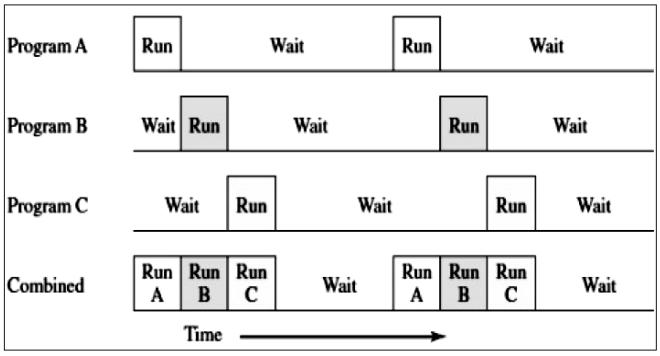
Q.1(a) Attempt any THREE of the following : [12]

Q.1(a) (i) Describe multiprogramming and multitasking. [4]

Ans.: Multiprogramming: In multiprogramming, more than one program lies in the memory. The scheduler selects the jobs to be placed in ready queue from a number of programs. The ready queue is placed in memory and the existence of more than one program in main memory is known as multiprogramming. Since there is only one processor, there multiple programs cannot be executed at a time. Instead the operating system executes part of one program, then the part of another and so on. Example of multiprogramming: user can open word, excel, access and other applications in a system.



OR

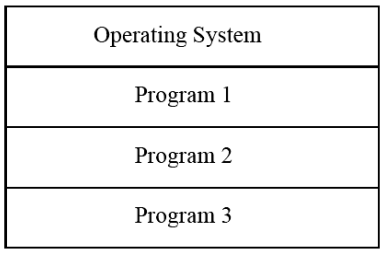


Fig- Multiprogramming with three programs

Multitasking:

A multitasking operating system is any type of system that is capable of running more than one task at a time. It also maintains the synchronization between I/O devices and processes so that user can use different application in background and current application in foreground. In multitasking the resources are made continuously working. The CPU switches from one task to another for reading and processing. Thus idle time of peripherals gets reduced. In multitasking Operating System the code as well as data of several processes is stored into main memory. For example, when you are printing a document of 100 pages, you can do other jobs like typing a new document. So, more than one task is performed.

Q.1(a) (ii) State any four types of system calls provided by an operating system. [4]

Ans.: System calls related to Process Control: End, Abort Load, Execute, Create process, Terminate process, Ready process, Dispatch process, Suspend, Resume, Get Process attribute, set attribute, Wait for time, Wait event, signal event.

System calls Related to File Management: Create file, delete file Open file , Close file Create directory Read, Write, Reposition Get file attribute , set file attribute, Create a link, Change the working directory.

System calls Related to Device Management: Request a device, Release a device Read, Write, Reposition, Get device attribute, Set device attribute

System calls Related to Information Maintenance: Get Time or Date, Set Time or Date, Get System data, Set system data, Get process, file or device attributes, Set Process, file or Device attributes.

System calls Related to Communication: create, delete communication connection send, receive messages, transfer status information, attach or detach remote devices

Q.1(a) (iii) Define realtime operating system. Explain with the help of example. [4]

Ans.: Real time systems are used in environment where a large number of events, mostly external to the computer system, must be accepted and processes in a short time or within certain deadlines. Such applications include real-time simulations, flight control, industrial control, military applications etc. A primary objective of real-time systems is to provide quick event response time and thus meet the scheduling deadlines. User convenience and resource utilization are of secondary concern to real-time system designers. In Real time systems, processor is allocated to the highest priority process among those that are ready to execute. Higher priority processes preempt execution of the lower priority processes. This form is called as 'priority –based preemptive scheduling'.

Example: Satellite application of real time OS
The satellite connected to the computer system sends the digital samples at the rate of 1000 Samples per second. The computer system has an application program that stores these samples in a file. The sample sent by the satellite arrives every millisecond to the application. So compute r must store or respond the sample in less than 1 millisecond. If the computer does not respond to the sample within this time, the sample will lost.
Some of the examples of Real time systems are: A web server, A word processor, An audio/video media center, A microwave oven, A chess computer.

Q.1(a) (iv) List any four functions of operating system. [4]

Ans.: The major functions of an operating system are:

1. **Resource Management:**
   This function of OS allocates computer resources such as CPU time, main memory, secondary storage and input and output devices for use.

2. **Data management:**
   It observes input and output of the data and their location, storage and retrieval.

3. **Task management:** Task is a collection of one or more related programs and their data. This function prepares, schedules, controls and monitors jobs submitted for execution to ensure the most efficient processing.

4. **Allocation of Resources:** Handles system resources such as computer's memory and sharing of the central processing unit (CPU) time by various applications or peripheral devices

5. **Communication between User and Computer:** Provides a user interface, e.g. command line, graphical user interface (GUI)

6. Operating system enables startup application programs. OS must have text editor, a translator and an editor.

7. Operating system provides number of services such as for the programmer it provides utilities i.e. debugger, editors, file management which refers to the way that the operating system manipulates, stores, retrieves and saves data. It interprets the commands executed by the user. It handles disk input/output settings.

OR

1. **Process Management:** Managing the programs that are running.
2. **Memory Management:** Managing and rationing the memory between processes and data.
3. **Storage Management:** Managing the permanent Storage of data on disks or other media
4. **I/O Management:** Managing the input and output
5. **Device / Resource Management:** Managing devices and resources and allowing the users to share the resources
6. **Security and Protection:** Securing the system against possible unauthorized access to data or any other entity. Protecting the parts of the system against damage.
7. **Booting the System and getting it ready to work.**
8. **Data communications:** Providing interface to connect to other computers or allowing others to connect

Q.1(b) Attempt any ONE of the following : [8]

Q.1(b) (i) State and describe services provided by an operating system. [8]

Ans.: Operating Services:

1. Program execution
2. I/O operations
3. File-system manipulation
4. Communications
5. Error detection
6. Accounting

1. **Program execution:** The operating system loads the contents (or sections) of a file into memory and begins its execution. A user-level program could not be trusted to properly allocate CPU time.

2. **I/O operations:** Disks, tapes, serial lines, and other devices must be communicated with at a very low level. The user need only specify the device and the operation to perform on it, while the system converts that request into device- or controller-specific commands. User-level programs cannot be trusted to access only devices they should have access to and to access them only when they are otherwise unused.

3. **File-system manipulation:** There are many details in file creation, deletion, allocation, and naming that users should not have to per-form. Blocks of disk space are used by files and must be tracked. Deleting a file requires removing the name file information and freeing the allocated blocks. Protections must also be checked to assure proper file access. User programs could neither ensure adherence to protection methods nor be trusted to allocate only free blocks and deallocate blocks on file deletion.

4. **Communications:** Message passing between systems requires messages to be turned into packets of information, sent to the net-work controller, transmitted across a communications medium, and reassembled by the destination system. Packet ordering and data correction must take place. Again, user programs might not coordinate access to the network device, or they might receive packets destined for other processes.

5. **Error detection:** Error detection occurs at both the hardware and software levels. At the hardware level, all data transfers must be inspected to ensure that data have not been corrupted in transit. All data on media must be checked to be sure they have not changed since they were written to the media. At the software level, media must be checked for data consistency; for instance, whether the number of allocated and unallocated blocks of storage match the total number on the device. There, errors are frequently processindependent (for instance, the corruption of data on a disk), so there

must be a global program (the operating system) that handles all types of errors. Also, by having errors processed by the operating system, processes need not contain code to catch and correct all the errors possible on a system.

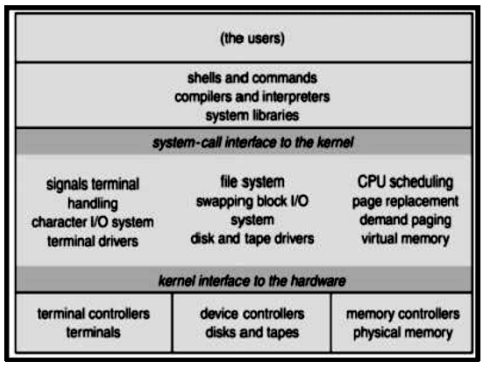
6. **Accounting:** We may want to keep track at which users use how much and what kind of computer resources. What was the login time for a particular user; is he working on the system right now, what is the process -1 D for the user, all such in formations we can manage using accounting service provided by many multiuser systems. This record keeping may be for the purpose of paying for the system & its operation, or simply for accounting usage statistics.

Q.1(b) (ii) Describe following operating system structures. [8]
(1) Monolithic (2) Microkarnel

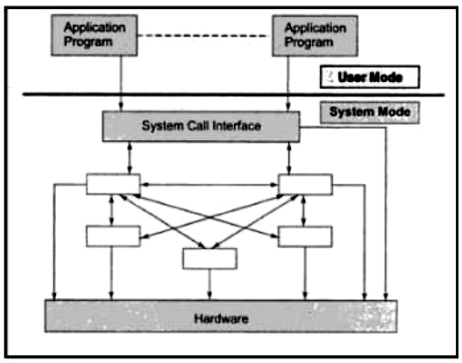
Ans.: (1) Monolithic Systems:

The structure is that there is no structure. The operating system is written as a collection of procedures, each of which can call any of the other ones whenever it needs to. When this technique is used, each procedure in the system has a well-defined interface in terms of parameters and results, and each one is free to call any other one, if the latter provides some useful computation that the former needs.

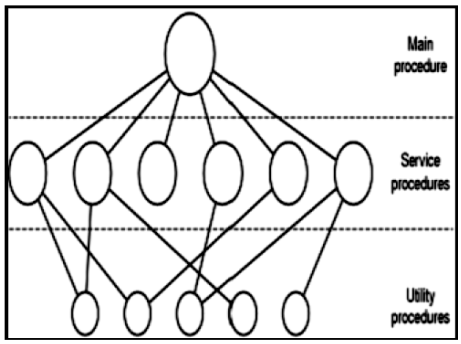
For constructing the actual object program of the operating system when this approach is used, one compiles all the individual procedures, or files containing the procedures, and then binds them all together into a single object file with the linker. In terms of information hiding, there is essentially none- every procedure is visible to every other one i.e. opposed to a structure containing modules or packages, in which much of the information is local to module, and only officially designated entry points can be called from outside the module.



OR



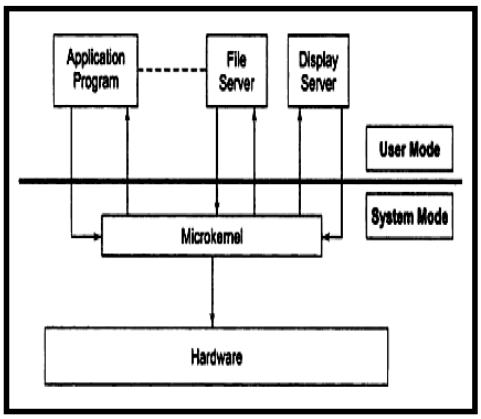
OR



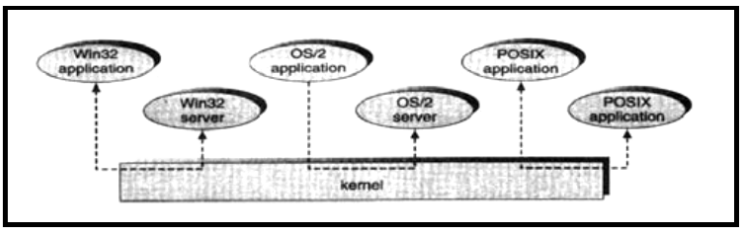
(2) Microkernel

A microkernel (also known as μ-kernel) is the near-minimum amount of software that can provide the mechanisms needed to implement an operating system (OS). These mechanisms include low-level address space management, thread management, and inter-process communication (IPC). If the hardware provides multiple rings or CPU modes, the

microkernel is the only software executing at the most privileged level (generally referred to as supervisor or kernel mode). Moves as much from the kernel into "user" space. Communication takes place between user modules using message passing.



OR



Q.2Attempt any FOUR of the following : [16]

Q.2(a) Differentiate between short term and long term scheduler. [4]

Ans.: {Note: Any other relevant difference shall be considered**}**

Sr. No	Short term scheduler	Long term scheduler
1	It is a CPU scheduler	It is a job scheduler
2	It selects processes from ready queue which are ready to execute and allocates CPU to one of them.	It selects processes from job pool and loads them into memory for execution.
3	Access ready queue and CPU.	Access job pool and ready queue
4	It executes frequently. It executes when CPU is available for allocation.	It executes much less frequently. It executes when memory has space to accommodate new process.
5	Speed is fast	Speed is less than short term scheduler
6	It provides lesser control over degree of multiprogramming	It controls the degree of multiprogramming

Q.2(b) Compare UNIX and LINUX operating system w.r.t. [4]

(i) User interface (ii) Name of provider

(iii) Processing speed (iv) Security

Ans.:

Parameter	Linux	Unix
User interface	Linux typically provides two GUIs, KDE and Gnome. But there are millions of alternatives such as LXDE, Xfce, Unity, Mate, twm, etc..	Initially Unix was a command based OS, but later a GUI was created called Common Desktop Environment. Most distributions now ship with Gnome.
Name of Provider	Redhat, Ubantu, Fedora	Osx, Solaris, All LINUX

Processing speed	**Low:** As it is GUI based processing time is more as compare to UNIX	**High:** As it is command based direct interpretation of commands is done so it takes less time as compare to LINUX
Security	Linux has had about 60- 100 viruses listed till date. None of them actively is spreading nowadays.	A rough estimate of UNIX viruses is between 85 -120 viruses reported till date.

Q.2(c) With neat diagram describe use of Process Control Block (PCB). [4]

Ans.: PCB is a record or a data structure that is maintained for each and every process. Every process has one PCB that is associated with it. A PCB is created when a process is created and it is removed from memory when process is terminated.

A PCB contains several types of information depending upon the process to which PCB belongs. The information stored in PCB of any process may vary from process to process. In general, a PCB may contain information regarding:

1. **Process Number:** Each process is identified by its process number, called process identification number (PID). Every process has a unique process-id through which it is identified. The Process-id is provided by the OS. The process id of two process could not be same because psid is always unique.

2. **Priority:** Each process is assigned a certain level of priority that corresponds to the relative importance of the event that it services Process priority is the preference of the one process over other process for execution. Priority may be given by the user/system manager or it may be given internally by OS. This field stores the priority of a particular process.

3. **Process State:** This information is about the current state of the process. i.e. whether process is in new, ready, running, waiting or terminated state.

4. **Program Counter:** This contains the address of the next instruction to be executed for this process.

5. **CPU Registers:** CPU registers vary in number and type, depending upon the computer architectures. These include index registers, stack pointers and general purpose registers etc. When an interrupt occurred, information about the current status of the old process is saved in registers along with the program counters. This information is necessary to allow the process to be continued correctly after the completion of an interrupted process.

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

7. **Memory Management Information:** This information may include such information as the value of base and limit registers, the page table or the segment table depending upon the memory system used by operating system.

8. **Accounting:** This includes actual CPU time used in executing a process in order to charge individual user for processor time.

9. **I/O Status:** It includes outstanding I/O request, allocated devices information, pending operation and so on.

10. **File Management:** It includes information about all open files, access rights etc.

Q.2(d) With neat diagram, explain structure of unix operating system. [4]

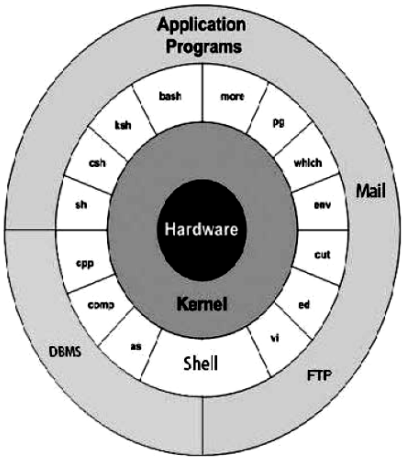
Ans.:

- **Kernel:** The kernel is the heart of the operating system. It interacts with the hardware and most of the tasks like memory management, task scheduling and file management.

- **Shell:** The shell is the utility that processes your requests. When you type in a command at your terminal, the shell interprets the command and calls the program that you want. The shell uses standard syntax for all commands. C Shell, Bourne Shell and Korn Shell are the most famous shells which are available with most of the Unix variants.

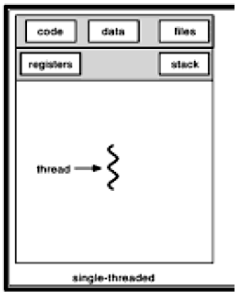
- **Commands and Utilities:** There are various commands and utilities which you can make use of in your day to day activities. cp, mv, cat and grep, etc. are few examples of commands and utilities. There are over 250 standard commands plus numerous others provided through 3rd party software. All the commands come along with various options.

- **Files and Directories:** All the data of Unix is organized into files. All files are then organized into directories. These directories are further organized into a tree-like structure called the **file system**.



Q.2(e) Define thread. State any three benefits of thread. [4]

Ans.: A thread, sometimes called a lightweight process, is a basic unit of CPU utilization. A traditional (or heavyweight) process has a single thread of control. If a process has multiple threads of control, it can do more than one task at a time. This is because there are situations in which it is desirable to have multiple threads of control in the same address space, running as though they were separate processes.



Threads benefits The benefits of multithreaded programming can be broken down into four major categories.

1. **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. **For example:** A multithreaded web browser could still allow user interaction in one thread while an image is being loaded in

another thread. A multithreaded Web server with a front-end and (thread) processing modules.

2. **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 threads of activity all within the same address space. A word processor with three threads. **For example:** A multithreaded word processor allows all threads to have access to the document being edited.

3. **Economy**: Because threads share resources of the process to which they belong, it is more economical to create and switch threads, than create and context switch processes (it is much more time consuming). For example: in Sun OS Solaris 2 creating a process is about 30 times slower than is creating a thread (context switching is about five times slower than threads switching).

4. **Utilization of multiprocessor architectures**: The benefits of multithreading can be greatly increased in a multiprocessor architecture (or even in a single-CPU architecture), where each thread may be running in parallel on a different processor.

Q.2(f) Define the following terms : [4]

(i) Preemptive scheduling (ii) Non-preemptive scheduling

Ans.: (i) Preemptive Scheduling

1. Even if CPU is allocated to one process, CPU can be preempted to other process if other process is having higher priority or some other fulfilling criteria.
2. It is suitable for RTS.
3. Only the processes having higher priority are scheduled.
4. It doesn't treat all processes as equal.
5. Circumstances for preemptive
   - process switch from running to ready state
   - process switch from waiting to ready State

(ii) Non-Preemptive Scheduling

1. Once the CPU has been allocated to a process the process keeps the CPU until it releases CPU either by terminating or by switching to waiting state.
2. It is not suitable for RTS.
3. Processes having any priority can get scheduled.
4. It treats all process as equal.
5. Circumstances for Non preemptive
   - Process switches from running to waiting state
   - Process terminates

Q.3Attempt any FOUR of the following : [16]

Q.3(a) Explain Round Robin algorithm with suitable example. [4]

Ans.: Round –Robin Scheduling:

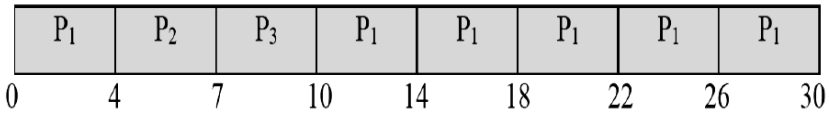
The Round–Robin (RR) scheduling algorithm is designed especially for time sharing systems. It is similar to FCFS scheduling, but preemption is added to enable the system to switch between processes. A small unit of time, called as time quantum or time slice, is defined. A time quantum is generally from 10 to 100 milliseconds in length. 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. One of two things will then happen. The process may have a CPU burst of less than 1 time quantum. In this case, the process itself will release the CPU voluntarily. The scheduler will then process to the

next process in the ready queue. Otherwise, if the CPU burst of the currently running process is longer than 1 time quantum, the timer will go off and will cause an interrupt to the operating system. A context switch will be executed, and the process will be put at the tail of the ready queue. The CPU scheduler will then select the next process in the ready queue. The average waiting time under the RR policy is often long. Consider the following set of processes that arrive at time 0, with the length of the CPU burst given in milliseconds:

Example:

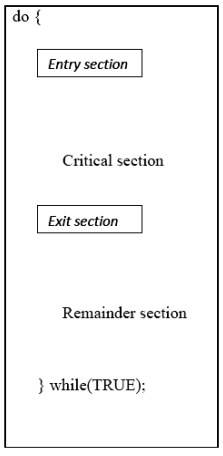
Process	Burst Time
P1	24
P2	3
P3	3

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



Q.3(b) Describe the critical-section problem. **[4]**

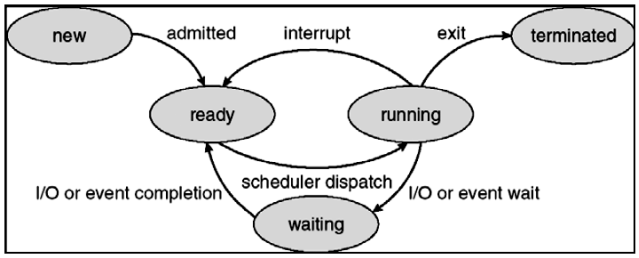
Ans.: Each process contains two sections. One is critical section where a process may need to access common variable or objects and other is remaining section containing instructions for processing of sharable objects or local objects of the process. Each process must request for permission to enter inside its critical section. The section of code implementing this request is the entry section. In entry section if a process gets permission to enter into the critical section then it works with common data. At this time all other processes are in waiting state for the same data. The critical section is followed by an exit section. Once the process completes its task, it releases the common data in exit section. Then the remaining code placed in the remainder section is executed by the process.



Two processes cannot execute their critical sections at the same time. The critical section problem is to design a protocol that the processes can use to cooperate i.e. allowing entry to only one process at a time inside the critical section. Before entering into the critical section each process must request for permission to entry inside critical section.

Q.3(c) Draw the process state diagram and describe each state in one/two sentences. [4]

Ans.:



New: The process being created is available in the new state. It is the new state because the system is not permitted it to enter the ready state due to limited memory available in the ready queue. If some memory becomes available, then the process from the new state will go to ready state.

Ready State: The process which is not waiting for any external event such as I/O operation and which is not running is said to be in ready state. It is not in the running state because some other process is already running. It is waiting for its turn to go to the running state.

Running State: The process which is currently running and has control of the CPU is known as the process in running state. In single user system, there is only one process which is in the running state. In multiuser system, there are multiple processes which are in the running state.

Blocked State: The process that is currently waiting for external event such as an I/O operation is said to be in blocked state. After the completion of I/O operation, the process from blocked state enters in the ready state and from the ready state when the process turn will come it will again go to running state.

Terminated / Halted State: The process whose operation is completed, it will go the terminated state from the running state. In halted state, the memory occupied by the process is released.

Q.3(d) State and explain criteria in CPU scheduling. [4]

Ans.: 1. CPU utilization
2. Throughput
3. Turnaround time
4. Waiting time
5. Response time

Explanation of criteria for CPU scheduling

1. **CPU utilization:** Keep the CPU as busy as possible.
2. **Throughput:** Number of processes that complete their execution per time unit.
3. **Turnaround time:** Amount of time to execute a particular process. The interval from the time of submission of a process to the time of completion is the turnaround time.
4. **Waiting time:** Amount of time a process has been waiting in the ready queue
5. **Response time:** Amount of time it takes from when a request was submitted until the first response is produced, not output (for time-sharing environment)

Q.3(e) Compare paging and segmentation memory management techniques. [4]

Ans.:

Sr. No.	Paging	Segmentation
1	It divides the physical memory into frames and program's address space into same size pages.	It divides the computer's physical memory and program's address space into segments.
2	Page is always of fixed block size.	Segment is of variable size.
3	The size of the page is specified by the hardware.	The size of the segment is specified by the user.

4	It may lead to internal fragmentation as the page is of fixed block size.	It may lead to external fragmentation as the memory is filled with the variable sized blocks.
5	Page table is used to map pages with frames from memory.	Segment table is used to map segments with physical memory.
6	Page table contains page number and frame number.	Segment table contains segment number, length of segment and base address of segment from memory.

Q.3(f) Explain in detail how deadlock can be handled. [4]

Ans.: Method for Handling Deadlocks

There are three different methods for dealing with the deadlock problem:

- We can use a protocol to ensure that the system will never enter a deadlock state
- We can allow the system to enter a deadlock state and then recover.
- We can ignore the problem all together, and pretend that deadlocks never occur in system. This solution is the one used by most operating systems, including UNIX.

1. To ensure that deadlocks never occur, the system can be use either a 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.

2. 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, and the future requests and releases of each process, to decide whether the current request can be satisfied or must be delayed.

3. If the system does not employ either a deadlock-prevention or deadlock – avoidance algorithm, 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.

4. 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

Q.4Attempt any TWO of the following : [16]

Q.4(a) Differentiate between short term, medium term and long term scheduling. [8]

Ans.:

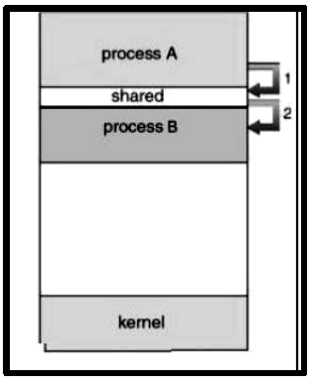
Sr. No.	Long term scheduler	Medium term scheduler	Short term scheduler
1	It is a job scheduler	It is a process swapping scheduler	It is a CPU scheduler
2	It selects processes from job pool and loads them into memory for execution.	It selects a process from swapped-out process.	It selects processes from ready queue which are ready to execute and allocates CPU to one of them.
3	Access job pool and ready queue	Access process from swapped out process queue.	Access ready queue and CPU.

4	It executes much less frequently when ready queue has space to accommodate new process.	It executed whenever swapped queue contains a swapped out process.	frequently select a new process for the CPU, at least once every100 milliseconds
5	Speed is less than short term scheduler	Speed is in between both short and long term scheduling	Speed is fast
6	It is almost absent or minimal in time sharing system	It is a part of time sharing system	It is also minimal in time sharing system
7	It controls the degree of multiprogramming	It reduces the degree of multiprogramming	It provides lesser control over degree of multiprogramming

Q.4(b) What is inter process communication? Describe any one technique of it. [8]

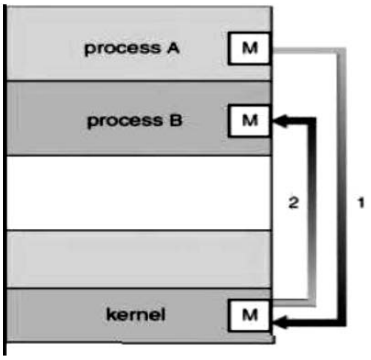
Ans.: Inter-process communication: Cooperating processes require an Inter- process communication (IPC) mechanism that will allow them to exchange data and information. There are two models of IPC

1. Shared memory



In this a region of the memory residing in an address space of a process creating a shared memory segment can be accessed by all processes who want to communicate with other processes. All the processes using the shared memory segment should attach to the address space of the shared memory. All the processes can exchange information by reading and/or writing data in shared memory segment. The form of data and location are determined by these processes who want to communicate with each other. These processes are not under the control of the operating system. The processes are also responsible for ensuring that they are not writing to the same location simultaneously. After establishing shared memory segment, all accesses to the shared memory segment are treated as routine memory access and without assistance of kernel.

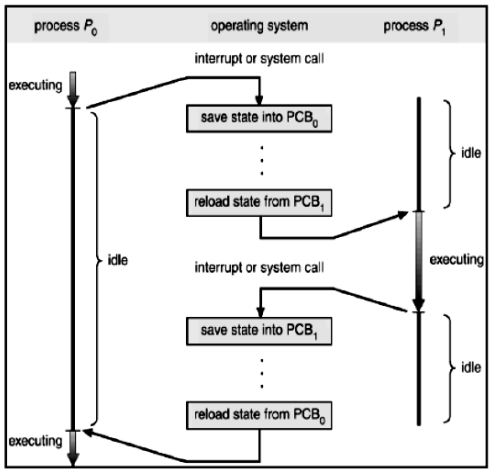
2. Message Passing



In this model, communication takes place by exchanging messages between cooperating processes. It allows processes to communicate and synchronize their action without sharing the same address space. It is particularly useful in a distributed environment when communication process may reside on a different computer connected by a network. Communication requires s sending and receiving messages through the kernel. The processes that want to communicate with each other must have a communication link between them. Between each pair of processes exactly one communication link.

Q.4(c) Explain context switch with help of diagram. [8]

Ans.: Switching the CPU to another process requires saving the state of current process and loading the saved state for new process. This process is known as a context switch. The context switch is represented in PCB. System saves context of old process in its PCB and loads context of new process into the memory which is schedule to run next.



Q.5Attempt any TWO of the following : [16]

Q.5(a) Explain priority scheduling algorithm with example. List its advantages and disadvantages. [8]

Ans.: Priority scheduling algorithm:

The SJF algorithm is a special case of the general priority scheduling algorithm Number (integer) is associated with each process.

The CPU is allocated A priority to the process with the highest priority (smallest integer = highest priority)

Priority scheduling can be either pre-emptive or non-pre-emptive

- A pre-emptive priority algorithm will pre-emptive the CPU if the priority of the newly arrival process is higher than the priority of the currently running process.
- A non-pre-emptive priority algorithm will simply put the new process at the head of the ready queue.

A major problem with priority scheduling is indefinite blocking or starvation. A solution to the problem of indefinite blockage of the low-priority process is aging. Aging is a technique of gradually increasing the priority of processes that wait in the system for a long period of time. SJF is a priority scheduling where priority is the predicted next CPU burst time

Advantage Priority Scheduling

- Simplicity.
- Reasonable support for priority.
- Suitable for applications with varying time and resource requirements.

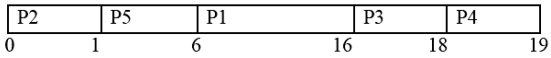
Disadvantages of Priority Scheduling

- Indefinite blocking or starvation.
- A priority scheduling can leave some low priority waiting processes indefinitely for CPU.
- If the system eventually crashes then all unfinished low priority processes gets lost.

Example:

PROCESS	BURST TIME	PRIORITY
P1	10	3
P2	1	1
P3	2	4
P4	1	5
P5	5	2

Gantt chart:



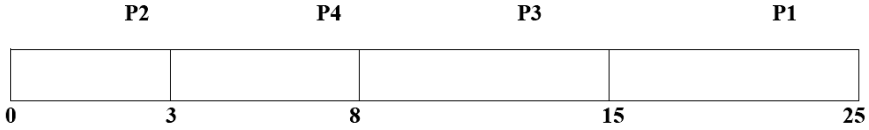
Waiting time for each process: p1 =6, p2 =0, p3 =16, p4 =18, p5 =1
Average waiting time: = (6+0+16+18+1)/5 =41/5 = 8.2 milliseconds

Q.5(b) Solve the following problem using SJF and Round Robin (RR) scheduling algorithm. [8]
Find average waiting time for each algorithm.

Process	Burst time
P_1	10
P_2	3
P_3	7
P_4	5

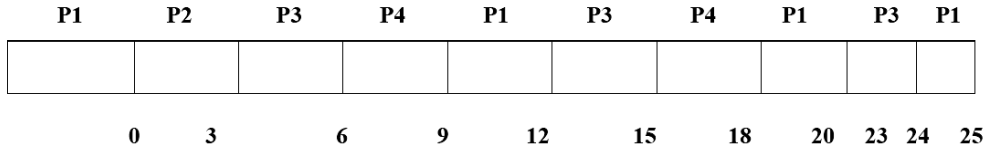
Ans.: SJF

Gantt chart



Waiting time
P1 15 mesc
P2 0 mesc
P3 8 msec
P4 3 msec
Average waiting time=waiting time of all processes/number of processes
=waiting time of (P1+P2+P3+P4)/4
=15+0+8+3/4=26/4=6.5 mesc

Round Robin:
[Note: assuming time quantum of 3 msec]**
Gantt chart:



Waiting time
P1 = 0 + (12-3) + (20-15) + (24-23) = 0 + 9 + 5 + 1 = 15msec
P2 = 3msec
P3 = 6 + (15-9) + (23-18) = 6 + 6 + 5 = 17msec
P4 = 9 + (18-12) = 9 + 6 = 15msec
Average waiting time=waiting time of all processes/number of processes
=waiting time of (P1+P2+P3+P4)/4
=15+3+17+15/4=50/4=12.5msec

Q.5(c) Write steps for Banker's Algorithm to avoid dead lock. Also give one example showing working of Banker's Algorithm. [8]

Ans.: Banker's Algorithm:

This algorithm calculates resources allocated, required and available before allocating resources to any process to avoid deadlock. It contains two matrices on a dynamic basis. Matrix A contains resources allocated to different processes at a given time. Matrix B maintains the resources which are still required by different processes at the same time.

Algorithm F: Free resources

Step 1: When a process requests for a resource, the OS allocates it on a trial basis.

Step 2: After trial allocation, the OS updates all the matrices and vectors. This updating can be done by the OS in a separate work area in the memory.

Step 3: It compares F vector with each row of matrix B on a vector to vector basis.

Step 4: If F is smaller than each of the row in Matrix B i.e. even if all free resources are allocated to any process in Matrix B and not a single process can completes its task then OS concludes that the system is in unstable state.

Step 5: If F is greater than any row for a process in Matrix B the OS allocates all required resources for that process on a trial basis. It assumes that after completion of process, it will release all the recourses allocated to it. These resources can be added to the free vector.

Step 6: After execution of a process, it removes the row indicating executed process from both matrices.

Step 7: This algorithm will repeat the procedure step 3 for each process from the matrices and finds that all processes can complete execution without entering unsafe state. For each request for any resource by a process OS goes through all these trials of imaginary allocation and updation. After this if the system remains in the safe state, and then changes can be made in actual matrices.

Example:

5 processes *P*0 through *P*4;

3 resource types:

A (10 instances), *B* (5instances), and *C* (7 instances)

Snapshot at time *T*0:

	Allocation	Max	Available
	A B C	*A B C*	*A B C*
P0	0 1 0	7 5 3	3 3 2
P1	2 0 0	3 2 2	
P2	3 0 2	9 0 2	
P3	2 1 1	2 2 2	
P4	0 0 2	4 3 3	

The content of the matrix Need is defined to be Max – Allocation

	Need
	A B C
P0	7 4 3
P1	1 2 2
P2	6 0 0
P3	0 1 1
P4	4 3 1

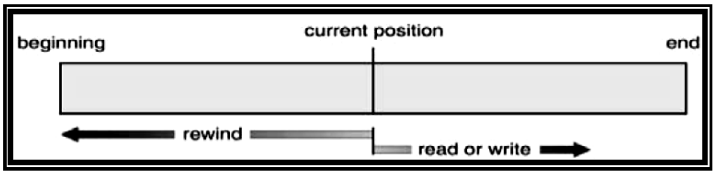
The system is in a safe state since the sequence < P1, P3, P4, P2, P0 > satisfies safety criteria

Q.6Attempt any TWO of the following : [16]

Q.6(a) Describe sequential file access method and direct access? [8]

Ans.: Information in the file is processed in order, one record after the other. This is by far the most common mode of access of files. For example, computer editors usually access files in this fashion.

A read operation reads the next portion of the file and automatically advances the file pointer. Similarly, a write appends to the end of the file and the file pointer. Similarly, a write appends to the end of the file and the file pointer. Similarly, a write appends to the end of the end of the file and advances to the end of the newly written material (the new end of file). Such a file can be reset to the beginning, and, on some systems, a program may be able to skip forward or backward n records, for some integer n. This scheme is known as sequential access to a file. Sequential access is based on a tape model of a file.



Q.6(b) Describe the concept of virtual memory with suitable example. [8]

Ans.: Virtual memory is the separation of user logical memory from physical memory. This separation allows an extremely large virtual memory to be provided for programmers when only a smaller physical memory is available. Virtual memory makes the task of programming much easier, because the programmer no longer needs to worry about the amount of physical memory available, or about what code can be placed in overlays, but can concentrate instead on the problem to be programmed. On systems which support virtual memory, overlays have virtually disappeared.

Example:

For example, a 16M program can run on a 4M machine by carefully choosing which 4M to keep in memory at each instant, with pieces of the program being swapped between disk and memory as needed.

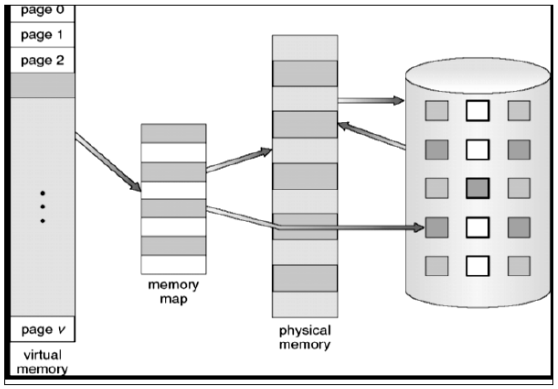


Fig: Virtual Memory

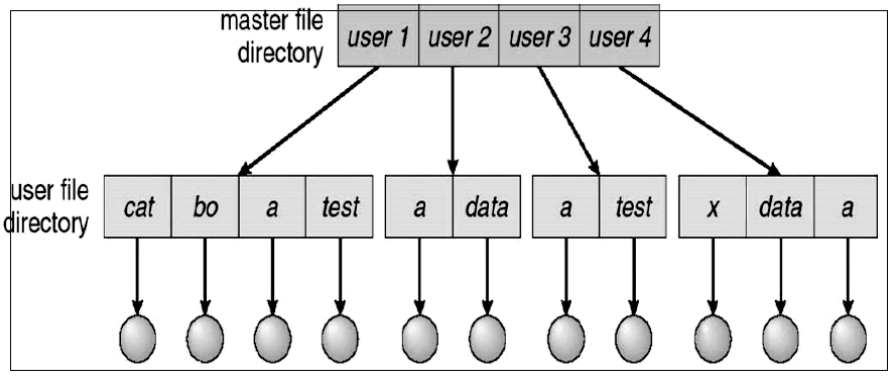
Q.6(c) Explain two level Directory Structure with suitable diagram and single level. [8]

Ans.: Two level directory:

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

Advantages:
(i) Path name
(ii) Can have the same file name for different user
(iii) Efficient searching

Disadvantages:
No grouping capability



❑ ❑ ❑ ❑ ❑ ❑