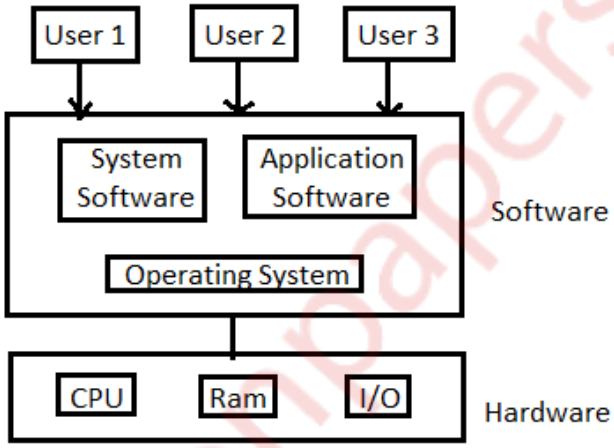
Operating System

(May 2019)

Q-1

a. Define Operating System. Brief the Functions of OS. **5**

➔ Definition: Operating System is an interface between user & the hardware. It is an
Software used for communication between user and the hardware and also controls the
execution of application programs. It is also called as "Resource Management".



Functions: Operating systems have various functions mentioned below

1) Process management –
* Creation, Execution & Deletion of user and system processes.
* Control the execution of user's application.
* Scheduling of process.
* Synchronization, inter-process communication and deadlock handling for processes.

2) Memory management -
* It allocates primary as well as secondary memory to the user and system and system process.
* Reclaims the allocated memory from all the processes that have finished its execution.
* Once used block become free, OS allocates it again to the processes.
* Monitoring and keeping track of how much memory used by the process.

3) File management -
* Creation & Deletion of files and directories.
* It allocates storage space to the files by using different file allocations methods.
* It keeps back-up of files and offers the security for files.

4) Device management -
* Device drivers are open, closed and written by OS.
* Keep an eye on device driver. communicate, control and monitor the device driver.

5) Protection & Security -
* The resources of the system are protected by the OS.
* Keeps back-up of data.
* Lock the system for security purpose by password.

b. Explain Shell. Explain use of chmod command in linux. 5

➔Shell

Shell is a system in which we can run our commands, programs, and shell scripts, There are different flavours of a shell, just as there are different flavour's of operating systems. Each flavour of shell has its own set of recognizable commands and functions.

Chmod

chmod – It change file mode bits.

- chmod changes the permissions of each given file according to given mode, where the mode describes the permissions to modify. Mode can be specified with octal numbers or with letters.

- In Linux, who can do what to a file or directory is controlled through sets of permissions. There are three sets of permissions. One set for the owner of the file, another set for the members of the file's group, and a final set for everyone else.

- In Linux operating systems, **chmod** is the command and system call which is used to change the access permissions of file system .

- Classes of users are used to distinguish to whom the permissions giving. If no classes are specified 'all' is implied. The classes are represented by one or more of the following letters:

Reference	Class	Description
u	user	Owner of file
g	group	File group members
o	others	Not owner and not a member of file system
a	all	All the three of above commands

c. Discuss various scheduling criteria. **5**

➔ A number of scheduling algorithms are being designed that can be applied to different processes having different properties. The scheduling criteria are mentioned below.

- CPU Utilization: It is amount of time CPU remains busy.
- Throughput: Number of jobs processed per unit time.
- Turnaround Time: Time elapsed between submission of jobs an completion of its execution.
- Waiting Time: Processes waits in ready queue to get CPU. Sum of times spent In ready queues waiting time.
- Time: Time from submission till the first response is produced.
- Fairness: Every process should get fair share of the CPU time.

d. Explain the effect of page frame size on performance of page replacement algorithms. **5**

➔

- The number of frames is equal to the size of memory divided by the page-size. So and increase in page size means a decrease in the number of available frames.
- Having a fewer frames will increase the number of page faults because of the lower freedom in replacement choice.
- Large pages would also waste space by Internal Fragmentation.
- On the other hand, a larger page-size would draw in more memory per fault; so the number of fault may decrease if there is limited contention.
- Larger pages also reduce the number of TLB misses.

An important hardware design decision is the size of page to be used. There are several factors to consider.

Internal fragmentation:

- Clearly, the smaller the page size, the lesser is the amount of internal fragmentation. To optimize the use of main memory, we would like to reduce internal fragmentation.
- On the other hand, smaller the page, the greater is the number of pages required per process which could mean that some portion of page tables of active processes must be in virtual memory, not in main memory. This eventually leads to double page fault for a single reference of memory.

Rate at which page fault occurs:

- If the page size is very small, then ordinarily a large number of pages will be available in main memory for process, which after some time will contain portions of process near recent references leading to low page fault rate.
- As the size of page is increased, each individual page will contain locations further and further from any particular recent reference. Thus, the effect of the principle of locality is weakened and the page fault rate begins to rise.
- Eventually, however the page fault rate will begin to fall as the size of page approaches the size of the entire process.

e. Explain Thrashing. **5**

➔ -A process should have some minimum number of frames to support active pages which are in memory.

- It helps to reduce the number of page faults. If these numbers of frames are not available to the process then it will quickly page fault.

- To handle this page fault, it is necessary to replace the existing page from memory.

- Since all the pages fault, it is necessary to replace the existing page from memory.

-Since in paging, pages are transferred between main memory and disk, this has an enormous overhead.

- Because of this frequent movement of pages between main memory and disk, system throughput reduces.

- This frequent paging activity causing the reduction in system throughput called as thrashing.

-Although many processes are in memory, due processes are in memory, due to thrashing CPU utilization goes low.

- When operating system monitors this CPU utilization, it introduce new process in memory to increase the degree of multiprogramming.

- Now it is needed that the existing pages should be replaced for the new process.

-If global page replacement algorithm is used, it replaces the pages of other process and allocates frames to this newly introduced process. So other processes whose pages are replaced are also causes page faults.

- All these faulting processes go in wait state and waits for paging devices. In this case again CPU utilization goes low.

-There is no actual work getting done and processes spend time only in paging.

-This thrashing can be limited by using local page replacement algorithm instead of global page replacement algorithm.

Q-2

a. Differentiate between monolithic, layered and microkernel structure of OS. 10

Monolithic Kernel	**Micro Kernel**
1. If virtually entire operating system code is executed in kernel mode, then it is a monolithic program that runs in a single address space.	1. In microkernel, set of modules for managing the hardware is kept which can uniformly well be executed in user mode. A small microkernel contains only code that must execute in kernel mode. It is the second part of operating system.
2. There is same address space for user mode as well as kernel mode.	2. There is different address space for user mode as well as kernel mode.
3. It has a large space as compared to micro kernel.	3. It has a small space as compared to monolithic kernel.
4. Execution speed is faster than micro kernel.	4. Execution speed is slower than monolithic kernel.
5. If one service crashes whole operating system fails.	5. If one service crashes whole operating system do not fails, it does not affect working of other part micro kernel.
6. Kernel calls the function directly for communication.	6. Communication is done through message passing.
7. To write monolithic kernel less code is required.	7. To write microkernel more code is required.
8. It is hard to extend.	8. It is easily extendible.
9. Example: Linux, BSDs, etc.	9. Example: QNX, Symbian, L4Linux, etc.
10. It is not flexible as compared to microkernel.	10. It is more flexible.
11. kernel calls the function directly.	11. In microkernel, communication is through message passing.

b. Describe the differences among short term, medium-term, and long term Scheduling. **10**

➔

Sr. No	Long-term Scheduler	Short-term scheduler	Medium-term Scheduler
1	Select processes from the queue and loads them into the memory for execution.	Chose the process from ready queue and assign it to the CPU.	Swap in and out the processes from memory.
2	Speed is less than short term scheduler.	Speed is very fast and invoked frequently than long term scheduler.	Speed is in between both the short term and long term.
3	Transition of process state from new to ready.	Transition of process state from ready to executing.	No process state transition
4	Not present in time sharing system.	Minimal in time sharing system.	Present in time sharing system.
5	Supply a reasonable mix of jobs, such as I/O bound CPU bound.	Select new process to allocate to CPU frequently.	Processes are swapped in and out for balanced process mix.
6	It controls degree of multiprogramming	It has control over degree of multiprogramming	Reduce the degree of multiprogramming
7	It is also called as job scheduler.	It is also called as CPU scheduler.	Swapping scheduler
8	The decision to add to the pool of processes to be execute	The decision to add to the number of processes that are partially or fully in main memory.	The decision as to which available processes will be executed by the processor.

Q-3

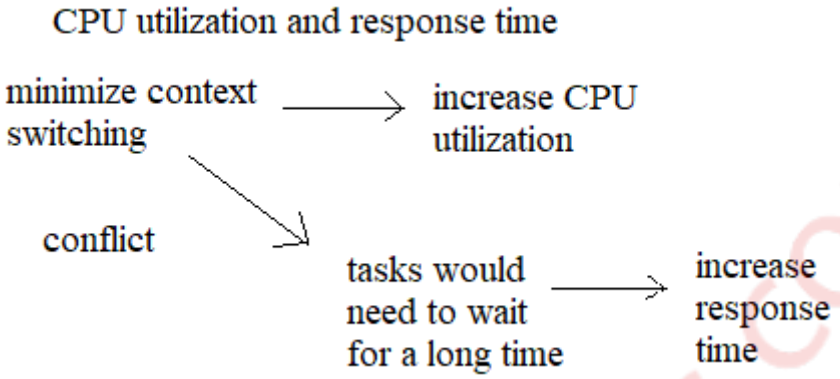
a. Discuss how the following pairs of scheduling criteria conflict in certain setting **10**

a) CPU utilization and response time

b) Average Turnaround time and maximum waiting time

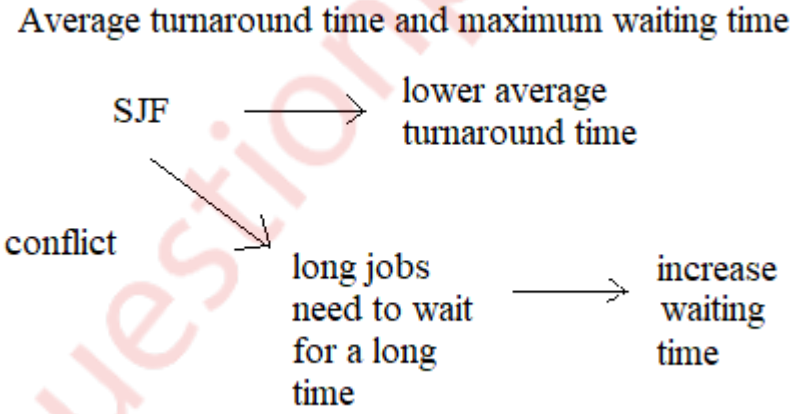
➔ **CPU utilization and response time:**

- CPU utilization is increased if the overhead associated with context switching is minimized. - The context switching overheads could be lowered by performing context switches infrequently.



Average turnaround time and maximum waiting time:

- Average turnaround time is minimized by executing the shortest tasks first.
- Such a scheduling policy could, however, starve long-running tasks and thereby increase their waiting time.
- Average turnaround time and maximum waiting time SJF lower average turnaround time.



I/O device utilization and CPU utilization: CPU utilization is maximized by running long-running CPU-bound tasks without performing context switches, I/O device utilization is maximized by scheduling I/O-bound jobs as soon as they become ready to run, thereby incurring the overheads of context switches.

I/O device utilization and CPU utilization

maximum CPU utilization ⟵ run long running CPU-bound tasks without context switching

maximize I/O device utilization ⟵ schedule I/O-bound tasks as soon as they're ready need context switching

b. Consider the following snapshot of the system. Using Bankers Algorithm, determine whether or not system is in safe state. If yes determine the safe sequence. **10**

	Allocation	**Max**	**Available**
	A B C D	**A B C D**	**A B C D**
P0	**3 0 1 4**	**5 1 1 7**	**0 3 0 1**
P1	**2 2 1 0**	**3 2 1 1**	
P2	**3 1 2 1**	**3 3 2 1**	
P3	**0 5 1 0**	**4 6 1 2**	
P4	**4 2 1 2**	**6 3 2 5**	

➔

Banker's Algorithm is a deadlock avoidance algorithm that checks for safe or unsafe state of a System after allocating resources to a process.

When a new process enters into system ,it must declare maximum no. of instances of each resource that it may need.After requesting operating system run banker's algorithm to check whether after allocating requested

resources,system goes into deadlock state or not. If yes then it

will deny the request of resources made by process else it allocate resources to that process.

No. of requested resources (instances of each resource) may not exceed no. of available resources in operating system and when a process completes it must release all the requested and already allocated resources.

Process	Need(Max-Allocation)			
	A	B	C	D
P0	2	1	0	3
P1	1	0	0	1
P2	0	2	0	0
P3	4	1	0	2
P4	2	1	1	3

Need matrix is calculated by subtracting Allocation Matrix from the Max matrix.

To check if system is in a safe state

P0 starts with available and proceed with the process P0

now available memory is 0301+3014=3315

Now P1 starts and P1 needs 1001

now available memory is 3315+1001=4316

now P2 starts and P2 needs 0200

now available memory is 4361+0200=4561

now p3 starts and P3 needs 4102

now available memory is 4561+4102= 8663

now P4 starts and P4 needs 2113

now available memory is 8663+2113= 10, 7, 7, 6

Available memory is 10A, 7B, 7C, 6D

The system is not in safe state

Q-4

a. Calculate number of page faults and page hits for the page replacement policies FIFO, Optimal and LRU for given reference string 6, 0, 5, 2, 0, 3, 0, 4, 2, 3, 0, 3, 2, 5, 2, 0, 5, 6, 0, 5 (assuming three frame size). 10

➔ **Frame Size =3**

i) FIFO

Frame	6	0	5	2	0	3	0	4	2	3	0	3	2	5	2	0	5	6	0	5
0	6	6	6	2	2	2	2	4	4	4	0	0	0	0	0	0	0	6	6	6

1		0	0	0	0	3	3	3	2	2	2	2	2	5	5	5	5	5	0	0
2			5	5	5	5	0	0	0	3	3	3	3	3	2	2	2	2	2	5
PF	Y	Y	Y	Y	-	Y	Y	Y	Y	Y	Y	-	-	Y	Y	-	-	Y	Y	Y

ii) LRU

Frame	6	0	5	2	0	3	0	4	2	3	0	3	2	5	2	0	5	6	0	5
0	6	6	6	2	2	2	2	4	4	4	0	0	0	5	5	5	5	5	5	5
1		0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
2			5	5	5	3	3	3	2	2	2	2	2	2	2	2	2	6	6	6
PF	Y	Y	Y	Y	-	Y	-	Y	Y	Y	Y	-	-	Y	-	Y	-	Y	-	-

iii) Optimal

Frame	6	0	5	2	0	3	0	4	2	3	0	3	2	5	2	0	5	6	0	5
0	6	6	6	2	2	2	2	2	2	2	2	2	2	2	2	2	2	6	6	6
1		0	0	0	0	0	0	4	4	4	0	0	0	0	0	0	0	0	0	0
2			5	5	5	3	3	3	3	3	3	3	3	5	5	5	5	5	5	5
PF	Y	Y	Y	Y	-	Y	-	Y	-	-	Y	-	-	Y	-	-	-	Y	-	-

b. Explain synchronization problem in detail. How counting semaphore can be used to solve readers writers problem. 10

➔ **Synchronization:** Process Synchronization is a sharing system resources by processes in such a way that, Concurrent access to shared data is handled thereby minimizing the chance of inconsistent data.

- Each process executes its own operations on shared variables sequentially, as specified by its own program. Nevertheless, different processes may execute their operations on the same shared variable concurrently. That is, operation

executions of different processes may overlap, and they may affect one another.

- Each operation on a shared variable, when executed indivisibly, transforms the variable from one consistent value to another. However, when the operations arc executed concurrently on a shared variable, the consistency of its values may not the guaranteed.
- The behaviours of operation executions on shared variables must, be predictable for effective inter-process communication.
- Thus, operation executions on shared variables may need to be coordinated to ensure their consistency semantics.
- Coordination of accesses to shared variables is called synchronization.
- A synchronization solution coordinates accesses from processes to shared variables. where all accesses to shared variables are channeled through access controllers
- The controllers do the coordination. Most operating systems implement a few different synchronization schemes for process coordination purposes. Each scheme supports a set of primitives. The primitives are used when it is absolutely necessary to have orderly executions of operations on shared variables in a particular manner.
- While defining the reader/writer's problem, It is assumed that, many processes only read the file(readers) and many write to the file(writers). File is shared among a many number of processes. The conditions that must be satisfied are as follows
  1) Simultaneously reading of the file is allowed to many readers.
  2) Writing to the file is allowed to only one writer at any given point of time.
  3) Readers are not allowed to read the file while writer is writing to the file.
- in this solution, the first reader accesses the file by performing a down operation on the semaphore file. Other readers only increment a counter, read count. When readers finish the reading counter is decremented.
- when last one end by performing an up on the semaphore, permitting a blocked writer, if there is one, to write. suppose that while a reader is reading a file, another reader comes along. Since having two readers at the same time is not a trouble, the second readers and later readers can also be allowed if they come.
- After this assumes that a writer wants to perform a write operation on the file. The writer can't be allowed to write the file, since writer is suspended. The writer will suspend until no reader is reading the file. If new reader arrives continuously with short interval and perform reading, the writer will never obtain the access to the file.

Algorithm

Writer process

```
While(TRUE)
{
Wait(w)
# perform write operation
Signal(w)
}
```

Reader Process

```
While(TRUE)
{
Wait(m)
Read_count ++;                    //reader enters
If (Read_count == 1)
Wait(w)                           //blocks writer
Signal(m)
# perform read operation
Wait(m)
Read_count - -;
If (Read_count == 0)              //No reader
Signal(w)                         //Writer can write
Signal(m)
}
```

Q-5

a. Given memory partitions of 150k,500k,200k,300k,550k(in order) how would each of the first fit, best fit and worst fit algorithm places the processes of 220k,430k,110k,425k(in order).Evaluate, which algorithm makes most efficient use of memory? 10

➔ **- First Fit**

- 220 K is put in 500 K partition.

- 430 K is put in 550 K partition.

- 110 K is put in 150 K partition.

- 425 K must wait.

- Best Fit

- 220 K is put in 300 K partition.

- 430 K is put in 500 K partition.

- 110 K is put in 150 K partition.

- 425 K is put in 550 K partition.

- Worst Fit

- 220 K is put in 550 K partition.

- 430 K is put in 500 K partition.

- 110 K is put in 330 K partition. (550 K – 220 K)

- 425 K must wait.

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

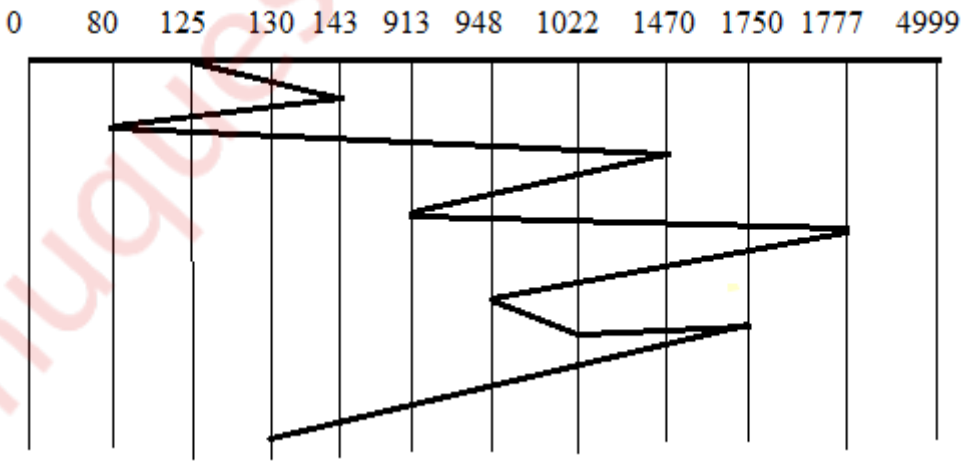
b. Suppose that a disk drive has 5000 cylinders, numbered 0 to 4999. The drive is currently serving a request at cylinder 143, and the previous request was at cylinder 125. The queue of pending requests in FIFO is ordered as 80, 1470, 913, 1777, 948, 1022, 1750,130. What is the total distance that the disk arm moves for following by applying following algorithms? 10

1. FCFS 2. SSTF 3. LOOK 4. SCAN

➔

1. FCFS: The FCFS schedule is

143, 80 1470, 913, 1777, 948, 1022, 1750, 130



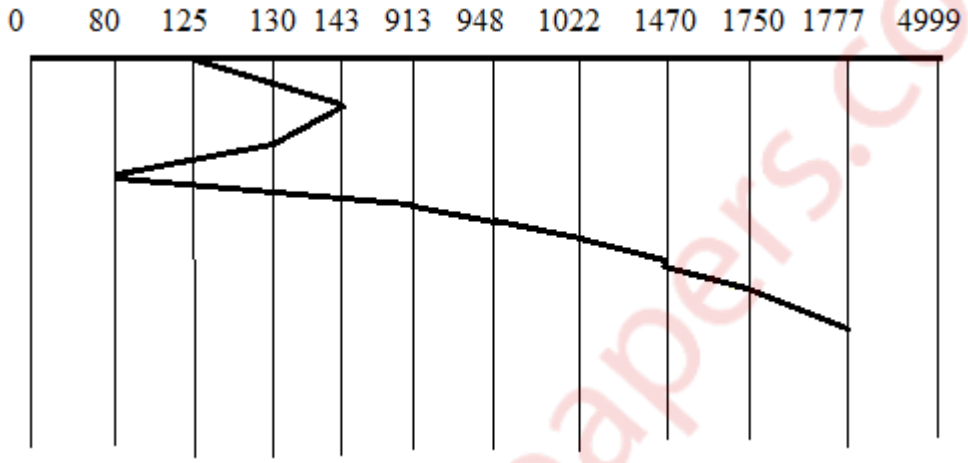
It gives (143-80) =63, (1470-80)=1390, (1470-913)=557, (1777-913)=864, (1777-948)=829, (1022-948)=74, (1750-1022)=728, (1750-80)=1670

Total head movements are:

63+1390+557+864+829+74+728+1670= 6125 Cylinders

2. SSTF: The SSTF schedule is

143, 130, 80, 913, 948, 1022, 1470, 1750, 1777.



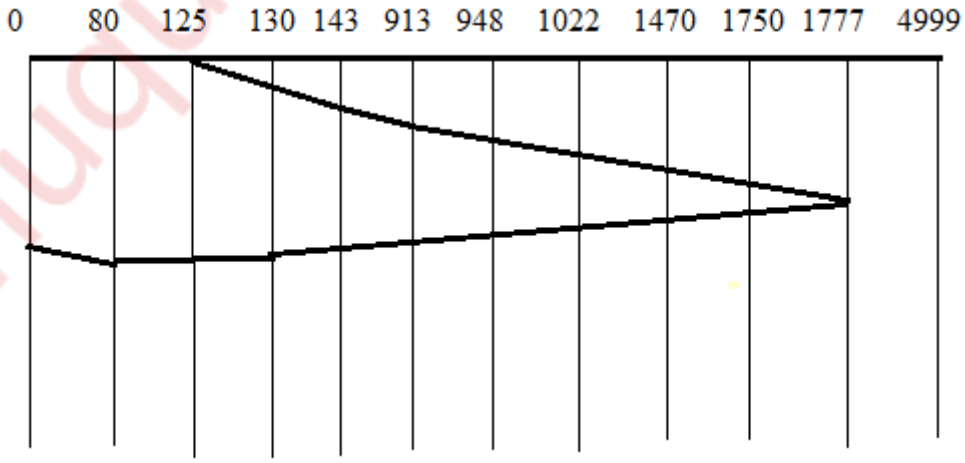
It gives (143-130) = 13, (130-80) = 50, (913-80)= 833, (948-913)=35, (1022-948)=74, (1470-1022)=448, (1750-1470)=280, (1777-1750)=27

Total head movements are:

13+50+833+35+74+448+280+27=1760 Cylinders

3. LOOK: The LOOK schedule is

143, 913, 948, 1022, 1470, 1750, 1777, 130, 80



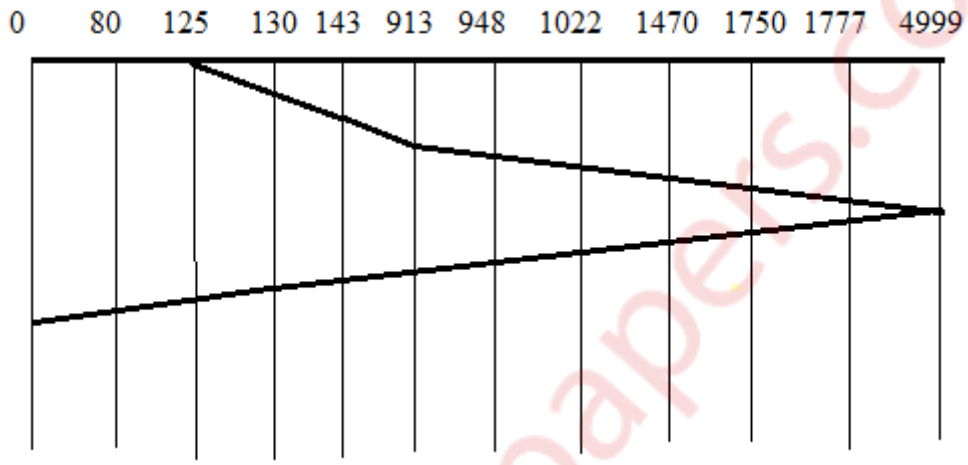
It gives (913-143)=770, (948-913)=35, (1022-948)=74, (1470-1022)=448**,** (1750-1470)=280, (1777-1750)= 27, (1777-130)=1647, (130-80)=47

Total head movements are:

770+35+74+448+280+27+1647+50=3331 Cylinders

4. SCAN: The SCAN schedule is

143, 913, 948, 1022, 1470, 1750, 1777, 4999, 130, 80.



It gives (913-143)=70, (948-913)=35, (1022-948)=74, (1470-1022)=448, (1750-1470)=280, (1777-1750)=27, (4999-1777)=3222, (4999-130)=4869, (130-80)=50

Total head movements are:

770+35+74+448+280+27+3222+4869+50 =9775 Cylinders

Q-6

Write short notes on: (any two) **20**

a. Linux Virtual File system

➔ The object oriented principle is used in Virtual File System (VFS).

- It has two modules: a set of definitions that states what file –system objects are permissible to seems to be and software layer for these objects manipulation.
- Following four major object types are defined by VFS:
    1) I-node Object – An individual file is represented by I-node Object.
    2) File Object – An open file is represented by file object.
    3) Superblock Object – An entire file system is represented by a Superblock Object.

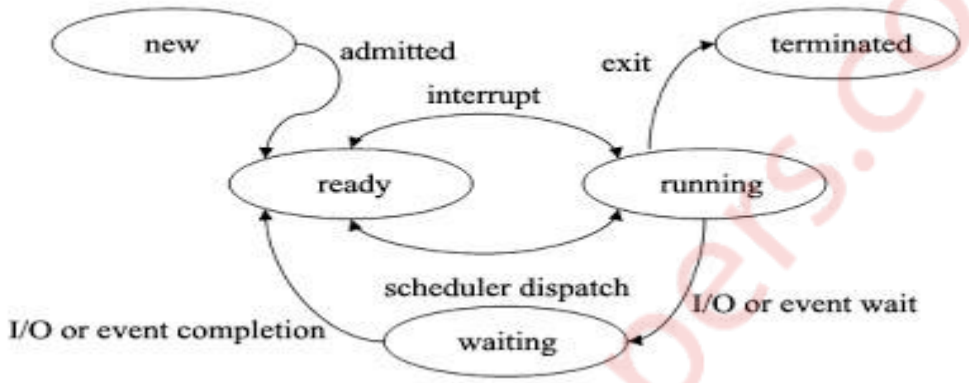
4) Dentry Object – An individual directory entry is represented by Dentry object.

- A set of operations are defined for each of the type of objects. Each object of one of these points to a function table.
- The record of addresses of the actual functions is kept in function table. These functions implement the defined set of operations for that object.
- The VFS software layer need not recognize earlier about what kind of object it is dealing with. It can carry out an operation on one of the file-system objects by invoking the right function from the object's function table.
- The VFS remains unaware about whether an i-node represents a networked file, a disk file, a network socket, or a directory file. The function table contains suitable function to read the file.
- The VFS software layer will invoke that function without worrying about the way of reading the data. The file can be accesses with the help of i-node and file objects.
- An i-node object is a data structure that holds pointers to the disk blocks. The actual file data is present on disk block.
- The file object denotes a point of access to the data in an open file. In order to access the i-node contents, the process first has to access a file object pointing to the i-node.
- The i-node objects do not belong to single process. Whereas file objects belong to single process.
- The i-node object is cached by the VFS to get better performance in near future access of the file. Therefore, although processes is not currently using the file, its i-node is cached by VFS.
- All cached file data are linked onto list in the file's i-node object. The i-node also keeps standard information about each file, for example the owner, size, and time most recently modified.
- Directory files are treated in a different way from other files.
- The programming interface of UNIX defines several operations on directories, for example creation, deletion, and renaming of a file in a directory.
- The system calls for these directory operations do not have need of the user open the files concerned, unlike the case for reading or writing data.
- Therefore, these directory operations are defined by VFS in the i-node object, instead of in the file object.
- The super block object represents files of entire file system.
- The operating system kernel keeps a single superblock object for each disk device mounted as a file system and each networked file system at present connected. The main duty of the superblock object is to offer access to i-nodes.
- The VFS recognize each i-node by a unique file-system / i-node number pair.

- It locates the i-node analogous to a particular i-node number by requesting the superblock object to return the i-node with that number.
- A entry object represents a directory entry that may include the name of a directory in the path name of a file (such as /usr) or the actual file.

b. Process State transition

➔



- Process can have one of the following five states at a time.

1. New state: A process that just has been created but has not yet been admitted to the pool of execution processes by the operating system. Every new operation which is requested to the system is known as the new born process.

2. Ready state: When the process is ready to execute but he is waiting for the CPU to execute then this is called as the ready state. After completion of the input and output the process will be on ready state means the process will wait for the processor to execute.

3. Running state: The process that is currently being executed. When the process is running under the CPU, or when the program is executed by the CPU, then this is called as the running process and when a process is running then this will also provide us some outputs on the screen.

4. Waiting or blocked state: A process that cannot execute until some event occurs or an I/O completion. When a process is waiting for some input and output operations then this is called as the waiting state and in this process is not under the execution instead the process is stored out of memory and when the user will provide the input and then this will again be on ready state.

5. Terminated state: After the completion of the process, the process will be automatically terminated by the CPU. So this is also called as the

terminated state of the process. After executing the complete process the processor will also deallocate the memory which is allocated to the process. So this is called as the terminated process.

c. System Calls

➔

1) The interface between OS and user programs is defined by the set of system calls that the operating system offers. System call is the call for the operating system to perform some task on behalf of the user's program. Therefore, system calls make up the interface between processes and the operating system.
2) The system calls are functions used in the kernel itself. From programmer's point view, the system call is a normal C function call.
3) Due to system call, the code is executed in the kernel so that there must be a mechanism to change the process mode from user mode to kernel mode.

System call is categorized in five groups.

Sr. No.	**Group**	**Examples**
1	Process control	End, abort, load, execute, create process, get process attributes, set process attributes, wait for time, wait event, allocate and free memory.
2	Device Manipulation	Request device, release device, read, write, reposition, get device attributes, set device attributes, logically attach or detach devices.
3	Communications	Create, delete communication connection, send, receive message, transfer status information, attach or detach devices.
4	File Manipulation	Create file, delete file, open, close, read, write, reposition, get file attributes, set file attributes.
5	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.

Some examples of System Calls

Open()	A program initializes access to a file in a file system using the open system call.
Read()	A program that needs to access data from a file stored in a file system uses the read system call.
Write()	It writes data from a buffer declared by the user to a given device, maybe a file. This is primary way to output data from a program by directly using a system call.
Exec()	exec is a functionality of an operating system that runs an executable file in the context of an already existing process, replacing the previous executable
Fork()	fork is an operation whereby a process creates a copy of itself. Fork is the primary method of process creation on Unix-like operating systems.