
 - b. Hit ratio**
 - c. Percentage
 - d. None of the above
21. Which among the following process must be selected for the termination
- a. minimum cost process**
 - b. maximum cost process
 - c. Any of the process
 - d. First process
22. The Base register is called as
- a. Limit Register
 - b. Relocation registers**
 - c. Register
 - d. None of the above
23. Physical memory is divided into fixed sized blocks called
- a. Pages
 - b. Frames**
 - c. Blocks
 - d. Sections

24. Logical memory is divided into blocks of same size called

- a. **Pages**
- b. Frames
- c. Blocks
- d. Sections

25. Paging causes

- a. **Increase in context switch time**
- b. Decrease in context switch time
- c. Context time doesn't alter
- d. Context time is zero

LONG QUESTIONS

UNDERSTANDING

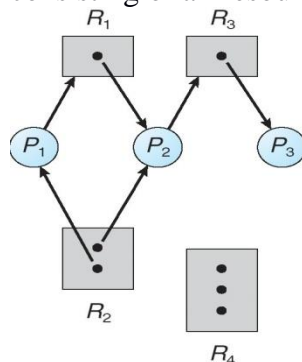
1. **Summarize different deadlock characterization.**

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. A deadlock situation can arise if the following four conditions hold simultaneously in a system:

- **Mutual exclusion:** At least one resource must be held in a non sharable model that is, only one process at a time can use the resource. If another process requests that resource, the requesting process must be delayed until the resource has been released.
- **Hold and wait:** A process must be holding at least one resource and waiting to acquire additional resources that are currently being held by other processes.
- **No preemption:** Resources cannot be preempted; that is, the process holding it, after that process has completed its task can release a resource only voluntarily.
- **Circular wait:** A set $\{P_0, P_1, \dots, P_n\}$ of waiting processes must exist such that P_0 is waiting for a resource that is held by P_1 , P_1 is waiting for a resource that is held by P_2 , ..., P_{n-1} is waiting for a resource that is held by P_n , and P_n is waiting for a resource that is held by P_0 .

2. **With neat diagram explain resource allocation graph.**

Deadlocks can be described more precisely in terms of a directed graph called a system resource-allocation graph. This graph consists of a set of vertices V and a set of edges E . The set of vertices V is partitioned into two different types of nodes $P = \{P_1, P_2, \dots, P_n\}$, the set consisting of the entire active processes in the system, and $R = \{R_1, R_2, \dots, R_m\}$, the set consisting of all resource types in the system.



Process states:

- Process P_1 is holding an instance of resource type R_2 , and is waiting for an instance of resource type R_1 .
- Process P_2 is holding an instance of R_1 and R_2 , and is waiting for an instance of resource type R_3 .
- Process P_3 is holding an instance of R_3

If the graph contains no cycles, then no process in the system is deadlocked. If the graph does contain a cycle, then a deadlock may exist.

$P_1 \rightarrow R_1 \rightarrow P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_1$ $P_2 \rightarrow R_3 \rightarrow P_3 \rightarrow R_2 \rightarrow P_2$

3. Discuss different conditions which can cause deadlock.

- **Mutual Exclusion** :The mutual-exclusion condition must hold for non sharable resources. For example, a printer cannot be simultaneously shared by several processes. Sharable resources, on the other hand, do not require mutually exclusive access, and thus cannot be involved in a deadlock. Read-only files are a good example of a sharable resource. we cannot prevent deadlocks by denying the mutual-exclusion condition: Some resources are intrinsically non sharable.
- **Hold and Wait** : The hold-and-wait condition can be prevented by requiring that a process request all of its required resources at one time and blocking the process until all requests can be granted simultaneously. This approach is inefficient in two ways. First, a process may be held up for a long time waiting for all of its resource requests to be filled. Another problem is that a process may not know in advance all of the resources that it will require.
- **No Preemption** :This condition can be prevented in several ways. First, if a process holding certain resources is denied a further request, that process must release its original resources and, if necessary, request them again together with the additional resource. Alternatively, if a process requests a resource that is currently held by another process, the OS may preempt the second process and require it to release its resources.
- **Circular Wait** :The circular-wait condition can be prevented by defining a linear ordering of resource types. If a process has been allocated resources of type R , then it may subsequently request only those resources of types following R in the ordering. Now suppose that two processes, A and B, are deadlocked because A has acquired R_i and requested R_j , and B has acquired R_j and requested R_i . This condition is impossible because it implies $i < j$ and $j < i$.

4. Illustrate the concept of memory allocation and compare different strategies.

One simple approach is to divide the memory into fixed sized partitions. Each partition may contain exactly one process. The degree of multiprogramming is bounded by the number of partitions. In multiple partition method, when a partition is free, a process is selected from the input queue and is loaded into the free partition. When the process terminates, the partition becomes free for another process. The operating system keeps a table indicating which parts of the memory are available and which are occupied. Initially all memory is available for user processes and is considered as one large block of available memory - a hole. When a process arrives and needs memory, we search for a hole that is large enough for the process. If we find one, we allocate only as much memory as needed, keeping the rest to satisfy future requirements. As the processes enter the system, they are put in an input queue.

When a process arrives and needs memory, the system searches this set for a hole that is large enough for this process. If the hole is too large, it is split into two: one part is allocated to the arriving process and the other is returned to the set of holes. If a new hole is adjacent to other holes these adjacent holes are merged to form a larger hole. At this point of time, the system needs to check whether there are processes waiting for memory and whether their

requirements can be satisfied. This is an instance of dynamic storage allocation problem - how to satisfy a request of size n from a list of free holes. The set of holes is searched to determine which of the hole is best to allocate. The most common strategies are

- **First fit** - Allocate the hole that is large enough. Searching can start at the beginning of the set of holes or where the previous first-fit search ended. We can stop searching once we find a free hole that is large enough.
- **Best fit** - Allocate the smallest hole that is big enough. We must search the entire list unless the list is kept ordered by size. This strategy produces smallest leftover holes.
- **Worst fit** - Allocate the largest hole. We must search the entire list. This strategy produces the largest leftover holes which may be more useful than the smaller leftover holes from the best fit approach.

5. Discuss different methods for handling deadlock.

Principally, we can deal with the deadlock problem in one of three ways:

- We can use a protocol to prevent or avoid deadlocks, ensuring that the system will never enter a deadlock state.
- We can allow the system to enter a deadlock state, detect it, and recover.
- We can ignore the problem altogether, and pretend that deadlocks never occur in the system. This solution is used by most operating systems, including UNIX.

To ensure that deadlocks never occur, the system can use either deadlock prevention or a deadlock-avoidance scheme. Deadlock prevention is a set of methods for ensuring that at least one of the necessary conditions cannot hold. These methods prevent deadlocks by constraining how requests for resources can be made.

Deadlock avoidance, on the other hand, requires that the operating system be given in advance additional information concerning which resources a process will request and use during its lifetime. With this additional knowledge, we can decide for each request whether or not the process should wait. To decide whether the current request can be satisfied or must be delayed, the system must consider the resources currently available, the resources currently allocated to each process, and the future requests and releases of each process.

If a system does not employ either a deadlock-prevention or a deadlock avoidance algorithm, and then a deadlock situation may occur. In this environment, the system can provide an algorithm that examines the state of the system to determine whether a deadlock has occurred, and an algorithm to recover from the deadlock. If a system does not ensure that a deadlock will never occur, and also does not provide a mechanism for deadlock detection and recovery, then we may arrive at a situation where the system is in a deadlock state yet has no way of recognizing what has happened.

6. Write a note on following

a. Mutual exclusion

The mutual-exclusion condition must hold for non sharable resources. For example, a printer cannot be simultaneously shared by several processes. Sharable resources, on the other hand, do not require mutually exclusive access, and thus cannot be involved in a deadlock. Read-only files are a good example of a sharable resource. If several processes attempt to open a read-only file at the same time, they can be granted simultaneous access to the file. A process never needs to wait for a sharable resource. In general, however, we cannot prevent deadlocks by denying the mutual-exclusion condition: Some resources are intrinsically non sharable.

b. Hold and Wait

The hold-and-wait condition can be prevented by requiring that a process request all of its required resources at one time and blocking the process until all requests can be granted

simultaneously. This approach is inefficient in two ways. First, a process may be held up for a long time waiting for all of its resource requests to be filled, when in fact it could have proceeded with only some of the resources. Second, resources allocated to a process may remain unused for a considerable period, during which time they are denied to other processes. Another problem is that a process may not know in advance all of the resources that it will require. There is also the practical problem created by the use of modular programming or a multithreaded structure for an application. An application would need to be aware of all resources that will be requested at all levels or in all modules to make the simultaneous request.

7. Explain deadlock detection for Single Instance of each Resource Type.

If all resources have only a single instance, then we can define a deadlock detection algorithm that uses a variant of the resource-allocation graph, called a wait for graph. We obtain this graph from the resource-allocation graph by removing the nodes of type resource and collapsing the appropriate edges.

More precisely, an edge from P_i to P_j in a wait-for graph implies that process P_i is waiting for process P_j to release a resource that P_i needs. An edge $P_i \rightarrow P_j$ exists in a wait-for graph if and only if the corresponding resource allocation graph contains two edges $P_i \rightarrow R_q$ and $R_q \rightarrow P_j$ for some resource R_q . We present a resource-allocation graph and the corresponding wait-for graph as follows.

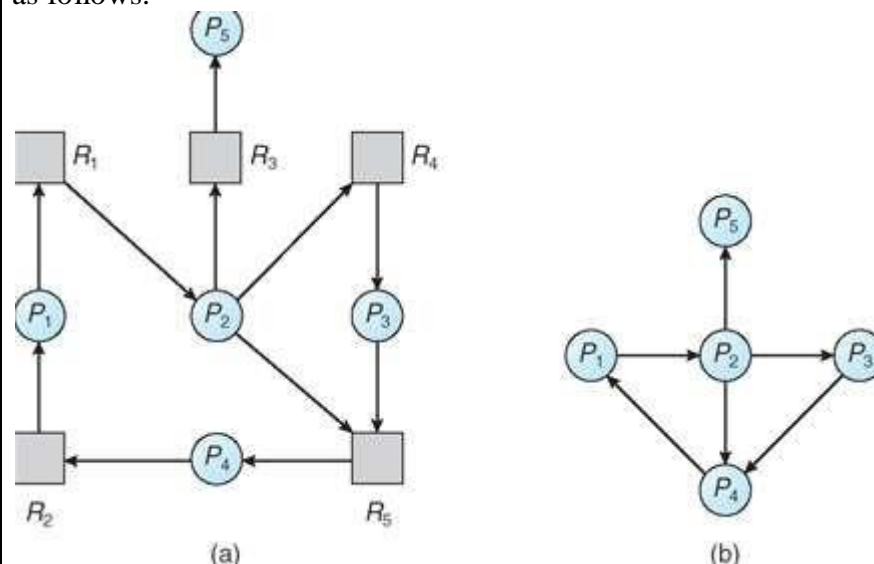


Figure: (a) Resource-allocation graph. (b) Corresponding wait-for graph.

A deadlock exists in the system if and only if the wait-for graph contains a cycle. To detect deadlocks, the system needs to maintain the wait-for graph and periodically to invoke an algorithm that searches for a cycle in the graph. An algorithm to detect a cycle in a graph requires an order of n^2 operations, where n is the number of vertices in the graph.

APPLICATION

8. Illustrate deadlock system model.

A system consists of a finite number of resources to be distributed among a number of competing processes. The resources are partitioned into several types, each of which consists of some number of identical instances. Memory space, CPU cycles, files, and I/O devices (such as printers and tape drives) are examples of resource types. If a system has two CPUs, then the resource type CPU has two instances. If a process requests an instance of a resource type, the allocation of an instance of the type will satisfy the request.

A process must request a resource before using it, and must release the resource after using it. A process may request as many resources as it requires carrying out its designated

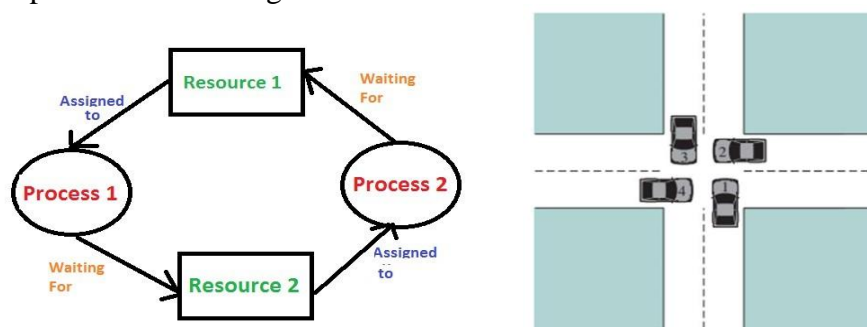
task. Obviously, the number of resources requested may not exceed the total number of resources available in the system. In other words, a process cannot request three printers if the system has only two. Under the normal mode of operation, a process may utilize a resource in only the following sequence:

- **Request:** If the request cannot be granted immediately (for example, the resource is being used by another process), then the requesting process must wait until it can acquire the resource.
- **Use:** The process can operate on the resource (for example, if the resource is a printer, the process can print on the printer).
- **Release:** The process releases the resource.

9. Demonstrate different deadlock process state.

Deadlock is a situation where a set of processes are blocked because each process is holding a resource and waiting for another resource acquired by some other process. Consider an example when two trains are coming toward each other on same track and there is only one track, none of the trains can move once they are in front of each other. Similar situation occurs in operating systems when there are two or more processes hold some resources and wait for resources held by other(s).

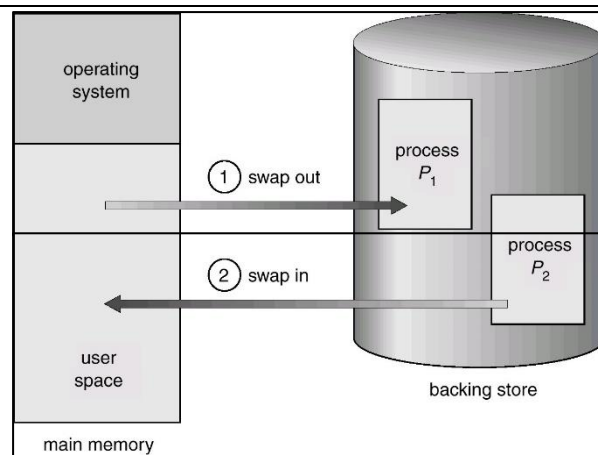
Process 1 is holding Resource 1 and waiting for resource 2 which is acquired by process 2, and process 2 is waiting for resource 1.



10. Demonstrate swapping concept.

A process needs to be in memory to be executed. A process can be swapped temporarily out of memory to a backing store and then brought back to memory for continued execution. For example: assume a multi programmed environment with round robin CPU scheduling. When a quantum expires the memory manager will start to swap out the process that just finished and swap in another process to the memory space that has just been freed. In the mean time the CPU will allocate time slice to some other process in the memory. A variant of this policy is used in priority based scheduling algorithm. If a higher priority process arrives and wants service, the memory manager can swap out the lower priority process so that it can load and execute high priority process. When a higher priority process finishes, the lower priority process can be swapped in and continued. This variant of swapping is called roll-out roll-in.

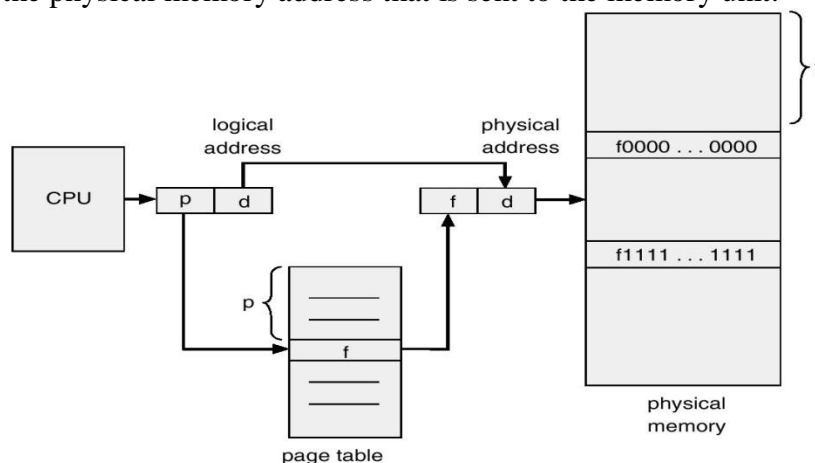
Swapping requires a backing store. Backing store is usually a fast disk. It must be large enough to accommodate copies of all memory images for all users. It must provide direct access to these memory images. The system maintains a ready queue consisting of all processes whose memory images are on the backing store or in memory and are ready to run. Whenever the CPU scheduler decides to execute a process, it calls the dispatcher. The dispatcher checks to see whether the next process in the queue is in memory. If there is no free memory region the dispatcher swaps out a process currently in memory and swaps in a desired process. It then reloads the registers as normal and transfers control to the selected process.



11. Give a brief outline about paging concept.

Paging is a memory management scheme that permits the physical address space of a process to be non-contiguous. Paging avoids problems of fitting varying-sized memory chunks on to the backing store.

Physical memory is divided into fixed-sized blocks called frames. Logical memory is broken into blocks of same size called pages. When a process is to be executed, its pages are loaded into any available memory frames from the backing store. The backing store is divided into fixed sized blocks that are of the same size as memory frames. Every address that is generated by the CPU is divided into two parts: a page number (P) and a page offset (d). The page number is used as an index to the page table. The page table contains the base address of each page in the physical memory. This base address is combined with the page offset to define the physical memory address that is sent to the memory unit.



12. write a note on fragmentation and segmentation.

Memory fragmentation could be internal as well as external. . As processes are loaded and removed from memory, the free memory space is broken into little pieces. External fragmentation exists when enough total memory space exists to satisfy a request but it is not contiguous. Storage is fragmented into a large number of small holes. Physical memory is broken into fixed sized blocks and allocated memory in unit of block sizes. In this approach, memory allocated to a process will be slightly larger than the requested memory. The difference in memory space between the allocated and requested memory is called internal fragmentation - memory that is internal to a partition but not being used. One solution to the problem of external fragmentation is compaction. The goal here is to shuffle the memory contents to place all the free memory together in one large block. Compaction is possible only if the relocation

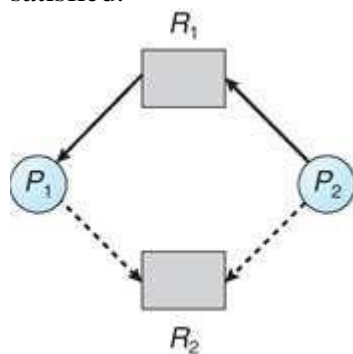
is dynamic. The simplest compaction algorithm is simply to move all processes to one end of memory and all holes in the other direction, producing one large hole of available memory.

Segmentation is one of the most common ways to achieve memory protection. In a computer system using segmentation, an instruction operand that refers to a memory location includes a value that identifies a segment and an offset within that segment. A segment has a set of permissions, and a length, associated with it. If the currently running process is allowed by the permissions to make the type of reference to memory that it is attempting to make, and the offset within the segment is within the range specified by the length of the segment, the reference is permitted; otherwise, a hardware exception is raised. Segmentation is a memory-management scheme that supports this programmer view of memory. A logical address space is a collection of segments. Each segment has a name and a length. The addresses specify both the segment name and the offset within the segment. The programmer therefore specifies each address by two quantities: a segment name and an offset.

13. Show how resource allocation graph helps in deadlock avoidance.

If we have a resource-allocation system with only one instance of each resource type, a variant of the resource-allocation graph defined can be used for deadlock avoidance. In addition to the request and assignment edges, we introduce a new type of edge, called a claim edge. A claim edge $P_i \rightarrow R_j$ indicates that process P_i may request resource R_j at some time in the future. This edge resembles a request edge in direction, but is represented by a dashed line. When process P_i requests resource R_j , the claim edge $P_i \rightarrow R_j$ is converted to a request edge. Similarly, when a resource R_j is released by P_i , the assignment edge $R_j \rightarrow P_i$ is reconverted to a claim edge $P_i \rightarrow R_j$. We note that the resources must be claimed a priori in the system. That is, before process P_i starts executing, all its claim edges must already appear in the resource-allocation graph. We can relax this condition by allowing a claim edge $P_i \rightarrow R_j$ to be added to the graph only if all the edges associated with process P_i are claim edges.

Suppose that process P_i requests resource R_j . The request can be granted only if converting the request edge $P_i \rightarrow R_j$ to an assignment edge $R_j \rightarrow P_i$ does not result in the formation of a cycle in the resource-allocation graph. Note that we check for safety by using a cycle-detection algorithm. An algorithm for detecting a cycle in this graph requires an order of n^2 operations, where n is the number of process in the system. If no cycle exists, then the allocation of the resource will leave the system in a safe state. If a cycle is found; then the allocation will put the system in an unsafe state. Therefore, process P_i will have to wait for its requests to be satisfied.



UNIT IV

MULTIPLE CHOICE QUESTIONS

UNDERSTANDING

- 1 Which is the technique that allows execution of processes that may not be completely in memory
 - a. **Virtual Memory**
 - b. Default Memory
 - c. Physical Memory
 - d. None of the above
- 2 When Operating system cannot bring the desired page to memory what is it called as
 - a. **Trap**
 - b. Failure
 - c. Fault
 - d. Error
- 3 Full Form of FIFO
 - a. **First in First out**
 - b. Far in First out
 - c. First in Far out
 - d. None of the above
- 4 Full form of LRU
 - a. **Least Recently Used**
 - b. Least Recent Unit
 - c. Least Required Unit
 - d. Less Required Unit
- 5 In this Frame allocation equation " $a_i = (s_i/S) * m$ " What does "m" mean
 - a. **Total number of available frames**
 - b. Size of particular process
 - c. Sum of size of all processes
 - d. None of the above
- 6 What is spending time in paging more than execution called as
 - a. Paging
 - b. **Trashing**
 - c. Moving
 - d. Executing
- 7 Repositioning within a file is also called as
 - a. **File seek**
 - b. Truncating a file
 - c. Reading a file
 - d. Writing a file
- 8 Which file attribute is in non-human readable format

- a. Name
- b. Identifier**
- c. Type
- d. Location

9 Which among the following extension belongs to Text file type

- a. txt**
- b. exe
- c. c
- d. mpeg

10 Which among the following extension belongs to Executable file type

- a. Txt
- b. exe**
- c. c
- d. mpeg

11 Which among the following extension belongs to Source code file type

- a. txt
- b. exe
- c. c**
- d. mpeg

12 Which among the following extension belongs to Multimedia file type

- a. txt
- b. exe
- c. c
- d. mpeg**

SKILL

13 Which among the following extension belongs to Multimedia file type

- a. txt
- b. exe
- c. c
- d. mov**

14 File and its extensions are separated by

- a. .**
- b. ,
- c. :
- d. ;

15 Which operation of directories allows us to access every directory and also each and every file within the directory

- a. Search for a file
- b. List a directory
- c. Create a file
- d. Traverse a file system**

- 16 Full form of UFD
- a. **User File Directory**
 - b. User Find Directory
 - c. Unique File Directory
 - d. None of the above
- 17 Full Form of MFD
- a. **Master File Directory**
 - b. Master Function Directory
 - c. Main Function Directory
 - d. Must Find Directory
- 18 Which type of path defines a path from current directory
- a. Absolute path name
 - b. **Relative path name**
 - c. Current path
 - d. Present path
- 19 Which type of path defines a path from root directory
- a. **Absolute path name**
 - b. Relative path name
 - c. Current path
 - d. Present path
- 20 Full form of FCB
- a. **File Control Block**
 - b. Function Control Block
 - c. File Control Base
 - d. Function Control Base
- 21 Full form of FAT
- a. **File Allocation Table**
 - b. Function Allocation Table
 - c. File Available Table
 - d. Function Available Table
- 22 Full Form of ACL
- a. **Access Control List**
 - b. Allocation Control List
 - c. Allocation Control Line
 - d. Access Control Line
- 23 The User who created the file is
- a. **Owner**
 - b. Group
 - c. Universe
 - d. None of the Above

24 The set of User who share the file and need similar access is

- a. Owner
- b. **Group**
- c. Universe
- d. None of the Above

25 How are the unused blocks indicated

- a. **0**
- b. 1
- c. 2
- d. 3

LONG QUESTIONS UNDERSTANDING

1. Discuss various attributes of files.

Every file has a name by which it is referred. The attributes of a file include

- **Name** – symbolic file name is the only information that is in human readable form.
- **Identifier** – This is a unique tag, usually a number. It identifies the file within the file system. It is a non-human readable name of the file.
- **Type** – this information is needed for those systems that support different types of files
- **Location** – this is a pointer to a device and to the location of the file on that device.
- **Size** – the current size of the file
- **Protection** – access control information that determines who can read, write or execute the file
- **Time, date and user identification** – this information may be kept for creation, last modification and last use. This data can be useful for protection, security and usage monitoring

2. Write a note on different access methods.

1. Sequential Access:

It is the simplest access method. Information in a file is processed one after the other. Compilers and editors use this mode of access. The operation that can be performed are read/write. A read operation reads the next portion of the file and automatically advances the file pointer. A write operation appends to the end of the file and advances to the new end of the file. Such files can be reset to the beginning of the file and also skip forward/backward n records. It works well on sequential access and random access devices.

2. Direct Access:

This method is also known as relative access. A file is made up of fixed length logical records that allow a program to read/write records rapidly in no particular order. In this mode, a file is viewed as a numbered sequence of blocks or records. A direct access file allows arbitrary blocks to be read or write. Databases are of this type. The file operations must be modified to include block number as a parameter. Thus, the operations on this file are

- o Read n for read next
- o Write n for write next , where n is the block number

The block number provided by the user to the OS is a relative block number. A relative block number is an index relative to the beginning of the file. Thus, relative block of the file zero even if the absolute disk address begins at say 4152

3. Other Access Methods:

Other access methods can be built on top of direct access methods. These methods involve construction of index for the file. The index contains pointers to various blocks. To search a record in a file, we first find the index and then use the pointer to access the file directly and find the required record.

3. Explain directory structure and different operations on Directories.

Computer systems store large volumes of data. To manage these data, we need to organize them. This organization is done in two parts

1. Disks are split into one or more partitions called minidisks or volumes.

Each disk on the system contains at least one partition, which is a low level structure in which files and directories reside. Partitions are used to provide separate areas within one disk, each treated as a separate storage device. The user needs to be concerned only about the logical directory and file structure

2. Each partition contains information about the files in it.

Each partition contains information about files within it. This information is kept in entries in a device directory or volume table of contents. The directory records all information like name, size, location, type for all files on that partition.

Operations on Directories:

- **Search for a file** – We should be able to search a directory to find entry for a particular file. In addition, we must also be able to find all files whose name match a particular pattern
- **Create a file** – new files should be created and added to the directory
- **Delete a file** – we should be able to remove a file from a directory
- **List a directory** – we should be able to list all the contents of a directory
- **Rename a file** – allow to change the name of an existing file
- **Traverse a file system** – we should be able to access every directory and also each and every file within the directory

4. Explain various access control for protection of file system.

The most common approach to protection problem is to make access dependant on the identity of the user. Various users may have different types of access to files or directories. Identity dependent access is implemented by associating each file/directory with Access Control List (ACL) specifying the user name and the types of access allowed for each user. When a user requests access to a particular file, the OS checks the access list associated with that file. If that user is listed for that requested access then, the access is allowed. Otherwise, protection violation occurs and the user job is denied access to the file.

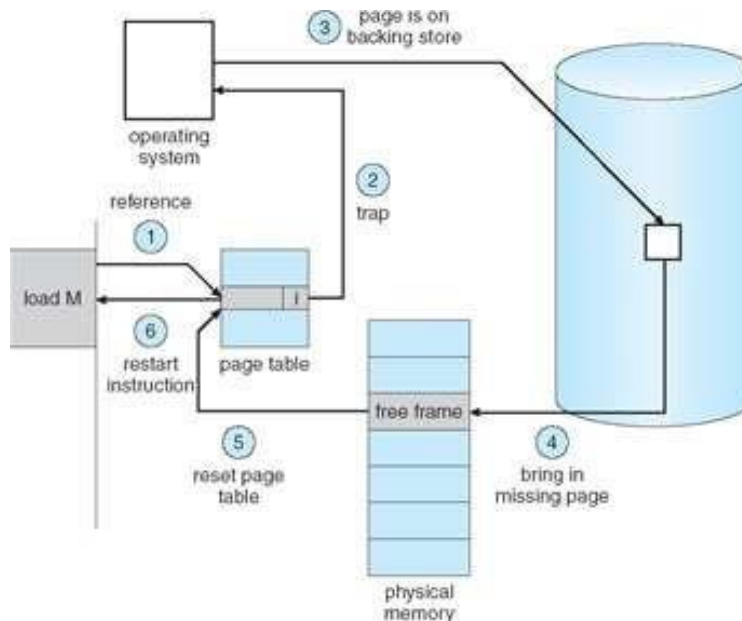
The main problem of this approach is length of access lists. To condense the length of ACL, systems have recognized three classifications of users in connection with each file.

- **Owner** – the user who created the file
- **Group** – a set of users who share the file and need similar access
- **Universe** – all other users of the system

One mechanism is to associate a file with a password. Passwords must be chosen randomly and changed often. This scheme is effective in limiting the access to a file only to those users who know the password.

5. Discuss the concept of Demand paging.

A demand-paging system is similar to a paging system with swapping (Above Figure) where processes reside in secondary memory (usually a disk). When we want to execute a process, we swap it into memory. Rather than swapping the entire process into memory, though, we use a lazy swapper. A lazy swapper never swaps a page into memory unless that page will be needed. In the context of a demand-paging system, use of the term “swapper” is technically incorrect. A swapper manipulates entire processes, whereas a pager is concerned with the individual pages of a process. We thus use “pager,” rather than “swapper,” in connection with demand paging.



The procedure to handle the trap is as follows.

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

6. Explain approaches taken for Page Replacement.

Page replacement takes the following approach - if no frame is free we find the one that is not currently being used and free it. We can free a frame by writing its contents to a swap space and changing the page table to indicate that the page is no longer in memory. Now the freed frame can be used to hold a page for which the process faulted. The page fault service routine is modified as follows:

1. Find a location of desired page on the disk.
2. Find a free frame
 - a. If there is a free frame use it.

- b. If there is no free frame, use a page replacement algorithm to select a victim frame.
- c. Write the victim page to the disk; change the page and frame tables accordingly.
3. Read the desired page into the new free frame. Change the page and frame tables.
4. Restart the user process.

Each page/frame may have a modify bit associated with it in the hardware. The modify bit in the page is set whenever a word/byte is written into it indicating the page has been modified. When a page is selected for replacement, the modify bit is examined. If it is set, we know that the page has been modified since it was read from the disk. In this case the page must be written to the disk. If the bit is not set, the page has not been modified since it was read into the memory.

7. Explain various file operations.

A file is an abstract data type. The basic operations on files are:

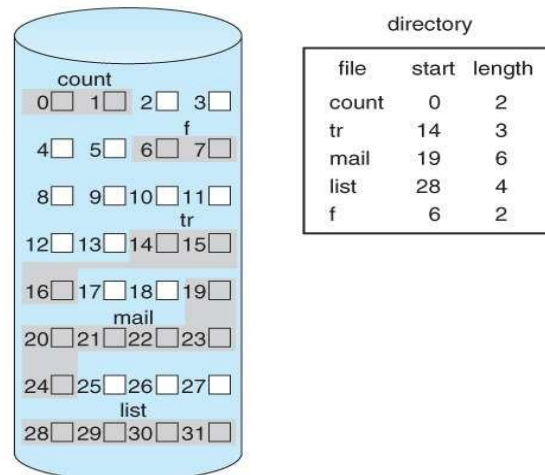
- **Creating a file** – Two steps are necessary to create a file. First, required space must be found in the file system. Second, an entry for the new file must be made in the directory. The directory records the name of the file, location in the file system and other information.
- **Writing a file** – To write a file, we make a system call specifying both the name of the file and the information to be written on to the file. Given the name of the file, the system searches for the directory to locate the file. The system must keep a write pointer to the location in a file where the next write is to take place. The write pointer must be updated whenever a write occurs
- **Reading a file** – To read from a file, we use a system call that specifies the name of the file and where the next block of file should be kept in memory. The system needs to keep a read pointer at a location in a file where the next read is to take place. Once the read has taken place, the read pointer is updated. Both read and write use the same pointer thus saving space and reducing system complexity
- **Repositioning within a file** – The directory is searched for appropriate entry and the current-file-position is set to a given value. This operation does not involve actual I/O. This file operation is also called file seek.
- **Deleting a file** – To delete a file, search the directory for a file. Having found the file, we release all the file space so that it can be reused by other files and erase that entry.
- **Truncating a file** – The user may want to erase the contents of a file. Rather than forcing the user to delete the file and recreate it, this function allows all attributes to remain unchanged except the length. It resets the file length to zero and releases the file space

APPLICATION

8. With a neat diagram explain contiguous allocation method.

This method requires each file to occupy a set of contiguous blocks on the disk. Disk addresses define linear ordering on the disk. With this ordering, it is possible to access $b+1$ block after b block with no head movement at all. Hence, the number of disk seeks in this method is minimal. Contiguous allocation of a file is defined by disk address and the length of the first block. If the file is n blocks long and starts at location b , then it occupies blocks $b, b+1, b+2, \dots, b+(n-1)$. The directory entry for each file indicates the address of the starting block and the length of the area allocated for the file. Accessing the file using this method is very easy. For sequential access, the file system remembers the disk address of the last block referenced and when necessary reads the next block. For direct access to block i of a file that starts at block b , we can immediately access block $b+i$.

One limitation here is to find free space for a new file. This is generally seen in dynamic storage allocation. First fit and the best fit are the most common strategies used to select a free hole from a set of available blocks. The above approach suffers from external fragmentation. As files are allocated and deleted, free disk space is broken into pieces. External fragmentation exists when free space is broken into chunks. When the largest contiguous chunk is insufficient for a request, storage is fragmented into a number of holes, none of which is large enough to store data. Another problem is to determine the space needed for a file



9. Consider the following reference string. Assuming three frames find the number of Page faults using FIFO page replacement algorithm.

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

The simplest page-replacement algorithm is a first-in first-out (FIFO) algorithm. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. We can create a FIFO queue to hold all pages in memory. We replace the page at the head of the queue. When page is brought into memory, we insert it at the tail of the queue.

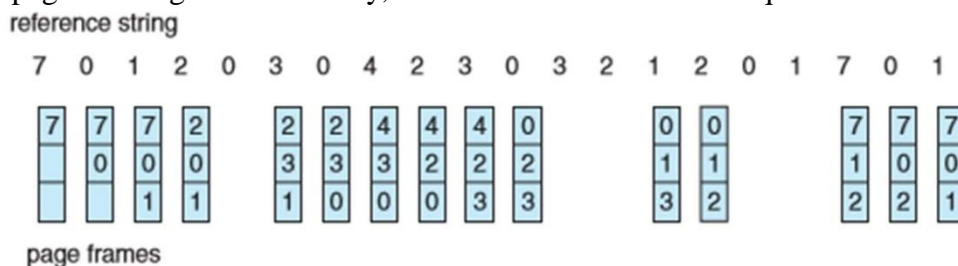


Figure 9.12 FIFO page-replacement algorithm.

Page faults/miss: 15

Page Hit: 5

10. Consider the following reference string. Assuming three frames find the number of Page faults using optimal page replacement algorithm.

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

Replace the page that will not be used for the longest period of time. Use of this page replacement algorithm guarantees the lowest possible page fault rate for a fixed number of frames.

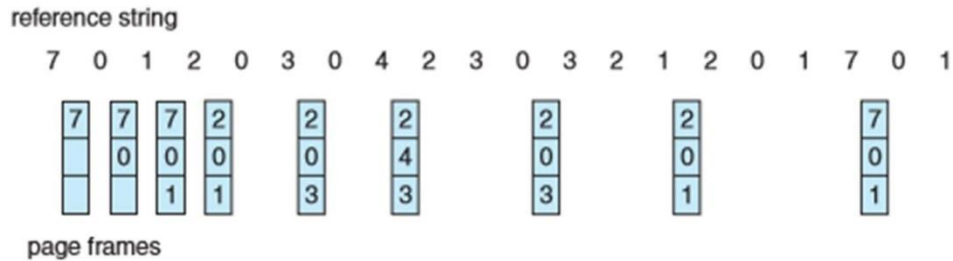


Figure 9.14 Optimal page-replacement algorithm.

Page faults/miss: 9

Page Hit: 11

11. Consider the following reference string. Assuming three frames find the number of Page faults using LRU replacement algorithm.

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

LRU replacement associates with each page the time of that page's last use. When a page must be replaced, LRU chooses the page that has not been used for the longest period of time. We can think of this strategy as the optimal page-replacement algorithm looking backward in time, rather than forward.

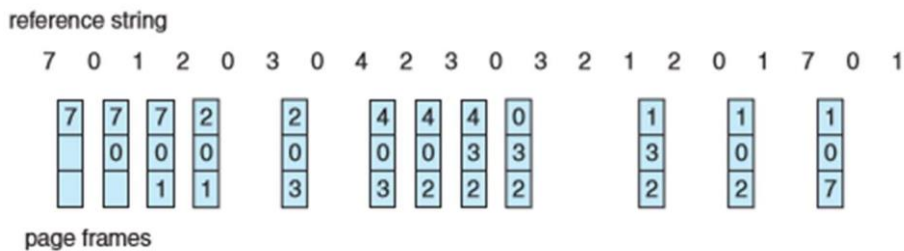


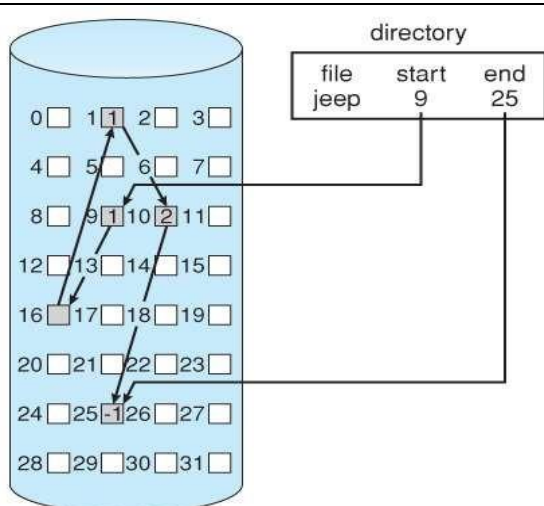
Figure 9.15 LRU page-replacement algorithm.

Page faults/miss: 12

Page Hit: 8

12. Briefly explain Linked allocation method with neat diagram.

This method solves the problems of contiguous allocation. With linked allocation, each file is a linked list of disk blocks. The disk blocks may be scattered anywhere on the disk. The directory contains a pointer to the first and last blocks of a file. To create a new file, we create a new entry in the directory. Each directory entry has a pointer to the first disk block of a file. This pointer is initialized to nil to signify an empty file. The size field is also set to 0. A write operation on the file causes a free block to be found and a new block is written to and linked to the end of the file. To read a file, we simply read the blocks by following the pointers. There is no external fragmentation and any free block on the free space can be used to satisfy a request, the size of a file need not be declared when the file is created.



13. Write a note on free space management.

Since disk space is limited, we need to reuse the space from deleted files for new files. To keep track of free disk space, the system maintains a free space list. It records all the free disk blocks- those not allocated to any file or directory. Free space is implemented as follows:

- **Bit vector:** Free space is usually implemented as a bit map or bit vector. Each block is represented by one bit. If the block is free the bit is 1 and if the block is allocated the bit is 0.
- **Linked list:** In this scheme, we link together all free disk blocks keeping a pointer to the first free block in a special location on the disk and caching it in memory. The first free block contains a pointer to the next free block and so on.
- **Grouping:** In this approach, the addresses of the first n free blocks are stored in the first free block. The first of these (n-1) blocks are free. The last block contains the address of another n blocks and so on.
- **Counting:** Here, we keep the address of the first free block and the number n of the free contiguous blocks that follow the first block. Each entry in the free space list consists of disk address and a count.

UNIT V

MULTIPLE CHOICE QUESTIONS APPLICATIONS

1. Which one of the following consist of all the shared library files
 - a. **Lib**
 - b. SBin
 - c. Bin
 - d. Var

2. Which one of the following consist of temporary file location
 - a. **tmp**
 - b. SBin
 - c. Bin
 - d. Var

3. Which among the following is the general command format
 - a. **command options command arguments**
 - b. options commandcommand arguments
 - c. command arguments options command
 - d. options command arguments command

4. The command line completes only after the user has hit the
 - a. **[Enter] key**
 - b. [control] key
 - c. [shift] key
 - d. [tab] key

5. Which command is used to list the content of the specified directory
 - a. **ls**
 - b. mkdir
 - c. rmdir
 - d. cd

6. Which among the following options of Ls command is used to list files in reverse order
 - a. **r**
 - b. t
 - c. p
 - d. s

7. Which among the following wild card character is used to represent any number of character
 - a. *****
 - b. ?
 - c. /
 - d. .

8. Which among the following wild card character is used to represent one character
 - a. *****

- b. ?
- c. /
- d. .

9. Which of the following commands is used to create new directory

- a. **mkdir**
- b. rmdir
- c. cd
- d. pwd

10. Which of the following commands is used to remove directory

- a. mkdir
- b. **rmdir**
- c. cd
- d. pwd

11. Which of the following commands is used to change the current working directory

- a. mkdir
- b. rmdir
- c. **cd**
- d. pwd

12. Which among the following command is used to copy the content of 1 file to another

- a. **cp**
- b. rm
- c. mv
- d. wc

13. Which among the following command is used to remove a file from specified directory

- a. cp
- b. **rm**
- c. mv
- d. wc

SKILL

14. Which among the following command is used to display the number of lines, words and characters of information stored on the specified file

- a. cp
- b. rm
- c. mv
- d. **wc**

15. Which among the following command is used to compare 2 files

- a. **cmp**
- b. cp
- c. mv
- d. wc

16. Which letter is used to represent execute mode of file access permissions
- a. **r**
 - b. w
 - c. x
 - d. e
17. Which command is used to execute specified Linux command at future time-
- a. **at**
 - b. kill
 - c. ps
 - d. batch
18. Which of the following prompt command is used to display next mail message if exist
- a. **+**
 - b. -
 - c. D
 - d. R
19. Which of the following command is used to display the specified month and year
- a. **cal**
 - b. who
 - c. date
 - d. who am I
20. Which command is used to display the system date and time
- a. **date**
 - b. cal
 - c. who
 - d. who am i
21. Which of the following command is used to display the users who are currently logged on the system
- a. cal
 - b. **who**
 - c. date
 - d. who am I
22. Which of the following command is used to perform arithmetic operations on integers
- a. **expr**
 - b. bc
 - c. split
 - d. none of the above
23. Which of the following command is used to perform arithmetic operations on integer and float
- a. expr
 - b. **bc**

- c. split
- d. None of the above

24. Linux considers which as the standard input

- a. **Keyboard**
- b. Mouse
- c. Terminal screen
- d. None of the above

25. Which command is used to sort the contents of a given file based on ASCII values of character

- a. **sort**
- b. list
- c. cat
- d. None of the above

LONG QUESTIONS SKILL

1. **Give the reasons for Popularity of Linux.**

- It is a freely distributed implementation of an UNIX-like kernel.
- Since it is written in 'c' language, it is flexible and portable.
- It supports multitasking: several programs running on the same machine at the same time, multiuser: several users working on the same machine at the same time and multiplatform: runs on many CPUs not just Intel.
- It is simply and consistent interface to peripheral devices.
- It is simple user interface, provides X-window system. The X-Window system provides the foundations for the graphical user interface available with LINUX.
- It provides data communication among users.
- It allows TCP/IP networking, including ftp, telnet, NFS, etc.
- The machine architecture is hidden from the user.
- It uses virtual memory using paging to disk (not swapping whole processes).

2. **Write a note on Linux File system.**

LINUX treats all information as files. Hence, files form an important part of the LINUX system. The LINUX file system is a data structure tree that resides on the part of a disk.

Almost the structure of the LINUX file system is same as UNIX file system.

Bin	It contains all essential executable LINUX program files
sbin	It contains executable files, which are used by super user
Horne	It contains the subdirectories of the LINUX users
etc	It contains the configuration files (administrative files) of all important programs
Var	It contains all your system logs
Usr	It is the central location for all your application program, x-windows, man pages, documents etc.
Lib	It contains all the shared library files

tmp	It is the temporary file location
dev	It contains the special device files for keyboard, mouse, console, etc.

3. Explain “ls” command along with its different options and syntax.

This command is used to list the content of the specified directory. General form is,

ls [-options] <directory_name>

Where **options** can be,

- **a** - Lists all directory entries including the hidden files
- **l** - Lists the files in long format (filenames along with file type, file permissions, number of links, owner of the file, file size, file creation/modification time, number of links for a file). The number of links for a file refers more than one name for a file, does not mean that there are more copies of that file. This **ls -l** option displays also the year only when the file was last modified more than a year back. Otherwise, it only displays the date without year.
- **r** - Lists the files in the reverse order
- **t** - Lists the files sorted by the last modification time
- **R** - Recursively lists all the files and sub-directories as well as the files in the sub-directories.
- **p** - Puts a slash after each director
- **s** - Displays the number of storage blocks used by a file.

<Directory _name> specifies a name of the directory whose contents are to be displayed. If the **<directory _name>** is not specified, then the contents of the current directory are displayed.

4. Summarize different general-purpose commands.

a. date

This command displays the system's date and time. General format is,

date +<format>

where **<format>** can be,

%H	Hour - 00 to 23
%I	Hour - 00 to 12
%M	Minute - 00 to 59
%s	Second - 00 to 59
%D	Date - MM / DD / YY
%T	Time - HH:MM:SS
%w	Day of the week
%r	Time in AM / PM
%y	Last two digits of the year

b. who

Since LINUX is a multi-user operating system, several users may work on this system. This command is used to display the users who are logged on the system currently. General format is,

who

c. who am i

This command tells you who you are. (Working on the current terminal). General format is,

who am i

d. cal

This command will display calendar for the specified month and year. General format is,

cal [<month>] <year>

Where, month can be ranged from 1 to 12.

e. lpr

This command is used to print one or more files on printer. General format is,

lpr [-options] <filename> <filename2> ... <filename N>

where **options** can be,

r Removes file(s) from directory after printing

m Mail inform you when printing is over

f. split

If a file is very large, then it cannot be edited on an editor. In such situations, we need to split the file into several small files. For this purpose, this split command is used. General format is,

split -<number> <filename>

The **-<number>** option splits the file specified in <filename> into <number> lined files.

g. expr

This command is used to perform arithmetic operations on integers. The arithmetic operators and their corresponding functions are given below.

- + Addition
- - Subtraction
- * Multiplication

- / Division
- % Remainder of division (modulus operator)

h. **bc**

This command is used to perform arithmetic operations on integers as well as on floats

5. Explain “sort” command with syntax, various options and valid examples.

This command sorts the contents of a given file based on ASCII values of characters. General format is,

sort [options] <filename>

where **options** can be,

-m <file list>	Merges sorted files specified in <filelist>
-o <filename>	Stores output in the specified <filename>
-r	Sorts the content in reverse order (reverse alphabetical order)
-u	Removes duplicate lines and display sorted content
-n	Numeric sort
-t "char "	Uses the specified "char " as delimiter to identify fields
-c	Checks if the file is sorted or not
+pos	Starts sort after skipping the pos th field
-pos	Stops sorting after the pos th field

Example

```
$ cat empname. txt
yasin
abdulla
ibrahim
$ sort empname. txt
abdulla
ibrahim
yasin
```

6. Explain different communication-oriented commands with appropriate syntax.
 - a. **write**

This online communication command lets you to write messages on another user’s terminal. General format is,

```
Write <RecipientLoginName>
<Message>
^d
```

If the recipient does not want to allow these messages (sent by other users) on his terminal, then he can use “ mesg n “ command. If he wants to revoke this option and want to allow anyone to communicate with him, then he can use “ mesg y “ command. General format is,

mesg [y | n]

y Allows write access to your terminal.

n Disallows write access to your terminal.

b. mail

This command offers off-line communication. General format is,

To send mail,

mail <username>

<message>

^d

The mail program mails the message to the specified user. If the user (recipient) is logged on the system, the message you have new mail is displayed on the recipient's terminal. However, the user is logged on the system or not, the mail will be kept in the mailbox until the user issues the necessary command to read the mails. However, the user is logged on the system or not, the mail will be kept in the mailbox until the user issues the necessary command to read the mails. To check mails, give the mail command without arguments. This command will list all the incoming mails received since the latest usage of the mail command. Some commands which can be given in mail prompt are given below,

Mail Prompt Commands

Functions

+	Displays the next mail message if exists
-	Displays the previous mail message if exists
<number>	Displays the <number> th mail message if exists
D	Deletes currently viewed mail and displays next mail message if exists
d <number>	Deletes the <number> th mail
s <filename>	Stores the current mail message to the file specified in <filename>
s<number> <filename>	Stores the <number> th mail message to the file specified in <filename>
R	Replies to the sender of the currently viewing mail

r <number> Replies the <number>th mail to its sender

Q Quits the mail program

7. Write a short note filters commands.

There are some LINUX commands that accept input from standard input or files, perform some manipulation on it, and produces some output to the standard output. Since these commands perform some filtering operations on data, they are appropriately called as "filters".

1. sort

This command sorts the contents of a given file based on ASCII values of characters. General format is,

sort [options] <filename>

where **options** can be,

-m <file list>	Merges sorted files specified in <filelist>
-O <filename>	Stores output in the specified <filename>
-r	Sorts the content in reverse order (reverse alphabetical order)
-u	Removes duplicate lines and display sorted content
-n	Numeric sort
-t "char "	Uses the specified "char " as delimiter to identify fields
-c	Checks if the file is sorted or not
+pos	Starts sort after skipping the pos th field
-pos	Stops sorting after the pos th field

2. uniq

This uniq (unique) command is used to handle duplicate lines in a file. If this command used without any option, it displays the lines by eliminating duplicate lines. General format is,

uniq [-option] <filename>

where **options** can be,

u	Displays only the non-repeated lines
d	Displays only the duplicated lines
c	Displays each line by eliminating duplicate lines, and prefixing the number of times it occurs

3. more

If the information to be displayed on the screen is very long, it scrolls up on the screen fastly. So, the user cannot be able to read it. This more command is used to display the output page by page. General format is,

more <filename>

4. pr

This command displays the contents of the specified file adding with suitable headers and footers. This command can be used with *lpr* command for neat hard copies. The Header part consists of the last modification date and time along with file-name and page number. General format is,

pr [-options] <filename>

where options can be,

-l <number> *It changes the page size to specified <number> of lines (By default, the page size is 66 lines)*

-<number> *Prepares the output in <number> columns*

-n *Numbers lines*

-t *Turns off the heading at the top of the page*

5. cut

This command is used to cut the columns or fields of a specified file (Like the head and tail commands cut the lines - rows). General format is,

cut [-options] <filename>

where **options** can be,

c<columns> Cuts the columns specified is<columns>. You must separate the column numbers by using commas

f <fields> Cuts the fields specified in <fields>. You must separate field numbers by using commas

6. paste

This command concatenates the contents of the specified files into a single file vertically. (Like cut command separates the columns, this paste command merges the columns). General format is,

paste <filename1> <filename2> ...

7. sed

This command acts as a line editor. General format is

sed 'EditComrnands' <filename>

Where **EditCommands** can be

- i** Inserts before line
- a** Appends after line
- c** Changes lines
- d** Deletes lines
- p** Prints lines
- q** Quits

8. Explain “write” command with syntax, various options and valid examples.

This online communication command lets you to write messages on another user’s terminal. General format is,

**Write <RecipientLoginName>
<Message>
^d**

Example

**\$ wirte ibrahim
Hello ibrahim
How are you?
^d**

On the execution of this command, the specified message is displayed on the terminal of the user - ibrahim. If the recipient does not want to allow these messages (sent by other users) on his terminal, then he can use “ mesg n “ command. If he wants to revoke this option and want to allow anyone to communicate with him, then he can use “ mesg y “ command. General format is,

mesg [y | n]

- y** Allows write access to your terminal.
- n** Disallows write access to your terminal.

The mesg without argument give the status of the mesg setting. The "finger" command can be used to know the users who are currently logged on the system and to know which terminals of the users are set to mesg y and which are set to mesg n. A '*' symbol is placed on those terminals where the mesg set to n.

9. Write a note on wild card characters with example.

“*” represents any number of characters and “?” represents a single character.

For Example,

\$ ls pgm*

This command will list out all the file-names of the current directory, which are starting with "pgm". Note that the suffix to pgm may be any number of characters.

\$ ls *s

This command will display all the filenames of the current directory, which are ending with "s". Note that the prefix to s may be any number of characters.

\$ ls? gms

This command will display four character filenames, which are ending with "gms" starting with any of the allowed character. Note that the prefix to gms is a single character. "[]" represents a subset of related filenames. This can be used with range operator "-" to access a set of files.

\$ ls pgm [1-5]

This command will list only the files named, pgm1, pgm2, pgm3, pgm4, pgm5 if they exist in the current directory. Note that the [1-5] represents the range from 1 to 5.

10. With the syntax and options explain following command

a. rm

This rm (remove) command is used to remove (delete) a file from the specified directory. To remove a file, you must have *write* permission for the directory that contains the file, but you need not have permission on the file itself. If you do not have write permission on the file, the system will prompt before removing. General format is,

rm [-options] <filename>

Where options can be,

- **r** deletes all directories including the lower order directories. Recursively deletes Entire contents of the specified directory and the directory itself
- **i** prompts before deleting
- **f** removes write-protected files also, without prompting

b. wc

This command is used to display the number of lines, words and characters of information stored on the specified file. General format is,

wc [-options] <filename>

Where options can be,

- **l** Displays the number of lines in the file
- **w** Displays the number of words in the file
- **c** Displays the number of characters in the file

11. With the syntax and example explain following command

a. **expr**

This command is used to perform arithmetic operations on integers. The arithmetic operators and their corresponding functions are given below.

- + Addition
- Subtraction
- * Multiplication
- / Division
- % Remainder of division (modulus operator)

A white space must be used on either side of an operator. Note that the multiplication operator (*) has to be escaped to prevent the Shell from interpreting it as the filename meta character. This command only works with integers, so, the division yields only integer part.

Examples

```
$ x=5
$ y=2
Then,
$ expr $x + $y
7
```

b. **bc**

This command is used to perform arithmetic operations on integers as well as on floats (decimal numbers). Type the arithmetic expression in a line and press [Enter] key. Then the answer will be displayed on the next line. After you have finished your work, press [Ctrl + d] keys to end up.

Examples

```
Add 10 and 20.
$ bc
10 + 20                // arithmetic expression
30                     // result is-displayed on the next line
Press ^d to end the work.
Divide 8 by 3.
```

12. Explain “date” command with various options give examples for each of the options.’

This command displays the system's date and time. General format is,

date +<format>

where <format> can be,

%H Hour - 00 to 23
%I Hour - 00 to 12

%M	Minute - 00 to 59
%s	Second - 00 to 59
%D	Date - MM / DD / YY
%T	Time - HH:MM:SS
%w	Day of the week
%r	Time in AM / PM
%y	Last two digits of the year

Examples

\$ date

Mon Dec 16 15:13:10 IST 2002

\$ date +%H

15

\$ date +%I

03

\$ date +%M

13

\$ date +%S

10

\$ date +%D

12/16/02

\$ date +%T

15:13:10

\$ date +%w

1 // 0=Sunday, 1=Monday, 2=Tuesday, ...

\$ date +%r

03:13:10 PM

\$ date +%y

02

13. Explain “mail” command with syntax, various options.

This command offers off-line communication. General format is,

To send mail,

mail <username>

<message>

^d

The mail program mails the message to the specified user. If the user (recipient) is logged on the system, the message you have new mail is displayed on the recipient's terminal. However, the user is logged on the system or not, the mail will be kept in the mailbox until the user issues the necessary command to read the mails. To check mails, give the mail command without arguments. This command will list all the incoming mails received since the latest usage of the mail command. Some commands which can be given in mail prompt are given below,

Mail Prompt Commands

Functions

+

Displays the next mail message if exists

-	Displays the previous mail message if exists
<number>	Displays the <number> th mail message if exists
D	Deletes currently viewed mail and displays next mail message if exists
d <number>	Deletes the <number> th mail
s <filename>	Stores the current mail message to the file specified in <filename>
s<number> <filename>	Stores the <number> th mail message to the file specified in <filename>
R	Replies to the sender of the currently viewing mail
r <number>	Replies the <number> th mail to its sender
Q	Quits the mail program

14. With the syntax and example explain following command

a. **kill**

If you want a command to terminate prematurely, press [ctrl + c]. This type of interrupt characters does not affect the background processes, because the background processes are protected by the Shell from these interrupt signals. This kill command is used to terminate a background process. General format is,

Kill [-SignalNumber] <PID>

The PID is the process identification number of the process that we want to terminate. Useps command to know the PIDs of the current processes. By default, this *kill* command uses the signal number 15 to terminate a process. However, some programs like login shell simply ignore this signal of interruption, and continue execution normally. In this case, you can use the signal number 9 (often referred as sure kill).

Examples

\$kill 120

This command terminate the process who has the PID 120.

b. **at**

This command is used to execute the specified LINUX commands at future time. General format is,

at <time>
<Commands>
^d

(Press [ctrl + d] at the end)

Here, <time> specifies the time at which the specified <commands> are to be executed.

Example

\$ at 12:00

echo, "LUNCH BREAK"

^d

at offers the keywords - now, noon, midnight, today and tomorrow and they convey

Special meanings.

\$ at noon // 12:00

echo, "LUNCH BREAK"

at also offers the keywords hours, days, weeks, months and years, can be used with + Operator as shown in the following Examples.

\$ at 12:00 + 1 day // at 12:00 tomorrow

15. Explain the following commands with example

a. mkdir

This mkdir (make directory) command is used to make (create) new directories. General format is,

mkdir [-p] <directory_name1> <directory_name2>

The **option -p** is used to create consequences of directories using a single mkdir command.

Examples

\$ mkdir ibr

This command will create 'ibr' a sub-directory of the current directory.

\$ mkdir ibr/ib/i

This command will make a directory i as a subdirectory of ibr/ib, but the directory structure -ibr / i must exist.

b. rmdir

This rmdir (remove directory) command is used to remove (delete) the specified directories. A directory should be empty before removing it. General format is,

rmdir [-p] <directory_name1> <directory_name2>

The **option -p** is used to remove consequences of directories using a single *rmdir command*.

Example

\$ rmdir ibr

This command will remove the directory ibr, which is the sub-directory of the current directory.

\$ rmdir ibr/ ib / i

This command will remove the directory i only.

\$ rmdir -p ibr / ib / i

This command will remove the directories ib, ibr and I consequently.