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.

Q.1. a) Explain the difference between monolithic and micro kernel 5

➔

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.

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.

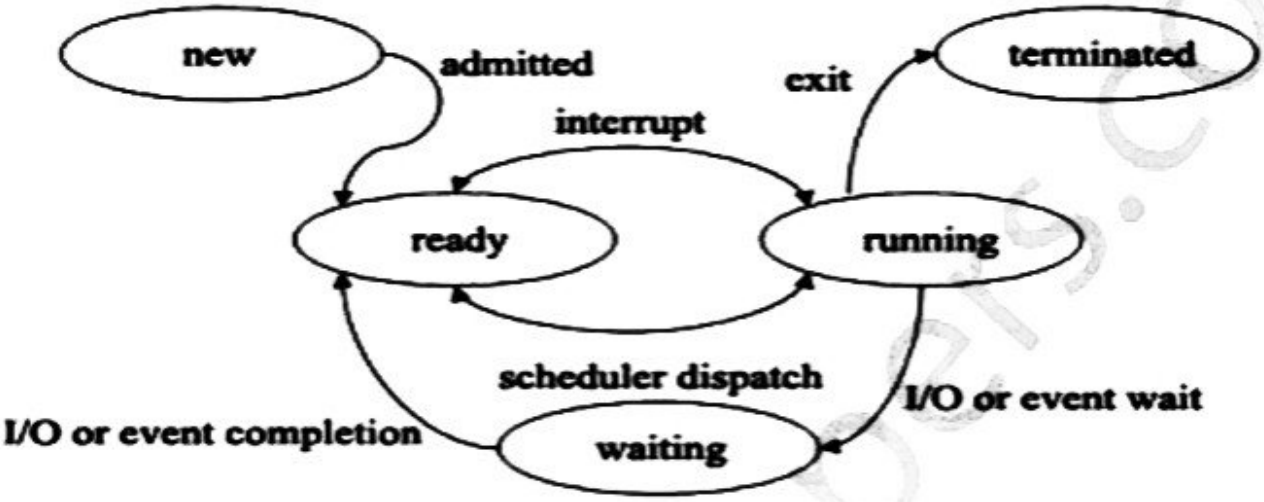
- chmod changes the permissions of each given file according to given mode, where the mode describes the permissions to modify. Mode can be specified with octal numbers or with letters.

- 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 .

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

c) Discuss various Scheduling Criteria. **5M**

➔ 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 and completion of its execution.
* Waiting Time – Processes waits in ready queue to get CPU. Sum of times spent In ready queues waiting time.
* Response Time – Time from submission till the first response is produced.
* Fairness – Every process should get fair share of the CPU time.

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.

b) What is deadlock? Explain the necessary and sufficient condition for deadlock. What is the difference between deadlock avoidance and prevention? 10M

➔ Deadlock:

* We know that processes need different resources in order to complete the execution.
* So in a multiprogramming environment, many processes may compete for a multiple Number of resources.
* In a system, resources are finite. So with finite number of resources, it is not possible to fulfill the resource request of all processes.
* When a process requests a resource and if the resource is not available at that time. The process enters a wait state. In multiprogramming environment, it may happen with many processes.
* There is chance that waiting processes will remain in same state and will never again change state.
* It is because the resource they have requested are held by other waiting processes. When such type of situation occurs then it is called as Deadlock.

Deadlock Avoidance	Deadlock Prevention
• The system dynamically considers every request and decides whether it is safe to grant it at this point. • Deadlock Avoidance Techniques: A deadlock avoidance algorithm dynamically examines the resource allocation state to ensure that a circular wait condition can never exist. The resource allocation state is defined by the number of available and allocated resources, and the maximum demand of the process. There are two algorithms: 1) Resource allocation graph. 2) Banker's algorithm.	• Preventing deadlock by constraining how requests for resources can be made in system and how they are handled. • Deadlock prevention Techniques: Mutual Exclusion ◊ this condition is needed to be checked for non-sharable resources (e.g. Printer) Hold and Wait ◊ It requires a process to request a resource and get allocated before execution or allow process to request resources when the process has none. No preemption ◊ If a process that is holding some resources requests another resource that cannot be immediately allocated to it, then all

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.

a) Explain different types of memory fragmentation. **8M**

- As processes are loaded and removed from memory, the free memory space is broken into little pieces.
- It happens after sometimes that processes cannot be allocated to memory blocks considering their small size and memory blocks remains unused. This problem is known as Fragmentation.

Fragmentation is of two types –

External Fragmentation:

- It exists when there us enough total memory space available to satisfy a request, but available memory space are not contiguous.
- Storage space is fragmented into large number of small holes.
- Both first fit and best fit strategies suffer from this.
- First fit is better in some systems, whereas best fit is better for other.
- Depending on the total amount of memory storage, size, external fragmentation may be minor or major problem.
- Statistically N allocated block, Another 0.5 N blocks will be lost to fragmentation. The 1/3 of memory is unusable.

Internal Fragmentation:

- Consider a multiple partition allocation scheme with a hole of 18,462 bytes. The next process request with 18,462 bytes. If we allocate, we are left with a hole of 2 bytes.
- The general approach to avoid this problem is to:
- Break physical memory into fixed sized blocks and allocate memory in units based on block size.
- Memory allocated to a process may be slightly large than the requested memory.

Solution to Internal Fragmentation

1) Compaction- The goal is to shuffle the memory content. so as to place all free memory together in one large block.

It is not always possible due to :-

If relocation is static and done at assembly or load time

It is possible

Only if relocation is dynamic and is done at execution time

2) Permit the logical address space to the processes to be non-contagious.

a) Discuss various File Allocation Mechanism and their advantages. **10M**

➔ Files are usually stored on secondary storage devices such as disk. These files are then called back when needed. As part of their implementation, files must be stored in the hard disk. This has to be done in such a way that the disk space is utilized effectively and the files can be accessed quickly. There are following methods used majority in different operating system:

Contiguous allocation
Linked List allocation
Linked List allocation using a table in memory.
Indexed allocation
I-nodes

1) **Contiguous Allocation:** Each file takes up a set of contiguous blocks on the disk. Disk address defines a linear ordering on the disk. If each block size on disk is 2 KB, then 30 KB file would be allocated 15 consecutive blocks. The directory entry for each file specifies the address of the starting block and the total number of blocks allocated for this file. Directory entry is shown below. File A starts at block 0 and it is 3 block long occupying block 0.

Advantage:
- Contiguous allocation is easy to implement.

Disadvantage:
- When allocated file is deleted, continuously allocated blocks to the file become free.

2) **Linked List Allocation:** This overcomes the disadvantage of contiguous allocation. In linked list allocation, each file is linked list of disk block. The scattered disk on the disk can be allocated to the file. The directory contains a pointer to the first and the last block. Below figure shows thee linked list allocation for file Pics. The file pics of 5 blocks starts at block 2 and continue 13, then block 18, then block 12, and finally block 7.

Advantage:
- Reading file sequentially is simple.

Disadvantage:
- In each block pointer takes some space, so each file requires slightly more disk space rather than its actual size.

3) **Linked List allocation using table in memory:** Each block needs to store pointer information; therefore, entire block is not fully used to store file content. This limitation can be overcome by keeping pointer information in table which always remains in memory. Refer Linked list fig for Table.

Advantage:

- Random access is much easier

Disadvantage:

- Whole table must be in memory all the time to make it work.

4) **Indexed allocation:** With file allocation table in memory, Linked list allocation support random access, but this entire table must be in memory all the time. In indexed allocation, all the pointers are kept in one location called as index block. There is an index block assigned to each file and this index block holds the disk block addresses of that particular file.

Advantage:

- Indexed allocation supports random access.

Disadvantage:

- The pointer overhead of index block is more with compare to the pointer overhead of linked allocation.

5) **I-nodes**: I-Nodes (Index Nodes) Is the data structure which records the attributes and disk addresses of the files blocks. I-nodes is associated with each file and it keeps track

of which block belongs to which file. If particular file is open, only its i-node to be in memory. This is more beneficial with compare to linked list allocation which requires entire file allocation table in memory. The size of file allocation table is proportional to the number of blocks that disk contains.

b) Discuss various disk scheduling methods. **10M**

➔

- In operating systems, seek time is very important. Since all device requests are linked in queues, the seek time is increased causing the system to slow down. Disk Scheduling Algorithms are used to reduce the total seek time of any request.
- **TYPES OF DISK SCHEDULING ALGORITHMS**

  1) First Come-First Serve (FCFS)

  2) Shortest Seek Time First (SSTF)

  3) Elevator (SCAN)

  4) Circular SCAN (C-SCAN)

  5) LOOK

  6) C-LOOK

- Given the following queue -- 95, 180, 34, 119, 11, 123, 62, 64 with the Read-write head initially at the track 50 and the tail track being at 199 let us now discuss the different algorithms.

1. First Come -First Serve (FCFS):

All incoming requests are placed at the end of the queue. Whatever number that is next in the queue will be the next number served. Using this algorithm doesn't provide the best results. To determine the number of head movements you would simply find the number of tracks it took to move from one request to the next. For this case it went from 50 to 95 to 180 and so on. From 50 to 95 it moved 45 tracks. If you tally up the total number of tracks, you will find how many tracks it had to go through before finishing the entire request. In this example, it had a total head movement of 640 tracks. The disadvantage of this algorithm is noted by the oscillation from track 50 to track 180 and then back to track 11 to 123 then to 64. As you will soon see, this is the worse algorithm that one can use.

2. Shortest Seek Time First (SSTF):

In this case request is serviced according to next shortest distance. Starting at 50, the next shortest distance would be 62 instead of 34 since it is only 12 tracks away from 62 and 16 tracks away from 34. The process would continue until all the process are taken care of. For example, the next case would be to move from 62 to 64 instead of 34 since there are only 2 tracks between them and not 18 if it were to go the other way. Although this seems to be a better service being that it moved a total of 236 tracks, this is not an optimal one. There is a great chance that starvation would take place. The reason for this is if there were a lot of requests close to each other the other requests will never be handled since the distance will always be greater.

3. Elevator (SCAN):

This approach works like an elevator does. It scans down towards the nearest end and then when it hits the bottom it scans up servicing the requests that it didn't get going down. If a request comes in after it has been scanned it will not be serviced until the process comes back down or moves back up. This process moved a total of 230 tracks. Once again this is more optimal than the previous algorithm, but it is not the best.

4. Circular Scan (C-SCAN)

Circular scanning works just like the elevator to some extent. It begins its scan toward the nearest end and works it way all the way to the end of the system. Once it hits the bottom or top it jumps to the other end and moves in the same direction. Keep in mind that the huge jump doesn't count as a head movement. The total head movement for this algorithm is only 187 tracks, but still this isn't the more sufficient.

5. LOOK

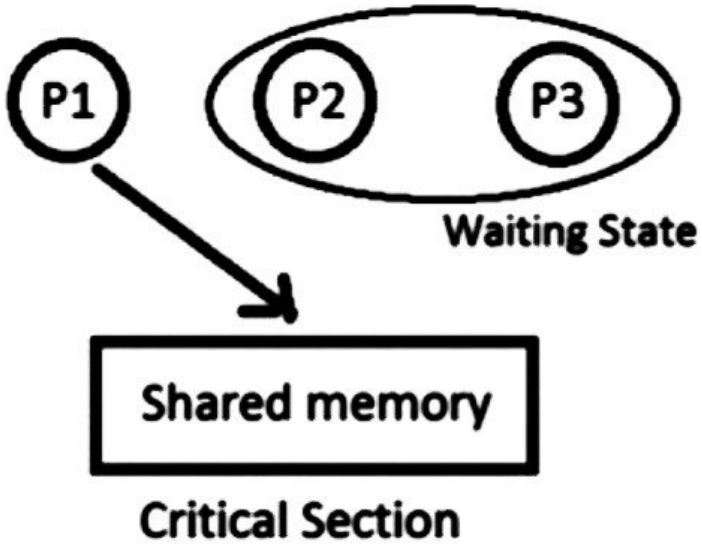
This is same as Scan scheduling but in this we do not move till end which reduce total head moment. This is the best scheduling algorithm because it has minimum total head
Moment.

6. C-LOOK

This is just an enhanced version of C-SCAN. In this the scanning doesn't go past the last request in the direction that it is moving. It too jumps to the other end but not all the way to the end. Just to the furthest request. C-SCAN had a total movement of 187 but this scan (C-LOOK) reduced it down to 157 tracks. From this you were able to see a scan change from 644 total head movements to just 157. You should now have an

b) What is Mutual Exclusion? Explain its Significance. **5M**

➔ At the same time as one processor executes the shared variable, all remaining processes wishing to accomplish so at the same instant should be kept waiting; when that process has over executing the shared variable, one of the processes waiting to perform so should be permitted to carry on. In this manner, each process executing the shared variable keep outs all others from doing so at the same time. Only one process should be allowed to execute in critical section. This is called as Mutual Exclusion.



Significance:

1) It avoids Race around condition.
2) Prevents multiple threads to enter critical section at the same time.