UNIT – 1

1 Explain the objectives and functions of operating systems?

Objectives of OS:

1.Convenience: An OS makes a computer more convenient to use.

2.Efficiency: An OS allows the computer system resources to be used in an efficient manner.

3.Ability to evolve: An OS should be constructed in such a way as to permit the effective development, testing, and introduction of new system functions without interfering with service.

Functions of an operating System are as follows:

Memory Management: Memory management refers to management of Primary Memory or Main Memory.An Operating System does the following activities for memory management: OS Keeps tracks of primary memory, i.e., what part of it are in use by whom, what part are not in use.In multi-programming, the OS decides which process will get memory when and how much.OS allocates the memory when a process requests it to do so. It de-allocates the memory when a process no longer needs it or has been terminated.

Processor Management: In multi-programming environment, the OS decides which process gets the processor when and for how much time. This function is called process scheduling. An Operating System does the following activities for processor management: OS keeps tracks of processor and status of process.OS allocates the processor (CPU) to a process.It de-allocates processor when a process is no longer required.

Device Management: An Operating System manages device communication via their respective drivers. It does the following activities for device management: Keeps tracks of all devices. The program responsible for this task is known as the I/O controller. Decides which process gets the device when and for how much time. OS allocates the device in the most efficient way.It de-allocates devices in most efficient way.

File Management: A file system is normally organized into directories for easy navigation and usage. These directories may contain files and other directions.An Operating System does the following activities for file management: Keeps track of information, location, uses, status etc. The collective facilities are often known as file system. OS Decides who gets the resources.It allocates the resources and also de-allocates the resources when not in need.

Security: OS prevents unauthorized access to programs and data.For shared or public systems, the OS controls access to the system as a whole and to specific system resources.

Control over system performance: OS will collect usage statistics for various resources and monitor performance parameters such as response time,Recording delays between request for a service and response from the system.

2 Explain basic services provided by Operating system on bare Hardware machine?

So basically Bare machine is logical hardware which is used to execute the program in the processor without using the operating system. as of now, we have studied that we can't execute any process without the Operating system. But yes with the help of the Bare machine we can do that.

Initially, when the operating systems are not developed, the execution of an instruction is done by directly on hardware without using any interfering hardware, at that time the only drawback was that the Bare machines accepting the instruction in only machine language, due to this those person who has sufficient knowledge about Computer field are able to operate a computer. so after the development of the operating system Bare machine is referred to as inefficient.

3 What is System Call in OS? Explain fork () system call in UNIX OS?

A **system call** is a mechanism that provides the interface between a process and the operating system. It is a programmatic method in which a computer program requests a service from the kernel of the OS.

fork()

Processes use this system call to create processes that are a copy of themselves. With the help of this system Call parent process creates a child process, and the execution of the parent process will be suspended till the child process executes.

4 List the types of operating systems and explain any one in detail?

- Batch Operating System
- Multitasking/Time Sharing OS
- Multiprocessing OS
- Real Time OS
- Distributed OS
- Network OS
- Mobile OS

Batch Operating System

Some computer processes are very lengthy and time-consuming. To speed the same process, a job with a similar type of needs are batched together and run as a group.

The user of a batch operating system never directly interacts with the computer. In this type of OS, every user prepares his or her job on an offline device like a punch card and submit it to the computer operator.

5 Write a note on Distributed Operating System. What is Operating System? Give functions of Operating System.

A distributed operating system is **an extension of the network operating system that supports higher levels of communication and integration of the machines on the network**. This system looks to its users like an ordinary centralized operating system but runs on multiple, independent Central Processing Units (CPUs).

An **Operating System (OS)** is a software that acts as an interface between computer hardware components and the user. Every computer system must have at least one operating system to run other programs. Applications like Browsers, MS Office, Notepad Games, etc., need some environment to run and perform its tasks. (For functions que no.1)

6 Give the features of Real Time Operating System and Time-Sharing Operating System.

S.NO	Time Sharing Operating System	Real-Time Operating System
1.	In time sharing operating system, quick response is emphasized for a request.	While in real time operating system, computation tasks are emphasized before its nominative point.
2.	In this operating system **Switching** method/function is available.	While in this operating system **Switching** method/function is not available.
3.	In this operating system any modification in the program can be possible.	While in this modification does not take place.
4.	In this OS, computer resources are shared to the external.	But in this OS, computer resources are not shared to the external.

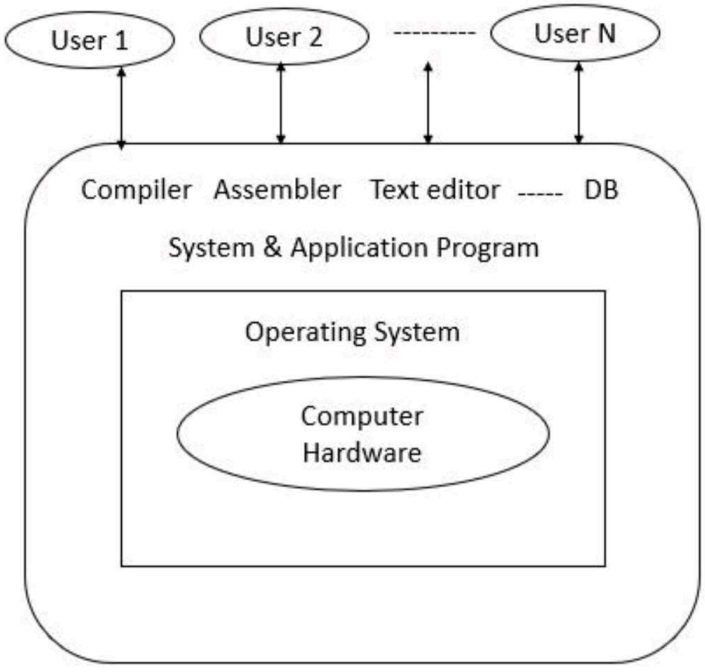
S.NO	Time Sharing Operating System	Real-Time Operating System
5.	It deals with more than processes or applications simultaneously.	Whereas it deals with only one process or application at a time.
6.	In this OS, the response is provided to the user within a second.	While in real time OS, the response is provided to the user within time constraint.
7.	In time sharing system, high priority tasks can be preempted by lower priority tasks, making it impossible to guarantee a response time for your critical applications.	Real time operating systems, give users the ability to prioritize tasks so that the most critical task can always take control of the process when needed.

7 What is an operating System? Explain the abstract view of the components of a computer system.

Os definition question – 5

Abstract View

The abstract view of components of computer system is as follows −



Components of Computer System

A computer system can be divided into four components, which are as follows −

Hardware − The hardware is the physical part which we can touch and feel, the central processing unit (CPU), the memory, and the input/output (I/O) devices are the basic computing resources of a computer system.

Application programs − Application programs are user or programmer created programs like compilers, database systems, games, and business programs that define the ways in which these resources can be used to solve the computing problems of the users.

Users − There are different types of users like people, machines, and even other computers which are trying to solve different problems.

Operating system − An operating system is the interface between the user and the machine which controls and coordinates the use of the hardware among the various application programs for the various users.

8 Explain different types of OS and also Explain different types of tasks done by OS

Types of Operating System (OS)

Following are the popular types of OS (Operating System):

- Batch Operating System
- Multitasking/Time Sharing OS
- Multiprocessing OS
- Real Time OS
- Distributed OS
- Network OS
- Mobile OS

9 What is system call? What is interrupt? How it is handled by OS?

An interrupt is a signal emitted by hardware or software when a process or an event needs immediate attention. It alerts the processor to a high-priority process requiring interruption of the current working process. In I/O devices, one of the bus control lines is dedicated for this purpose and is called the ***Interrupt Service Routine*** (ISR).

When a device raises an interrupt at the process, the processor first completes the execution of an instruction. Then it loads the ***Program Counter*** (PC) with the address of the first instruction of the ISR. Before loading the program counter with the address,

the address of the interrupted instruction is moved to a temporary location. Therefore, after handling the interrupt, the processor can continue with the process.

While the processor is handling the interrupts, it must inform the device that its request has been recognized to stop sending the interrupt request signal. Also, saving the registers so that the interrupted process can be restored in the future increases the delay between the time an interrupt is received and the start of the execution of the ISR. This is called ***Interrupt Latency***.

10 Give the features of Batch Operating System.

Some computer processes are very lengthy and time-consuming. To speed the same process, a job with a similar type of needs are batched together and run as a group.

The user of a batch operating system never directly interacts with the computer. In this type of OS, every user prepares his or her job on an offline device like a punch card and submit it to the computer operator.

11 Give the advantages of Distributed Operating System

Advantages of Distributed Operating System:

- Failure of one will not affect the other network communication, as all systems are independent from each other
- Electronic mail increases the data exchange speed
- Since resources are being shared, computation is highly fast and durable
- Load on host computer reduces
- These systems are easily scalable as many systems can be easily added to the network
- Delay in data processing reduces

12 Write short notes on following: Real Time Operating Systemns

Answer – (Que-6)

13 Explain Goals of I/O Software

Uniform naming: For example naming of files systems in Operating Systems is done in a way that user does not have to be aware of underlying hardware name.
Synchronous versus Asynchronous: When the CPU is working on some process it goes in the block state when the interrupt occurs. Therefore most of the devices are asynchronous. And if the I/O operation are in blocking state it is much easier to write the I/O operation. It is always the operating system responsibility to create such a interrupt driven user program.

Device Independence: The most important part of I/O software is device independence. It is always most preferable to write program which can open all other I/O devices. For example, it is not necessary to write the input taking program again and again for taking input from various file and devices. As this creates much work to do and also much space to store the different programs.
Buffering: Data that we enter into a system cannot be stored directly in memory. For example the data is converted into smaller groups and then transferred to outer buffer for examination.
Buffer have major impact on I/O software as it is the one which ultimately helps storing the data and copying data. Many device have constraints and just to avoid it some data is always put into the buffer in advance so the buffer rate of getting filled with data and getting empty remains balanced
Error handling: Errors and mostly generated by controller and also they are mostly handled by controller itself. When lower level solves the problem it does not reach the upper level.
Shareable and Non-Shareable Devices : Devices like Hard Disk can be shared among multiple process while devices like Printers cannot be shared. The goal of I/O software is to handle both types of devices.

UNIT – 2

1 Differentiate process and thread. How do you create and terminate the process? Draw a diagram which indicates the state of processes.

S.NO	Process	Thread
1.	Process means any program is in execution.	Thread means segment of a process.
2.	Process takes more time to terminate.	Thread takes less time to terminate.
3.	It takes more time for creation.	It takes less time for creation.
4.	It also takes more time for context switching.	It takes less time for context switching.
5.	Process is less efficient in term of communication.	Thread is more efficient in term of communication.
6.	Multi programming holds the concepts of multi process.	We don't need multi programs in action for multiple threads because a single process consists of multiple threads.
7.	Process is isolated.	Threads share memory.
8.	Process is called heavy weight process.	A Thread is lightweight as each thread in a process shares code, data and resources.
9.	Process switching uses interface in operating system.	Thread switching does not require to call a operating system and cause an interrupt to the kernel.
10.	If one process is blocked then it will not effect the execution of other process	Second thread in the same task could not run, while one server thread is blocked.

S.NO	Process	Thread
11.	Process has its own Process Control Block, Stack and Address Space.	Thread has Parents' PCB, its own Thread Control Block and Stack and common Address space.
12.	If one process is blocked, then no other process can execute until the first process is unblocked.	While one thread is blocked and waiting, a second thread in the same task can run.
13.	Changes to the parent process does not affect child processes.	Since all threads of the same process share address space and other resources so any changes to the main thread may affect the behavior of the other threads of the process.

4 Give the difference between a Process and a Program

Parameter	Process	Program
Definition	An executing part of a program is called a process.	A program is a group of ordered operations to achieve a programming goal.
Nature	The process is an instance of the program being executing.	The nature of the program is passive, so it's unlikely to do to anything until it gets executed.
Resource management	The resource requirement is quite high in case of a process.	The program only needs memory for storage.
Overheads	Processes have considerable overhead.	No significant overhead cost.
Lifespan	The process has a shorter and very limited lifespan as it gets terminated after the	A program has a longer lifespan as it is stored in the memory until it is not manually deleted.

Parameter	Process	Program
	completion of the task.	
Creation	New processes require duplication of the parent process.	No such duplication is needed.
Required Process	Process holds resources like CPU, memory address, disk, I/O, etc.	The program is stored on disk in some file and does not require any other resources.
Entity type	A process is a dynamic or active entity.	A program is a passive or static entity.
Contain	A process contains many resources like a memory address, disk, printer, etc.	A program needs memory space on disk to store all instructions.

5 Write different operating system services Explain multiprocessor operating system types in brief.

An Operating System provides services to both the users and to the programs.

- It provides programs an environment to execute.
- It provides users the services to execute the programs in a convenient manner.

Following are a few common services provided by an operating system −

- Program execution
- I/O operations
- File System manipulation
- Communication
- Error Detection

- Resource Allocation
- Protection

Multiprocessing Systems

A computer's capability to process more than one task simultaneously is called *multiprocessing*. A multiprocessing operating system is capable of running many programs simultaneously, and most modern network operating systems (NOSs) support multiprocessing. These operating systems include Windows NT, 2000, XP, and Unix.

Although Unix is one of the most widely used multiprocessing systems, there are others. For many years, OS/2 has been the choice for high-end workstations. OS/2 has been a standard operating system for businesses running complex computer programs from IBM. It is a powerful system, employs a nice graphical interface, and can also run programs written for DOS and Windows. However, OS/2 never really caught on for PCs.

The main reason why multiprocessing is more complicated than single-processing is that their operating systems are responsible for allocating resources to competing processes in a controlled environment.

With the growth of commercial networks, the practice of using multiple processors in embedded motherboard designs has become almost universal. Not too long ago, clients or network administrators constructed most multiprocessing configurations at the board or system level themselves. Today, motherboards are available incorporating multiple microprocessors on the same die.

A multiprocessing system uses more than one processor to process any given workload, increasing the performance of a system's application environment beyond that of a single processor's capability. This permits tuning of the server network's performance, to yield the required functionality. As described in Chapter 2, "Server Availability," this feature is known as *scalability*, and is the most important aspect of multiprocessing system architectures. Scalable system architecture allows network administrators to tune a server network's performance based on the number of processing nodes required.

Collections of processors arranged in a loosely coupled configuration and interacting with each other over a communication channel have been the most common multiprocessor architecture.

This communication channel might not necessarily consist of a conventional serial or parallel arrangement. Instead, it can be composed of shared memory, used by processors on the same board, or even over a backplane. These interacting processors operate as independent nodes, using their own memory subsystems.

Recently, the embedded server board space has been arranged to accommodate tightly coupled processors, either as a pair, or as an array. These processors share a common bus and addressable

memory space. A switch connects them, and interprocessor communications is accomplished through message passing. In the overall system configuration, the processors operate as a single node, and appear as a single processing element. Additional loosely coupled processing nodes increase the overall processing power of the system. When more tightly coupled processors are added, the overall processing power of a single node increases.

These processors have undergone many stages of refinement over the years. For example, the Xeon processors were designed for either network servers or high-end workstations. Similarly, Pentium 4 microprocessors were intended solely for desktop deployment, although Xeon chips had also been called "Pentiums" to denote their family ancestry. Pentium and Xeon processors are currently named separately. The Xeon family consists of two main branches: the Xeon dual-processor (DP) chip, and the Xeon multiprocessor (MP).

6 Differentiate Multiprogramming, Multitasking, Multiprocessing & Distributed Operating System.

1. Multi programming –
In a modern computing system, there are usually several concurrent application processes which want to execute. Now it is the responsibility of the Operating System to manage all the processes effectively and efficiently. One of the most important aspects of an Operating System is to multi program.
In a computer system, there are multiple processes waiting to be executed, i.e. they are waiting when the CPU will be allocated to them and they begin their execution. These processes are also known as jobs. Now the main memory is too small to accommodate all of these processes or jobs into it. Thus, these processes are initially kept in an area called job pool. This job pool consists of all those processes awaiting allocation of main memory and CPU.
CPU selects one job out of all these waiting jobs, brings it from the job pool to main memory and starts executing it. The processor executes one job until it is interrupted by some external factor or it goes for an I/O task.

Multitasking, in an operating system, is allowing a user to perform more than one computer task (such as the operation of an application program) at a time. The operating system is able to keep track of where you are in these tasks and go from one to the other without losing information. Microsoft Windows 2000, IBM's OS/390, and Linux are examples of operating systems that can do multitasking (almost all of today's operating systems can). When you open your Web browser and then open Word at the same time, you are causing the operating system to do multitasking.

Being able to do multitasking doesn't mean that an unlimited number of tasks can be juggled at the same time. Each task consumes system storage and other resources. As more tasks are started, the system may slow down or begin to run out of shared storage.

It is easy to confuse multitasking with Multithreading, a somewhat different idea.

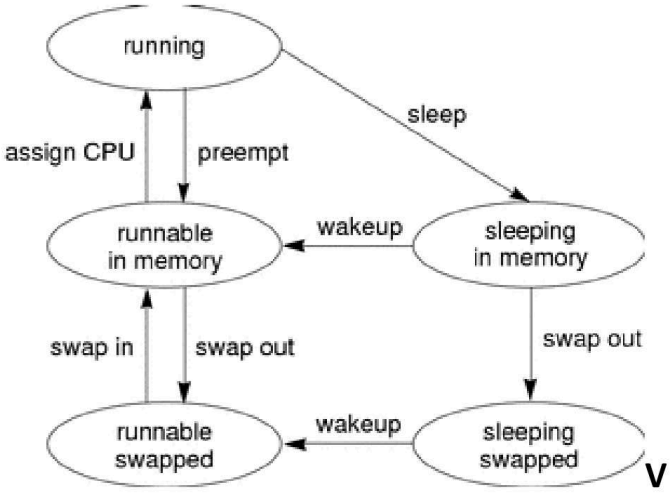
2. Multiprocessing –
In a uni-processor system, only one process executes at a time. Multiprocessing is the use of two or more CPUs (processors) within a single Computer system. The term also refers to the ability of a system to support more than one processor within a single computer system. Now since there are multiple processors available, multiple processes can be executed at a time. These multi processors share the computer bus, sometimes the clock, memory and peripheral devices also.

7 Define a process. Explain the process state transition with a neat diagram.

AnS – PROCESS(QUE-4)

Process State Transition

Applications that have strict real-time constraints might need to prevent processes from being swapped or paged out to secondary memory. A simplified overview of UNIX process states and the transitions between states is shown in the following figure.



V

An active process is normally in one of the five states in the diagram. The arrows show how the process changes states.

- A process is running if the process is assigned to a CPU. A process is removed from the running state by the scheduler if a process with a higher priority becomes runnable. A process is also pre-empted if a process of equal priority is runnable when the original process consumes its entire time slice.
- A process is runnable in memory if the process is in primary memory and ready to run, but is not assigned to a CPU.
- A process is sleeping in memory if the process is in primary memory but is waiting for a specific event before continuing execution. For example, a process sleeps while waiting for an I/O operation to complete, for a locked resource to be unlocked, or for a timer to expire. When the event occurs, a wakeup call is sent to the process. If the reason for its sleep is gone, the process becomes runnable.
- When a process' address space has been written to secondary memory, and that process is not waiting for a specific event, the process is runnable and swapped.
- If a process is waiting for a specific event and has had its whole address space written to secondary memory, the process is sleeping and swapped.

  If a machine does not have enough primary memory to hold all its active processes, that machine must page or swap some address space to secondary memory.

- When the system is short of primary memory, the system writes individual pages of some processes to secondary memory but leaves those processes runnable. When a running process, accesses those pages, the process sleeps while the pages are read back into primary memory.
- When the system encounters a more serious shortage of primary memory, the system writes all the pages of some processes to secondary memory. The system marks the pages that have been written to secondary memory as swapped. Such processes can only be scheduled when the system scheduler daemon selects these processes to be read back into memory.

Both paging and swapping cause delay when a process is ready to run again. For processes that have strict timing requirements, this delay can be unacceptable.

To avoid swapping delays, real-time processes are never swapped, though parts of such processes can be paged. A program can prevent paging and swapping by locking its text and data into primary memory. For more information, see the memcntl(2) man page. How much memory can be locked is limited by how much memory is configured. Also, locking too much can cause intolerable delays to processes that do not have their text and data locked into memory.

8 What is thread and what are the differences between user level threads and kernel supported threads? Under what circumstances is one type "better" than the other?

What is a Thread?

A thread is a path of execution within a process. A process can contain multiple threads.

User level thread	Kernel level thread
User thread are implemented by users.	kernel threads are implemented by OS.
OS doesn't recognize user level threads.	Kernel threads are recognized by OS.
Implementation of User threads is easy.	Implementation of Kernel thread is complicated.
Context switch time is less.	Context switch time is more.
Context switch requires no hardware support.	Hardware support is needed.
If one user level thread perform blocking operation then entire process will be blocked.	If one kernel thread perform blocking operation then another thread can continue execution.
User level threads are designed as dependent threads.	Kernel level threads are designed as independent threads.
Example : Java thread, POSIX threads.	Example : Window Solaris.

Under what circumstances is one type better than the other? Answer: a. **User-level threads are unknown by the kernel, whereas the kernel is aware of kernel threads**.

11 Define Process. List the major events for creation of a process and explain them

Process

A process is basically a program in execution. The execution of a process must progress in a sequential fashion.

A process is defined as an entity which represents the basic unit of work to be implemented in the system.

To put it in simple terms, we write our computer programs in a text file and when we execute this program, it becomes a process which performs all the tasks mentioned in the program.

When a program is loaded into the memory and it becomes a process, it can be divided into four sections — stack, heap, text and data. The following image shows a simplified layout of a process inside main memory −

There are four principal events that cause processes to be created they are system initialization, execution of a process creation system call by a running process, a user request to create a new process, and initiation of a batch job

12 What is PCB? Discuss its major fields.

Process Control Block (PCB) is a data structure in the operating system kernel containing the information needed to manage a particular process.

Process State

This specifies the process state i.e. new, ready, running, waiting or terminated.

Process Number

This shows the number of the particular process.

Program Counter

This contains the address of the next instruction that needs to be executed in the process.

Registers

This specifies the registers that are used by the process. They may include accumulators, index registers, stack pointers, general purpose registers etc.

List of Open Files

These are the different files that are associated with the process

CPU Scheduling Information

The process priority, pointers to scheduling queues etc. is the CPU scheduling information that is contained in the PCB. This may also include any other scheduling parameters.

Memory Management Information

The memory management information includes the page tables or the segment tables depending on the memory system used. It also contains the value of the base registers, limit registers etc.

I/O Status Information

This information includes the list of I/O devices used by the process, the list of files etc.

Accounting information

The time limits, account numbers, amount of CPU used, process numbers etc. are all a part of the PCB accounting information.

Location of the Process Control Block

The process control block is kept in a memory area that is protected from the normal user access. This is done because it contains important process information. Some of

the operating systems place the PCB at the beginning of the kernel stack for the process as it is a safe location.

15 What is process? What are the different types of states Of any process? Explain different data structures to handle process management.

Ans – (Que-11)

What are the different states of a Process?

- New. This is the state when the process has just been created. ...
- Ready. In the ready state, the process is waiting to be assigned the processor by the short term scheduler, so it can run. ...
- Ready Suspended. ...
- Running. ...
- Blocked. ...
- Blocked Suspended. ...
- Terminated.

New

This is the state when the process has just been created. It is the initial state in the process life cycle.

Ready

In the ready state, the process is waiting to be assigned the processor by the short term scheduler, so it can run. This state is immediately after the new state for the process.

Ready Suspended

The processes in ready suspended state are in secondary memory. They were initially in the ready state in main memory but lack of memory forced them to be suspended and gets placed in the secondary memory.

Running

The process is said to be in running state when the process instructions are being executed by the processor. This is done once the process is assigned to the processor using the short-term scheduler.

Blocked

The process is in blocked state if it is waiting for some event to occur. This event may be I/O as the I/O events are executed in the main memory and don't require the processor. After the event is complete, the process again goes to ready state.

Blocked Suspended

This is similar to ready suspended. The processes in blocked suspended state are in secondary memory. They were initially in the blocked state in main memory waiting

for some event but lack of memory forced them to be suspended and gets placed in the secondary memory. A process may go from blocked suspended to ready suspended if its work is done.

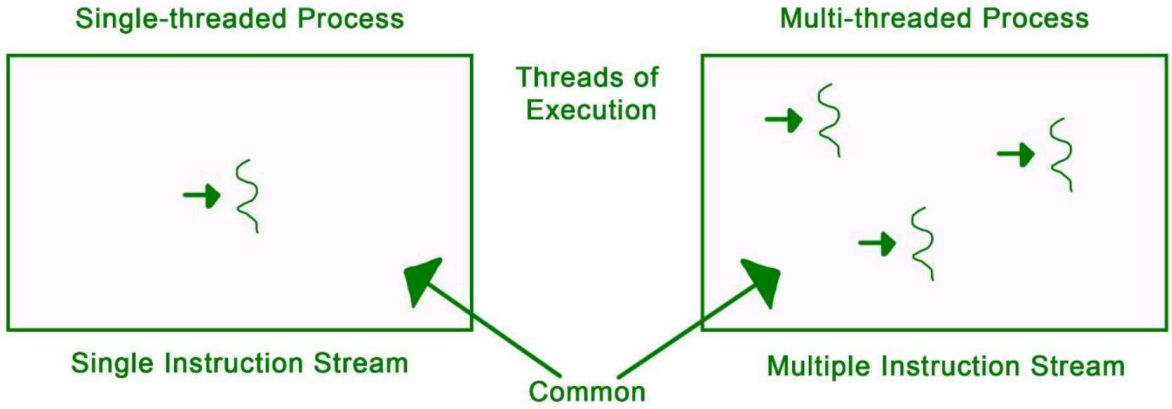
Terminated

The process is terminated once it finishes its execution. In the terminated state, the process is removed from main memory and its process control block is also deleted.

16 Write short notes on following: (i)Multithreading

A **thread** is a path which is followed during a program's execution. Majority of programs written now a days run as a single thread.Lets say, for example a program is not capable of reading keystrokes while making drawings. These tasks cannot be executed by the program at the same time. This problem can be solved through multitasking so that two or more tasks can be executed simultaneously.
Multitasking is of two types: Processor based and thread based. Processor based multitasking is totally managed by the OS, however multitasking through multithreading can be controlled by the programmer to some extent.

The concept of **multi-threading** needs proper understanding of these two terms – **a process and a thread**. A process is a program being executed. A process can be further divided into independent units known as threads.
A thread is like a small light-weight process within a process. Or we can say a collection of threads is what is known as a process.



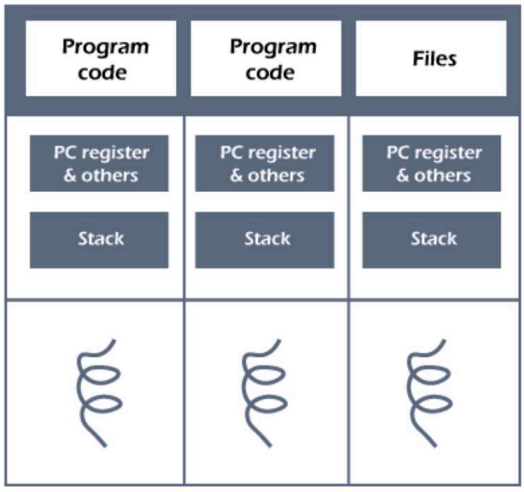
Single Thread and Multi Thread Process

Applications –
Threading is used widely in almost every field. Most widely it is seen over the internet now days where we are using transaction processing of every type like recharges, online transfer, banking etc. Threading is a segment which divide the code into small parts that are of very light weight and has less burden on CPU memory so that it can be easily worked out and can achieve goal in desired field. The concept of threading is designed due to the problem of fast and regular changes in technology and less the work in

different areas due to less application. Then as says “need is the generation of creation or innovation” hence by following this approach human mind develop the concept of thread to enhance the capability of programming.

17 What is thread? Explain thread structure.

A thread is a single sequential flow of execution of tasks of a process so it is also known as thread of execution or thread of control. There is a way of thread execution inside the process of any operating system. Apart from this, there can be more than one thread inside a process. Each thread of the same process makes use of a separate program counter and a stack of activation records and control blocks. Thread is often referred to as a lightweight process.



Three threads of same process

The process can be split down into so many threads. **For example**, in a browser, many tabs can be viewed as threads. MS Word uses many threads - formatting text from one thread, processing input from another thread, etc.

Need of Thread:

- It takes far less time to create a new thread in an existing process than to create a new process.
- Threads can share the common data, they do not need to use Inter- Process communication.
- Context switching is faster when working with threads.
- It takes less time to terminate a thread than a process.

Types of Threads

In the operating system, there are two types of threads.

1. Kernel level thread.
2. User-level thread.

Benefits of Threads

- **Enhanced throughput of the system:** When the process is split into many threads, and each thread is treated as a job, the number of jobs done in the unit time increases. That is why the throughput of the system also increases.
- **Effective Utilization of Multiprocessor system:** When you have more than one thread in one process, you can schedule more than one thread in more than one processor.
- **Faster context switch:** The context switching period between threads is less than the process context switching. The process context switch means more overhead for the CPU.
- **Responsiveness:** When the process is split into several threads, and when a thread completes its execution, that process can be responded to as soon as possible.
- **Communication:** Multiple-thread communication is simple because the threads share the same address space, while in process, we adopt just a few exclusive communication strategies for communication between two processes.
- **Resource sharing:** Resources can be shared between all threads within a process, such as code, data, and files. Note: The stack and register cannot be shared between threads. There is a stack and register for each thread.

18 Explain PCB. – Ans(Que-12)

19 What is a scheduler? Explain queuing diagram representation of process scheduler with figure.

Scheduler are the process which decides which task and process should be accessed and run at what time by the system resources.

It is required to maintain

20 Define a distributed system. Explain the characteristics of distributed system.

A *distributed system* is a system in which components are located on different *networked computers*, which
can *communicate* and *coordinate* their actions by passing messages to one another. The components interact with one another in order to achieve a common goal.

Key characteristics of distributed systems are

- **Resource Sharing**

Resource sharing means that the existing *resources* in a distributed system can be accessed or remotely accessed across multiple computers in the system. Computers in distributed systems shares resources like *hardware* (disks and printers), *software* (files, windows and data objects) and *data*. Hardware resources are shared for reductions in cost and convenience. Data is shared for consistency and exchange of information.

Resources are managed by a software module known as a resource manager. Every resource has its own management policies and methods.

- **Heterogeneity**

In distributed systems components can have variety and differences in Networks, Computer hardware, Operating systems, Programming languages and implementations by different developers.

- **Openness**

Openness is concerned with extensions and improvements of distributed systems. The distributed system must be open in terms of *Hardware* and *Softwares*. In order to make a distributed system open,

1. A detailed and well-defined interface of components must be published.

2. Should standardize the interfaces of components

3. The new component must be easily integrated with existing components

- **Concurrency**

Concurrency is a property of a system representing the fact that multiple activities are executed at the same time. The concurrent execution of activities takes place in different components running on multiple machines as part of a distributed system. In addition, these activities may perform some kind of interactions among them. Concurrency *reduces the latency* and *increases the throughput* of the distributed system.

- **Scalability**

Scalability is mainly concerned about how the distributed system handles the *growth* as the number of users for the system increases. Mostly we scale the distributed system by adding more computers in the network. Components should not need to be changed when we

scale the system. Components should be designed in such a way that it is scalable.

- **Fault Tolerance**

In a distributed system hardware, software, network anything can fail. The system must be designed in such a way that it is available all the time even after something has failed.

- **Transparency**

Distributed systems should be perceived by users and application programmers as a whole rather than as a collection of cooperating components. Transparency can be of various types like access, location, concurrency, replication, etc.

21 Explain "5 State" Process Transition Diagram with illustration

A process is a program under execution that consists of a number of elements including, program code and a set of data. To execute a program, a process has to be created for that program. Here the process may or may not run but if it is in a condition of running then that has to be maintained by the OS for appropriate progress of the process to be gained.

Need for Five-State Process Model

In five- state model the states have been split into two non-running states: *ready* and *blocked*, and along with this, two more states are added for the purpose: ***New*** and ***Exit/Terminate***. These two states are used because, in the previous models, the main memory is capable to store all programs but that is not true because the programs are now very large, and loading those processes in the main memory is very tough/ or even not possible and also if there's a requirement for using the previous resources that are released by the process, then that is not possible here.
The five states that are being used in this process model are:

1. **Running:** It means a process that is currently being executed. Assuming that there is only a single processor in the below execution

process, so there will be at most one processor at a time that can be running in the state.
2. **Ready:** It means a process that is prepared to execute when given the opportunity by the OS.
3. **Blocked/Waiting:** It means that a process cannot continue executing until some event occurs like for example, the completion of an input-output operation.
4. **New:** It means a new process that has been created but has not yet been admitted by the OS for its execution. A new process is not loaded into the main memory, but its process control block (PCB) has been created.
5. **Exit/Terminate:** A process or job that has been released by the OS, either because it is completed or is aborted for some issue.

Execution of Process in Two-state Model

This model consists of five states i.e, running, ready, blocked, new, and exit. The model works when any new job/process occurs in the queue, it is first admitted in the queue after that it goes in the ready state. Now in the Ready state, the process goes in the running state. In the running state, a process has two conditions i.e., either the process goes to the event wait or the process gets a time-out.

If the process has timed out, then the process again goes to the ready state as the process has not completed its execution. If a process has an event wait condition then the process goes to the blocked state and after that to the ready state. If both conditions are true, then the process goes to running state after dispatching after which the process gets released and at last it is terminated.

Possible State Transitions

There can be various events that lead to a state transition for a process. The possible state transitions are given below:

1. **Null -> New:** A new process is created for the execution of a process.
2. **New -> Ready:** The system will move the process from new to ready state and now it is ready for execution. Here a system may set a limit so that multiple processes can't occur otherwise there may be a performance issue.
3. **Ready -> Running:** The OS now selects a process for a run and the system chooses only one process in a ready state for execution.
4. **Running -> Exit:** The system terminates a process if the process indicates that is now completed or if it has been aborted.
5. **Running -> Ready:** The reason for which this transition occurs is that when the running process has reached its maximum running time for uninterrupted execution. An example of this can be a process

running in the background that performs some maintenance or other functions periodically.

6. **Running -> Blocked:** A process is put in the blocked state if it requests for something it is waiting. Like, a process may request some resources that might not be available at the time or it may be waiting for an I/O operation or waiting for some other process to finish before the process can continue.
7. **Blocked -> Ready:** A process moves from blocked state to the ready state when the event for which it has been waiting.
8. **Ready -> Exit:** This transition can exist only in some cases because, in some systems, a parent may terminate a child's process at any time.

Benefits:

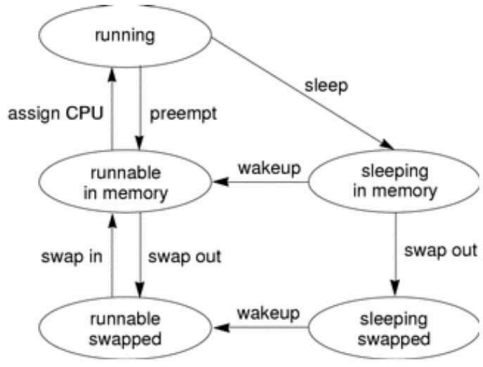
1. The New and Exit state are very useful constructs for managing the process.
2. It is an efficient way of the previous two-state process model.

Drawbacks:

1. If a process gets terminated or exit from the OS, its data is not preserved by the OS.
2. There can be a performance issue as in a situation where each process goes to a blocked state the CPU stays idle until at least one process leaves the waiting state.

22 What is process? Explain Process State Transition Diagram in detail.

ANS- (Que-11)



An active process is normally in one of the five states in the diagram. The arrows show how the process changes states.

- A process is running if the process is assigned to a CPU. A process is removed from the running state by the scheduler if a process with a higher priority becomes runnable. A process is also pre-empted if a process of

equal priority is runnable when the original process consumes its entire time slice.

- A process is runnable in memory if the process is in primary memory and ready to run, but is not assigned to a CPU.
- A process is sleeping in memory if the process is in primary memory but is waiting for a specific event before continuing execution. For example, a process sleeps while waiting for an I/O operation to complete, for a locked resource to be unlocked, or for a timer to expire. When the event occurs, a wakeup call is sent to the process. If the reason for its sleep is gone, the process becomes runnable.
- When a process' address space has been written to secondary memory, and that process is not waiting for a specific event, the process is runnable and swapped.
- If a process is waiting for a specific event and has had its whole address space written to secondary memory, the process is sleeping and swapped.

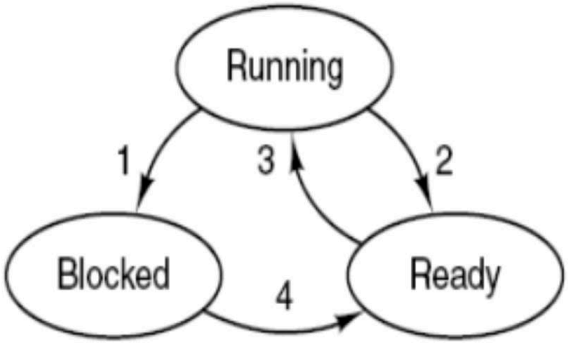
  If a machine does not have enough primary memory to hold all its active processes, that machine must page or swap some address space to secondary memory.

- When the system is short of primary memory, the system writes individual pages of some processes to secondary memory but leaves those processes runnable. When a running process, accesses those pages, the process sleeps while the pages are read back into primary memory.
- When the system encounters a more serious shortage of primary memory, the system writes all the pages of some processes to secondary memory. The system marks the pages that have been written to secondary memory as swapped. Such processes can only be scheduled when the system scheduler daemon selects these processes to be read back into memory.

Both paging and swapping cause delay when a process is ready to run again. For processes that have strict timing requirements, this delay can be unacceptable.

23 What is process? Explain Process State Transition Diagram in detail. Compare various disk arm scheduling algorithm.

ANS- (Que-11)



1. Process blocks for input
2. Scheduler picks another process
3. Scheduler picks this process
4. Input becomes available

Process state transition diagram

- Figure above shows the state transition diagram.
- Logically, the first two states are similar. In both cases the process is willing to run, but in the ready state there is no CPU temporarily available for it.
- In blocked state, the process cannot run even if the CPU is available to it as the process is waiting for some external event to take place.
- There four possible transitions between these three states.
- Transition 1 occurs when the operating system discovers that a process cannot continue right now due to unavailability of input. In other systems including UNIX, when a process reads from a pipe or special file (e.g. terminal) and there is no input available, the process is automatically blocked.
- Transition 2 and 3 are caused by the process scheduler (a part of the operating system), without the process even knowing about them.
- Transition 2 occurs when the scheduler decides that the running process has run long enough, and it is time to let another process have some CPU time.
- Transition 3 occurs when all the other processes have had their fair share and it is time for the first process to get the CPU to run again. The subject of scheduling, that is, deciding which process should run when and for how long, is an important one.
- Transition 4 occurs when the external event for which a process was waiting (such as the arrival of some input) happens. If no other process is running at that time, transition 3 will be triggered and the process will start running. Otherwise it may have to wait in ready state for a little time until the CPU is available and its turn comes.

UNIT – 3

1 What Critical section Problem and list the requirements to solve it. Write Peterson’s Solution for the same

The solution to the Critical Section Problem

A solution to the critical section problem must satisfy the following three conditions:

1. Mutual Exclusion

Out of a group of cooperating processes, only one process can be in its critical section at a given point of time.

2. Progress

If no process is in its critical section, and if one or more threads want to execute their critical section then any one of these threads must be allowed to get into its critical section.

3. Bounded Waiting

After a process makes a request for getting into its critical section, there is a limit for how many other processes can get into their critical section, before this process's request is granted. So after the limit is reached, the system must grant the process permission to get into its critical section.

Solutions for the Critical Section

The critical section plays an important role in Process Synchronization so that the problem must be solved.

Some widely used method to solve the critical section problem are as follows:

1.Peterson's Solution

This is widely used and software-based solution to critical section problems. Peterson's solution was developed by a computer scientist Peterson that's why it is named so.

With the help of this solution whenever a process is executing in any critical state, then the other process only executes the rest of the code, and vice-versa can happen.

This method also helps to make sure of the thing that only a single process can run in the critical section at a specific time.

This solution preserves all three conditions:

- Mutual Exclusion is comforted as at any time only one process can access the critical section.
- Progress is also comforted, as a process that is outside the critical section is unable to block other processes from entering into the critical section.
- Bounded Waiting is assured as every process gets a fair chance to enter the Critical section.

2 What is Semaphore? Give the implementation of Readers-Writers Problem using Semaphore.

A semaphore is an integer variable, shared among multiple processes. The main aim of using a semaphore is process synchronization and access control for a common resource in a concurrent environment.

The mutex semaphore ensures mutual exclusion and wrt handles the writing mechanism and is common to the reader and writer process code.

The variable rc denotes the number of readers accessing the object. As soon as rc becomes 1, wait operation is used on wrt. This means that a writer cannot access the object anymore.

3 What is Monitor? Write Solution to Dining-Philosopher problem Using monitor.

Monitors in Operating System

Monitors are used for process synchronization. With the help of programming languages, we can use a monitor to achieve mutual exclusion among the processes. **Example of monitors: *Java Synchronized methods such as Java offers notify() and wait() constructs.***

In other words, monitors are defined as the construct of programming language, which helps in controlling shared data access.

Monitor-based Solution to Dining Philosophers

Monitor is **used to control access to state variables and condition variables**. It only tells when to enter and exit the segment. This solution imposes the restriction that a philosopher may pick up her chopsticks only if both of them are available.

4 What is Semaphore? Give the implementation of Bounded Buffer Producer Consumer Problem using Semaphore. Write pseudo code for the same.

Definition – question2

5 Define and explain following terms: (i) Authentication (ii) Mutual Exclusion (iii) Deadlock (iv)Segmentation.

1. Authentication is the process of recognizing a user's identity. It is the mechanism of associating an incoming request with a set of identifying credentials. The credentials provided are compared to those on a file in a database of the authorized user's information on a local operating system or within an authentication server.
2. **Mutual exclusion** is a property of process synchronization which states that "no two processes can exist in the critical section at any given point of time". The term was first coined by Dijkstra. Any process synchronization technique being used must satisfy the property of mutual exclusion, without which it would not be possible to get rid of a race condition.
3. Deadlock is **a situation that occurs in OS when any process enters a waiting state because another waiting process is holding the demanded resource**. Deadlock is a common problem in multi-processing where several processes share a specific type of mutually exclusive resource known as a soft lock or software.
4.

In Operating Systems, Segmentation is a memory management technique in which the memory is divided into the variable size parts. Each part is known as a segment which can be allocated to a process.

The details about each segment are stored in a table called a segment table. Segment table is stored in one (or many) of the segments.

Segment table contains mainly two information about segment:

1. Base: It is the base address of the segment
2. Limit: It is the length of the segment.

5 Write short note: 1) Semaphores 2) Monitors

1. **Semaphore-** Semaphores are integer variables that are used to solve the critical section problem by using two atomic operations, wait and signal that are used for process synchronization.

Types of Semaphores

There are two main types of semaphores i.e. counting semaphores and binary semaphores. Details about these are given as follows −

- **Counting Semaphores**

  These are integer value semaphores and have an unrestricted value domain. These semaphores are used to coordinate the resource access, where the semaphore count is the number of available resources. If the resources are added, semaphore count automatically incremented and if the resources are removed, the count is decremented.

- **Binary Semaphores**

  The binary semaphores are like counting semaphores but their value is restricted to 0 and 1. The wait operation only works when the semaphore is 1 and the signal operation succeeds when semaphore is 0. It is sometimes easier to implement binary semaphores than counting semaphores.

Advantages of Semaphores

Some of the advantages of semaphores are as follows −

- Semaphores allow only one process into the critical section. They follow the mutual exclusion principle strictly and are much more efficient than some other methods of synchronization.
- There is no resource wastage because of busy waiting in semaphores as processor time is not wasted unnecessarily to check if a condition is fulfilled to allow a process to access the critical section.
- Semaphores are implemented in the machine independent code of the microkernel. So they are machine independent.

Disadvantages of Semaphores

Some of the disadvantages of semaphores are as follows −

- Semaphores are complicated so the wait and signal operations must be implemented in the correct order to prevent deadlocks.
- Semaphores are impractical for last scale use as their use leads to loss of modularity. This happens because the wait and signal operations prevent the creation of a structured layout for the system.
- Semaphores may lead to a priority inversion where low priority processes may access the critical section first and high priority processes later.

Monitor Monitors are used for process synchronization. With the help of programming languages, we can use a monitor to achieve mutual exclusion among the processes. **Example of monitors:** ***Java Synchronized methods such as Java offers notify() and wait() constructs.***

In other words, monitors are defined as the construct of programming language, which helps in controlling shared data access.
The Monitor is a module or package which encapsulates shared data structure, procedures, and the synchronization between the concurrent procedure invocations.

Characteristics of Monitors.

1. Inside the monitors, we can only execute one process at a time.
2. Monitors are the group of procedures, and condition variables that are merged together in a special type of module.

3. If the process is running outside the monitor, then it cannot access the monitor's internal variable. But a process can call the procedures of the monitor.
4. Monitors offer high-level of synchronization
5. Monitors were derived to simplify the complexity of synchronization problems.
6. There is only one process that can be active at a time inside the monitor.

Components of Monitor

There are four main components of the monitor:

1. Initialization
2. Private data
3. Monitor procedure
4. Monitor entry queue

Initialization: - Initialization comprises the code, and when the monitors are created, we use this code exactly once.
Private Data: - Private data is another component of the monitor. It comprises all the private data, and the private data contains private procedures that can only be used within the monitor. So, outside the monitor, private data is not visible.
Monitor Procedure: - Monitors Procedures are those procedures that can be called from outside the monitor.
Monitor Entry Queue: - Monitor entry queue is another essential component of the monitor that includes all the threads, which are called procedures.

Monitors	Semaphore
We can use condition variables only in the monitors.	In semaphore, we can use condition variables anywhere in the program, but we cannot use conditions variables in a semaphore.
In monitors, wait always block the caller.	In semaphore, wait does not always block the caller.
The monitors are comprised of the shared variables and the procedures which operate the shared variable.	The semaphore S value means the number of shared resources that are present in the system.
Condition variables are present in the monitor.	Condition variables are not present in the semaphore.

UNIT – 4

1 Explain the IPC Problem known as Dining Philosopher Problem.

The dining philosopher's problem is **the classical problem of synchronization which says that Five philosophers are sitting around a circular table and their job is to think and eat alternatively**. A bowl of noodles is placed at the center of the table along with five chopsticks for each of the philosophers.

2 Define: Race Condition, Mutual Exclusion, Throughput.

Race Condition

A race condition is a situation that may occur inside a critical section. This happens when the result of multiple thread execution in critical section differs according to the order in which the threads execute.

Race conditions in critical sections can be avoided if the critical section is treated as an atomic instruction. Also, proper thread synchronization using locks or atomic variables can prevent race conditions.

Mutual Exclusion

A mutual exclusion (mutex) is a program object that prevents simultaneous access to a shared resource. This concept is used in concurrent programming with a critical section, a piece of code in which processes or threads access a shared resource. Only one thread owns the mutex at a time, thus a mutex with a unique name is created when a program starts. When a thread holds a resource, it has to lock the mutex from other threads to prevent concurrent access of the resource. Upon releasing the resource, the thread unlocks the mutex

Throughput.

Throughput is a measure of how many units of information a system can process in a given amount of time. It is applied broadly to systems ranging from various aspects of computer and network systems to organizations. Related measures of system productivity include the speed with which some specific workload can be completed, and response time, the amount of time between a single interactive user request and receipt of the response.

3 Explain the Problem of Critical Section (CSP) through Producer Consumer Problem. Explain any one Solution in detail.

In the previous blogs, we have learned about process synchronization and we have also seen how to achieve the process synchronization by using semaphores. In this blog, we will learn

about the Producer-Consumer problem which is used for multi-process synchronization.

If you have no idea about the process synchronization, then learn from here. Also, learn about semaphore from here. Now, you are done with the prerequisites of the blog, so let's get started with the producer-consumer problem.

Solution-

1. **Semaphore S:** This semaphore variable is used to achieve mutual exclusion between processes. By using this variable, either Producer or Consumer will be allowed to use or access the shared buffer at a particular time. This variable is set to 1 initially.
2. **Semaphore E:** This semaphore variable is used to define the empty space in the buffer. Initially, it is set to the whole space of the buffer i.e. "n" because the buffer is initially empty.

4 What is race condition? Explain the producer consumer problem with a fatal race condition.

A race condition is **an undesirable situation that occurs when a device or system attempts to perform two or more operations at the same time**, but because of the nature of the device or system, the operations must be done in the proper sequence to be done correctly.

Producer-Consumer problem

The Producer-Consumer problem is a classical multi-process synchronization problem, that is we are trying to achieve synchronization between more than one process.

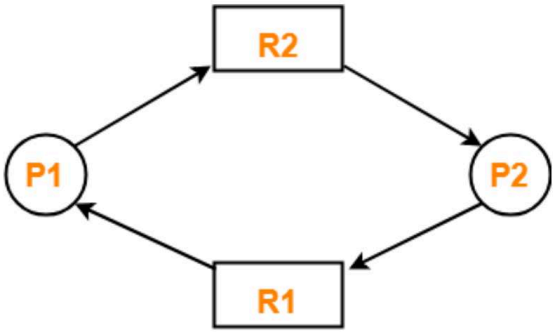
There is one Producer in the producer-consumer problem, Producer is producing some items, whereas there is one Consumer that is consuming the items produced by the Producer. The same memory buffer is shared by both producers and consumers which is of fixed-size.

The task of the Producer is to produce the item, put it into the memory buffer, and again start producing items. Whereas the task of the Consumer is to consume the item from the memory buffer.

1 What is deadlock? List the conditions that lead to deadlock.

- The execution of two or more processes is blocked because each process holds some resource and waits for another resource held by some other process.

Example;



Example of a deadlock

Here

- Process P1 holds resource R1 and waits for resource R2 which is held by process P2.
- Process P2 holds resource R2 and waits for resource R1 which is held by process P1.
- None of the two processes can complete and release their resource.
- Thus, both the processes keep waiting infinitely.

Conditions For Deadlock-

There are following 4 necessary conditions for the occurrence of deadlock-

1. Mutual Exclusion
2. Hold and Wait
3. No preemption
4. Circular wait

1. Mutual Exclusion-

By this condition,

- There must exist at least one resource in the system which can be used by only one process at a time.
- If there exists no such resource, then deadlock will never occur.
- Printer is an example of a resource that can be used by only one process at a time.

2. Hold and Wait-

By this condition,

- There must exist a process which holds some resource and waits for another resource held by some other process.

3. No Preemption-

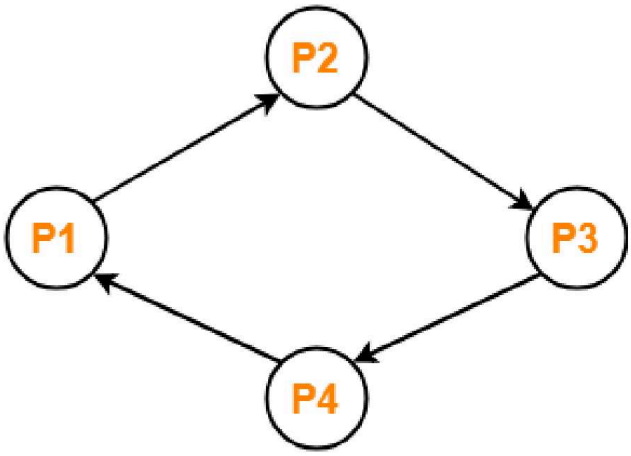
By this condition,

- Once the resource has been allocated to the process, it can not be preempted.
- It means resource can not be snatched forcefully from one process and given to the other process.
- The process must release the resource voluntarily by itself.

4. Circular Wait-

By this condition,

- All the processes must wait for the resource in a cyclic manner where the last process waits for the resource held by the first process.



Circular Wait

Here,

- Process P1 waits for a resource held
- Process P4 waits for a resource held by process P1.

2. Consider the snapshot of the system with Five Processes and Four types of resources A, B, C, D. Currently Available set of resources is (1,5,2,0). Answer the following Questions using banker's algorithm. 1) Find the content of Need Matrix. 2) Is the System in Safe State? 3) If request from Process P1 arrives for (0,4,2,0) can the request be granted immediately

	Allocation				Max				Available			
	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	1	5	2	0
P_1	1	0	0	0	1	7	5	0				
P_2	1	3	5	4	2	3	5	6				
P_3	0	6	3	2	0	6	5	2				
P_4	0	0	1	4	0	6	5	6				

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

	Need(Max-Allocation)			
	A	B	C	D
P_0	0	0	0	0
P_1	0	7	5	0
P_2	1	0	0	2
P_3	0	0	2	0
P_4	0	6	4	2

To check if system is in a safe state

- The Available matrix is $[1520]$[1520].
- A process after it has finished execution is supposed to free up all the resources it hold.
- We need to find a safety sequence such that it satisfies the criteria need $Need \leq Available$Need≤Available.
- Since $Need(P0) \leq Available$Need(P0)≤Available , we select $P0.[Available]=[Available]+[Allocation(P0)]$P0.[Available]=[Available] +[Allocation(P0)]

$Available=[1520]+[0012]=[1532]$Available=[1520]+[0012]=[1532]

- Need(P2) ≤ Available →→Available=[1 5 3 2]+[1 3 5 4]=[2 8 8 6]
- Need(P3) ≤ Available→→ Available=[2 8 8 6]+[0 6 3 2]=[2 14 11 8]
- Need(P4) ≤ Available→→Available=[2 14 11 8]+[0 0 1 4]=[2 14 12 12]
- Need(P1) ≤ Available→→Available=[2 14 12 12]+[1 0 0 0]=[3 14 12 12]
- Safe Sequence is <p0,p2,p3,p4,p1>

A request from process P1 arrives for (0,4,2,0)

- System receives a request for P1 for $Req(P1)[0420]$Req(P1)[0420]

- First we check if Req(P1) is less than $Need(P1)$Need(P1)$\rightarrow[0420]<[0750]istrue$[0420]<[0750]istrue
- Now we check if Req(P1) is less than $Available$Available$\rightarrow[0420]<[1520]istrue$[0420]<[1520]istrue.
- So we update the values as:
    - $Available=Available-Request=[1520]-[0420]=[1100]$Available=Available−Request=[1520]−[0420]=[1100]
    - $Allocation=allocation(P1)+Request=[1000]+[0420]=[1420]$Allocation=allocation(P1)+Request=[1000]+[0420]=[1420]
-
    - $Need=Need(P1)-Request=[0750]-[0420]=[0330]$Need=Need(P1)−Request=[0750]−[0420]=[0330]

	Allocation				Max				Need				Available			
	A	B	C	D	A	B	C	D	A	B	C	D	A	B	C	D
P_0	0	0	1	2	0	0	1	2	0	0	0	0	1	1	0	0
P_1	**1**	**4**	**2**	**0**	**1**	**7**	**5**	**0**	**0**	**3**	**3**	**0**				
P_2	1	3	5	4	2	3	5	6	1	0	0	2				
P_3	0	6	3	2	0	6	5	2	0	0	2	0				
P_4	0	0	1	4	0	6	5	6	0	6	4	2				

- This is the modified table
- On verifying, we see that the safe sequence still remains the same .The system continues to remain in a safe state.

3 List Deadlock Recovery Techniques and explain one of them.

When a Deadlock Detection Algorithm determines that a deadlock has occurred in the system, the system must recover from that deadlock. There are two approaches of breaking a Deadlock:

1. Process Termination:
To eliminate the deadlock, we can simply kill one or more processes. For this, we use two methods:

- **(a). Abort all the Deadlocked Processes:**
  Aborting all the processes will certainly break the deadlock, but with a great expense. The deadlocked processes may have computed for a long time and the result of those partial computations must be discarded and there is a probability to recalculate them later.

- **(b). Abort one process at a time until deadlock is eliminated:**
  Abort one deadlocked process at a time, until deadlock cycle is eliminated from the system. Due to this method, there may be considerable overhead, because after aborting each process, we

have to run deadlock detection algorithm to check whether any processes are still deadlocked.

4 Define: Starvation

Starvation happens if a method is indefinitely delayed. This can emerge once a method needs a further resource for execution that isn't assigned.

These resources are things like:

- CPU time
- memory
- disk space
- network bandwidth
- I/O access to network or disk

Starvation is the problem that occurs when low priority processes get *jammed* for an unspecified time as the high priority processes keep executing.

A steady stream of higher-priority methods will stop a low-priority process from ever obtaining the processor.

5 What is Deadlock? List the conditions that lead to deadlock. How Deadlock can be prevented?

Ans – (Que1)

- Deadlock prevention works by **preventing one of the four Coffman conditions from occurring**. Removing the mutual exclusion condition means that no process will have exclusive access to a resource. This proves impossible for resources that cannot be spooled. But even with spooled resources, the deadlock could still occur. Algorithms that avoid mutual exclusion are called non-blocking synchronization algorithms.
- The hold and wait or resource holding conditions may be removed by requiring processes to request all the resources they will need before starting up (or before embarking upon a particular set of operations). This advance knowledge is frequently difficult to satisfy and, in any case, is an inefficient use of resources. Another way is to require processes to request resources only when it has none; First they must release all their currently held resources before requesting all the resources they will need from scratch. This too is often impractical. It is so because resources may be allocated and remain unused for long periods. Also, a process requiring a popular resource may have to wait indefinitely, as such a resource may always be allocated to some process, resulting in resource starvation.

6 Explain the use of Banker's Algorithm for multiple resources for Deadlock Avoidance with illustration

It is a banker algorithm used to **avoid deadlock** and **allocate resources** safely to each process in the computer system. The '**S-State'** examines all possible tests or activities before deciding whether the allocation should be allowed to each process. It also helps the operating system to successfully share the resources between all the processes. The banker's algorithm is named because it checks whether a person should be sanctioned a loan amount or not to help the bank system safely simulate allocation resources. In this section, we will learn the **Banker's Algorithm** in detail. Also, we will solve problems based on the **Banker's Algorithm**. To understand the Banker's Algorithm first we will see a real word example of it.

Suppose the number of account holders in a particular bank is 'n', and the total money in a bank is 'T'. If an account holder applies for a loan; first, the bank subtracts the loan amount from full cash and then estimates the cash difference is greater than T to approve the loan amount. These steps are taken because if another person applies for a loan or withdraws some amount from the bank, it helps the bank manage and operate all things without any restriction in the functionality of the banking system.

Similarly, it works in an **operating system**. When a new process is created in a computer system, the process must provide all types of information to the operating system like upcoming processes, requests for their resources, counting them, and delays. Based on these criteria, the operating system decides which process sequence should be executed or waited so that no deadlock occurs in a system. Therefore, it is also known as **deadlock avoidance algorithm** or **deadlock detection** in the operating system.

7 What is RAG? Explain briefly.

RAG is the abbreviation for Resource Allocation Graph in operating systems. RAG is a directed graph which can be used to represent the state of a system in the form of picture. Deadlocks can be described more precisely in terms of a directed graph(RAG).
The graph(V, E) consists a set of vertices which can be partitioned into two different types of nodes:-

1. **Process vertices –**
   They represents the processes and are drawn as a circle, e.g., `{P1, P2, P3.....Pn}`
2. **Resource vertices –**
   These vertices represents the resources and are drawn as a square with dots in it, which represent the instance of resources, e.g., `{R1, R2, R3.....Rn}`

Resource Allocation Graphs are drawn in order to see the allocation relations between the processes and Resources. Here, Processes are represented inside a circle whereas Resources are represented inside a square with dots inside it indicating the number of instances of resources.

RAG have two types of Edges, one which represents the assignment and other represents the wait of a process for a resource.

- A resource is assigned to a process if the tail of the arrow is attached to an instance to the resource and the head is attached to a process.
- A process is waiting for a resource if the tail of an arrow is attached to the process while the head is pointing towards the resource.

If RAG contains a cycle, then the system is in deadlock otherwise not.

Characteristics of RAG:

- **Pictorial representation –**
  RAG is the pictorial representation of the states of a system
- **Deadlock detection –**
  Using RAG, we can easily detect that whether the system is in deadlock or not.
- **Resources Information –**
  RAG contains all the information of resources and its instances that whether they are free or currently being used by any other processes.
- **Processes Information –**
  RAG tells us about the processes that which process is holding which resource and which resource, it is requesting.

Advantages of RAG:

- It is very useful in deadlock detection.
- It is widely used in the Banker's Algorithm.
- It is a pictorial representation of a system..
- Sometimes we can tell whether the system is in deadlock or not by just having a glance at the graph.
- It takes less time to understand the allocation of resources via RAG.

Disadvantages of RAG:

- RAG is useful when we have less number of processes and resources.
- With large number of resources or processes, it is better to store data in a table rather than RAG.
- If there are large number of resources or processes, then the graph will be difficult to understand and will become complex.

8 Which are the necessary conditions for Deadlock? Explain Deadlock recovery in brief.

Necessary Conditions of Deadlock

There are four different conditions that result in Deadlock. These four conditions are also known as Coffman conditions and these conditions are not mutually exclusive. Let's look at them one by one.

- **Mutual Exclusion:** A resource can be held by only one process at a time. In other words, if a process P1 is using some resource R at a particular instant of time, then some other process P2 can't hold or use the same resource R at that particular instant of time. The process P2 can make a request for that resource R but it can't use that resource simultaneously with process P1.
- **Hold and Wait:** A process can hold a number of resources at a time and at the same time, it can request for other resources that are being held by some other process. For example, a process P1 can hold two resources R1 and R2 and at the same time, it can request some resource R3 that is currently held by process P2.

- **No preemption:** A resource can't be preempted from the process by another process, forcefully. For example, if a process P1 is using some resource R, then some other process P2 can't forcefully take that resource. If it is so, then what's the need for various scheduling algorithm. The process P2 can request for the resource R and can wait for that resource to be freed by the process P1.
- **Circular Wait:** Circular wait is a condition when the first process is waiting for the resource held by the second process, the second process is waiting for the resource held by the third process, and so on. At last, the last process is waiting for the resource held by the first process. So, every

process is waiting for each other to release the resource and no one is releasing their own resource. Everyone is waiting here for getting the resource. This is called a circular wait.

OS Deadlock Recovery

Deadlock recovery performs when a deadlock is detected.

When deadlock detected, then our system stops working, and after the recovery of the deadlock, our system start working again.

Therefore, after the detection of deadlock, a method/way must require to recover that deadlock to run the system again. The method/way is called as deadlock recovery.

Here are various ways of deadlock recovery that we will discuss briefly in this tutorial.

- Deadlock recovery through preemption
- Deadlock recovery through rollback
- Deadlock recovery through killing processes

Let's discuss about all the above three ways of deadlock recovery one by one.

Deadlock Recovery through Preemption

The ability to take a resource away from a process, have another process use it, and then give it back without the process noticing. It is highly dependent on the nature of the resource.

Deadlock recovery through preemption is too difficult or sometime impossible.

Deadlock Recovery through RollBack

In this case of deadlock recovery through rollback, whenever a deadlock is detected, it is easy to see which resources are needed.

To do the recovery of deadlock, a process that owns a needed resource is rolled back to a point in time before it acquired some other resource just by starting one of its earlier checkpoints.

Deadlock Recovery through Killing Processes

This method of deadlock recovery through killing processes is the simplest way of deadlock recovery.

Sometime it is best to kill a process that can be return from the beginning with no ill effects.

9 Define mutual exclusion. How mutual exclusion can be achieved? Explain

In computer science, **mutual exclusion** is a property of concurrency control, which is instituted for the purpose of preventing race conditions. It is the requirement that one thread of execution never enters a critical section while a concurrent thread of execution is already accessing critical section, which refers to an interval of time during which a thread of execution accesses a shared resource, such as [Shared data objects, shared resources, shared memory].

The shared resource is a data object, which two or more concurrent threads are trying to modify (where two concurrent read operations are permitted but, no two concurrent write operations or one read and one write are permitted, since it leads to data inconsistency). Mutual exclusion algorithm ensures that if a process is already performing write operation on a data object [critical section] no other process/thread is allowed to access/modify the same object until the first process has finished writing upon the data object [critical section] and released the object for other processes to read and write upon.

The requirement of mutual exclusion was first identified and solved by Edsger W. Dijkstra in his seminal 1965 paper "Solution of a problem in concurrent programming control",[1][2] which is credited as the first topic in the study of concurrent algorithms.[3]

1 Given memory partition of 100K, 500K, 200K, 300K, and 600K in order, How would each of the First fit, Best fit and Worst fit algorithms place the processes of 212K, 417K, 112K and 426K in order? Which algorithm makes the most efficient use of memory? Show the diagram of memory status in each case

First fit

212 K is put in 500 K partition.

417 K is put in 600 K partition.

112 K is put in 288 K partition. (New partition 288 K = 500 K - 212 K)

426 K must wait.

Best-fit

212 K is put in 300 K partition.

417 K is put in 500 K partition.

112 K is put in 200 K partition.

426 K is put in 600 K partition.

Worst-fit

212 K is put in 600 K partition.

417 K is put in 500 K partition.

112 K is put in 388 K partition. (600 K - 212 K)

426 K must wait.

In this example Best-fit is the best solution.

2 Explain the following in brief: Multiprogramming with Fixed Partitions and Multiprogramming with Variable Partition

1. **Fixed Partitioning :**
   Multi-programming with fixed partitioning is a contiguous memory management technique in which the main memory is divided into fixed sized partitions which can be of equal or unequal size. Whenever we have to allocate a process memory then a free partition that is big enough to hold the process is found. Then the memory is allocated to the process.If there is no free space available then the process waits in the queue to be allocated memory. It is one of the most oldest memory management technique which is easy to implement.
2. **Variable Partitioning :**
   Multi-programming with variable partitioning is a contiguous memory management technique in which the main memory is not divided into partitions and the process is allocated a chunk of free memory that is big enough for it to fit. The space which is left is considered as the free space which can be further used by other processes. It also provides the concept of compaction. In compaction the spaces that are free and the spaces which not allocated to the process are combined and single large memory space is made.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	1	1	0	0	0	3	3	3	3	3	2	2	2	2	2	1
	0	0	0	0	3	3	3	2	2	2	2	2	1	1	1	1	1	0	0
7	7	7	2	2	2	2	4	4	4	0	0	0	0	0	0	0	7	7	7
F	F	F	F		F	F	F	F	F	F			F	F			F	F	F

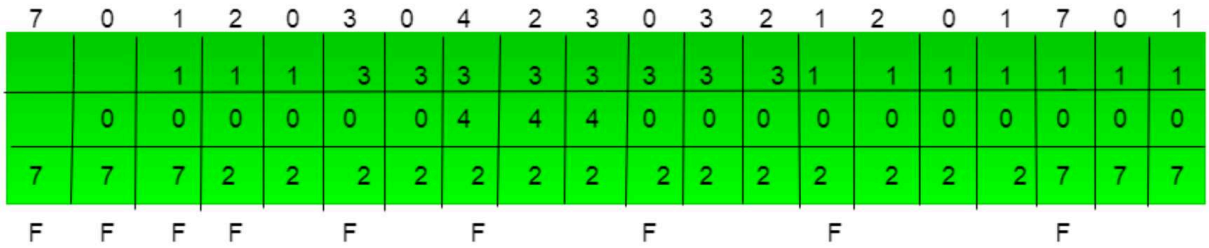
In FIFO algorithm 15 page fault will occur

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	2	2	3	3	4	2	3	3	3	2	1	2	2	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
F	F	F	F		F		F	F	F			F	F	F					

In LIFO algorithm 11 page fault will occur.

7	0	1	2	0	3	0	4	2	3	0	3	2	1	2	0	1	7	0	1
		1	1	1	3	3	3	2	2	2	2	2	2	2	2	2	7	7	7
	0	0	0	0	0	0	0	0	3	3	3	3	3	3	0	0	0	0	0
7	7	7	2	2	2	2	4	4	4	0	0	0	1	1	1	1	1	1	1
F	F	F	F		F		F	F	F	F			F	F			F		

In LRU algorithm 12 page fault will occur.



1. In optimal algorithm 9 page fault will occur.

So, option (D) will be correct.

4 Explain Swapping in Detail.

Swapping is a memory management technique used in multi-programming to increase the number of processes sharing the CPU. It is a technique of removing a process from the main memory and storing it into secondary memory, and then bringing it back into the main memory for continued execution. This action of moving a process out from main memory to secondary memory is called **Swap Out** and the action of moving a process out from secondary memory to main memory is called **Swap In**.

Swap-Space :
The area on the disk where the swapped-out processes are stored is called swap space.

Swap-Space Management :
Swap-Swap management is another low-level task of the operating system. Disk space is used as an extension of main memory by the virtual memory. As we know the fact that disk access is much slower than memory access, In the swap-space management we are using disk space, so it will significantly decreases system performance. Basically, in all our systems we require the best throughput, so the goal of this swap-space implementation is to provide the virtual memory the best throughput. In these article, we are going to discuss how swap space is used, where swap space is located on disk, and how swap space is managed.

Swap-Space Use :
Swap-space is used by the different operating-system in various ways. The systems which are implementing swapping may use swap space to hold the entire process which may include image, code and data segments. Paging systems may simply store pages that have been pushed out of the main memory. The need of swap space on a system can vary from a megabytes to gigabytes but it also depends on the amount of physical memory, the virtual memory it is backing and the way in which it is using the virtual memory.
It is safer to overestimate than to underestimate the amount of swap space required, because if a system runs out of swap space it may be forced to

abort the processes or may crash entirely. Overestimation wastes disk space that could otherwise be used for files, but it does not harm other.

UNIT – 7

1 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 order, is 86, 1470, 913, 1774, 948, 1509, 1022, 1750, 130 Starting from the current head position, what is the total distance (in cylinders) that the disk arm moves to satisfy all the pending requests, for each of the following disk scheduling i) FCFS, ii) SCAN

I) The FCFS schedule is 143, 86, 1470, 913, 1774, 948, 1509, 1022,1750, 130. The total seek distance is 7081.

II) The SCAN schedule is 143, 913, 948, 1022, 1470, 1509, 1750, 1774,4999, 130, 86. The total seek distance is 9769.

2 What are the uses of device driver and controller in OS?

Driver

A device driver allows a computer to interface and interact with a specific hardware device, such as a printer, sound card, graphics card, etc. The device controller receives the data from a connected device, stores it temporarily, and then communicates the data to its device driver

Device Controllers

Device drivers are software modules that can be plugged into an OS to handle a particular device. Operating System takes help from device drivers to handle all I/O devices. The Device Controller **works** like an interface between a device and a device driver.

3 EXPLAIN: Direct Memory Access

Direct memory access (DMA) is a method that allows an input/output (I/O) device to send or receive data directly to or from the main memory, bypassing the CPU to speed up memory operations.

The process is managed by a chip known as a DMA controller (DMAC)

Techopedia Explains Direct Memory Access (DMA)

A defined portion of memory is used to send data directly from a peripheral to the motherboard without involving the microprocessor, so that the process does not interfere with overall computer operation.

In older computers, four DMA channels were numbered 0, 1, 2 and 3. When the 16-bit industry standard architecture (ISA) expansion bus was introduced, channels 5, 6 and 7 were added.

ISA was a computer bus standard for IBM-compatible computers, allowing a device to initiate transactions (bus mastering) at a quicker speed. The ISA DMA controller has 8 DMA channels, each one of which associated with a 16-bit address and count registers.

ISA has since been replaced by accelerated graphics port (AGP) and peripheral component interconnect (PCI) expansion cards, which are much faster. Each DMA transfers approximately 2 MB of data per second.

A computer's system resource tools are used for communication between hardware and software. The four types of system resources are:

- I/O addresses.
- Memory addresses.
- Interrupt request numbers (IRQ).
- Direct memory access (DMA) channels.

DMA channels are used to communicate data between the peripheral device and the system memory. All four system resources rely on certain lines on a bus. Some lines on the bus are used for IRQs, some for addresses (the I/O addresses and the memory address) and some for DMA channels.

A DMA channel enables a device to transfer data without exposing the CPU to a work overload. Without the DMA channels, the CPU copies every piece of data using a peripheral bus from the I/O device. Using a peripheral bus occupies the CPU during the read/write process and does not allow other work to be performed until the operation is completed.

With DMA, the CPU can process other tasks while data transfer is being performed. The transfer of data is first initiated by the CPU. The data block can be transferred to and from memory by the DMAC in three ways.

In burst mode, the system bus is released only after the data transfer is completed. In cycle stealing mode, during the transfer of data between the DMA channel and I/O device, the system bus is relinquished for a few clock cycles so that the CPU can perform other tasks. When the data transfer is complete, the CPU receives an interrupt request from the DMA controller. In transparent mode, the DMAC can take charge of the system bus only when it is not required by the processor.

However, using a DMA controller might cause cache coherency problems. The data stored in RAM accessed by the DMA controller may not be updated with the correct cache data if the CPU is using external memory.

Solutions include flushing cache lines before starting outgoing DMA transfers, or performing a cache invalidation on incoming DMA transfers when external writes are signaled to the cache controller.

4 Explain SSTF and LOOK disk scheduling Algorithms

	LOOK Scheduling	
1.	The performance of LOOK is better than SSTF.	SSTF lags in performance.
2.	LOOK results in increased total seek time.	It reduces total seek time as compared to LOOK.
3.	It provides low variance in average waiting time and response time.	This algorithm provides high variance average response time and waiting time.
4.	As shown in above example, direction of head gets reversed when it serves the last request in one direction.	But here, direction of head plays an important role, in order to break tie between requests.
5.	In this algorithm, there is an overhead for finding end request.	Here, there is an overhead for finding out closest request.
6.	LOOK does not cause starvation to any request.	Here, the request which are far from head will suffer starvation.
7.	LOOK algorithm can handle requests more	Here handling of request is not so good as compared to LOOK algorithm.

effectively than
SSTF.

5 Explain Device Independent I/O software

I/O software is often organized in the following layers −

- **User Level Libraries** − This provides simple interface to the user program to perform input and output. For example, **stdio** is a library provided by C and C++ programming languages.
- **Kernel Level Modules** − This provides device driver to interact with the device controller and device independent I/O modules used by the device drivers.
- **Hardware** − This layer includes actual hardware and hardware controller which interact with the device drivers and makes hardware alive.

A key concept in the design of I/O software is that it should be device independent where it should be possible to write programs that can access any I/O device without having to specify the device in advance. For example, a program that reads a file as input should be able to read a file on a floppy disk, on a hard disk, or on a CD-ROM, without having to modify the program for each different device.

6 Explain the following in brief: Elevator Algorithm.

SCAN (Elevator) algorithm

In SCAN disk scheduling algorithm, head starts from one end of the disk and moves towards the other end, servicing requests in between one by one and reach the other end. Then the direction of the head is reversed and the process continues as head continuously scan back and forth to access the disk. So, this algorithm works as an elevator and hence also known as the **elevator algorithm**. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Advantages of SCAN (Elevator) algorithm

1. This algorithm is simple and easy to understand.
2. SCAN algorithm have no starvation.
3. This algorithm is better than FCFS Scheduling algorithm .

Disadvantages of SCAN (Elevator) algorithm

1. More complex algorithm to implement.
2. This algorithm is not fair because it cause long waiting time for the cylinders just visited by the head.
3. It causes the head to move till the end of the disk in this way the requests arriving ahead of the arm position would get immediate service but some other requests that arrive behind the arm position will have to wait for the request to complete.

Algorithm-

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. ‘head’ is the position of disk head.

2. Let direction represents whether the head is moving towards left or right.
3. In the direction in which head is moving service all tracks one by one.
4. Calculate the absolute distance of the track from the head.
5. Increment the total seek count with this distance.
6. Currently serviced track position now becomes the new head position.
7. Go to step 3 until we reach at one of the ends of the disk.
8. If we reach at the end of the disk reverse the direction and go to step 2 until all tracks in request array have not been serviced.

7 Explain any three Disk Arm Scheduling algorithms with suitable illustrations.

Disk Scheduling Algorithms

1. **FCFS:** FCFS is the simplest of all the Disk Scheduling Algorithms. In FCFS, the requests are addressed in the order they arrive in the disk queue.Let us understand this with the help of an example.
2. **SCAN:** In SCAN algorithm the disk arm moves into a particular direction and services the requests coming in its path and after reaching the end of disk, it reverses its direction and again services the request arriving in its path. So, this algorithm works as an elevator and hence also known as **elevator algorithm.** As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.
3. **CSCAN**: In SCAN algorithm, the disk arm again scans the path that has been scanned, after reversing its direction. So, it may be possible that too many requests are waiting at the other end or there may be zero or few requests pending at the scanned area.

These situations are avoided in *CSCAN* algorithm in which the disk arm instead of reversing its direction goes to the other end of the disk and starts servicing the requests from there. So, the disk arm moves in a circular fashion and this algorithm is also similar to SCAN algorithm and hence it is known as C-SCAN (Circular SCAN).

8 Explain the goals of I/O software.

Goals of I/O software

I/O Software is used for interaction with I/O devices like mouse, keyboard, USB devices, printers, etc. I/O software are organized in following ways:

User Level Libraries– Provides an simple interface to program for input output functions.
Kernel Level Modules– Provides device driver to interact with the device independent I/O modules and device controller.

Hardware-A layer including hardware controller and actual hardware which interact with device drivers.

Goals Of I/O Software

Here we talk about the goals of I/O software

Uniform naming: For example naming of files systems in Operating Systems is done in a way that user does not have to be aware of underlying hardware name.
Synchronous versus Asynchronous: When the CPU is working on some process it goes in the block state when the interrupt occurs. Therefore most of the devices are asynchronous. And if the I/O operation are in blocking state it is much easier to write the I/O operation. It is always the operating system responsibility to create such a interrupt driven user program.
Device Independence: The most important part of I/O software is device independence. It is always most preferable to write program which can open all other I/O devices. For example, it is not necessary to write the input taking program again and again for taking input from various file and devices. As this creates much work to do and also much space to store the different programs.
Buffering: Data that we enter into a system cannot be stored directly in memory. For example the data is converted into smaller groups and then transferred to outer buffer for examination.
Buffer have major impact on I/O software as it is the one which ultimately helps storing the data and copying data. Many device have constraints and just to avoid it some data is always put into the buffer in advance so the buffer rate of getting filled with data and getting empty remains balanced
Error handling: Errors and mostly generated by controller and also they are mostly handled by controller itself. When lower level solves the problem it does not reach the upper level.
Shareable and Non-Shareable Devices : Devices like Hard Disk can be shared among multiple process while devices like Printers cannot be shared. The goal of I/O software is to handle both types of devices.

9 Briefly describe SCAN

SCAN (Elevator) algorithm

In SCAN disk scheduling algorithm, head starts from one end of the disk and moves towards the other end, servicing requests in between one by one and reach the other end. Then the direction of the head is reversed and the process continues as head continuously scan back and forth to access the disk. So, this algorithm works as an elevator and hence also known as the **elevator algorithm**. As a result, the requests at the midrange are serviced more and those arriving behind the disk arm will have to wait.

Advantages of SCAN (Elevator) algorithm

1. This algorithm is simple and easy to understand.
2. SCAN algorithm have no starvation.
3. This algorithm is better than FCFS Scheduling algorithm .

Disadvantages of SCAN (Elevator) algorithm

1. More complex algorithm to implement.
2. This algorithm is not fair because it cause long waiting time for the cylinders just visited by the head.
3. It causes the head to move till the end of the disk in this way the requests arriving ahead of the arm position would get immediate service but some other requests that arrive behind the arm position will have to wait for the request to complete.

Algorithm-

1. Let Request array represents an array storing indexes of tracks that have been requested in ascending order of their time of arrival. 'head' is the position of disk head.
2. Let direction represents whether the head is moving towards left or right.
3. In the direction in which head is moving service all tracks one by one.
4. Calculate the absolute distance of the track from the head.
5. Increment the total seek count with this distance.
6. Currently serviced track position now becomes the new head position.
7. Go to step 3 until we reach at one of the ends of the disk.
8. If we reach at the end of the disk reverse the direction and go to step 2 until all tracks in request array have not been serviced.

10 Write short note: RAID levels.

- **RAID** (Redundant Array of Independent Disks) is a data storage virtualization technology that combines multiple physical disk drive components into a single logical unit for the purposes of data redundancy, performance improvement, or both.
- Data is distributed across the drives in one of several ways, referred to as RAID levels, depending on the required level of redundancy and performance.
- The different schemas, or data distribution layouts, are named by the word RAID followed by a number, for example RAID 0 or RAID 1.
- Each schema, or a RAID level, provides a different balance among the key goals: reliability, availability, performance, and capacity.
- RAID levels greater than RAID 0 provide protection against unrecoverable sector read errors, as well as against failures of whole physical drives.
- **RAID level descriptions:**

RAID	Description
0	consists of striping, without mirroring or parity
1	consists of data mirroring, without parity or striping
2	consists of bit-level striping with dedicated Hamming-code parity
3	consists of byte-level striping with dedicated parity
4	consists of block-level striping with dedicated parity
5	consists of block-level striping with distributed parity
6	consists of block-level striping with double distributed parity
0+1	creates a second striped set to mirror a primary striped set
1+0	creates a striped set from a series of mirrored drives

11 Draw the block diagram for DMA. Write steps for DMA data transfer.

Direct Memory Access (DMA) :

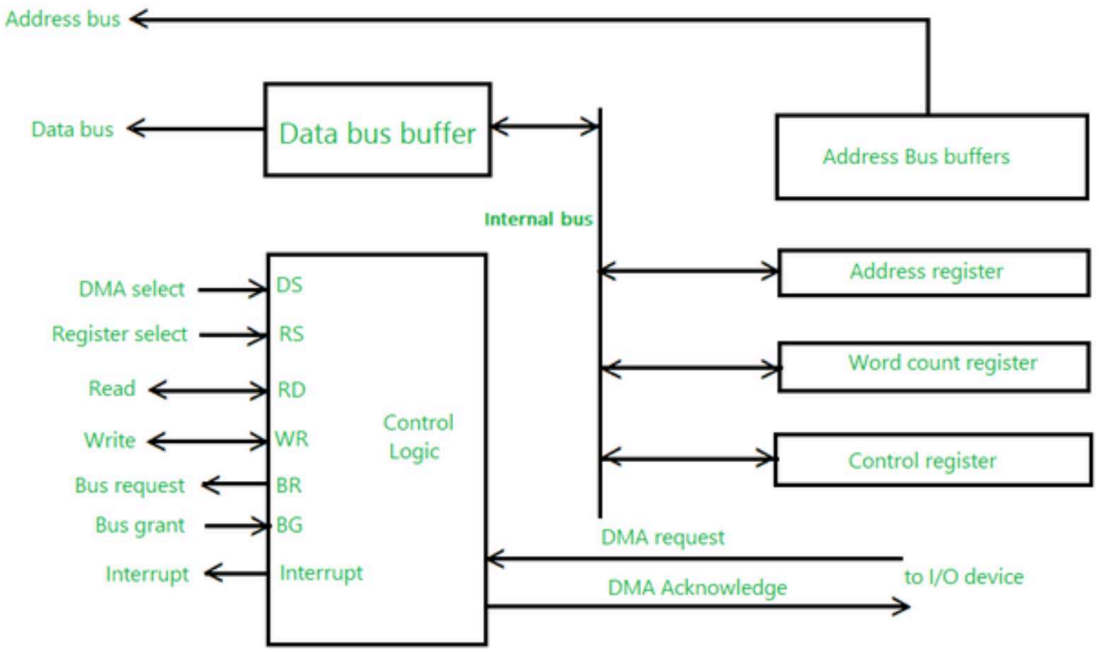
DMA Controller is a hardware device that allows I/O devices to directly access memory with less participation of the processor. DMA controller needs the same old circuits of an interface to communicate with the CPU and Input/Output devices.

Fig-1 below shows the block diagram of the DMA controller. The unit communicates with the CPU through data bus and control lines. Through the use of the address bus and allowing the DMA and RS register to select inputs, the register within the DMA is chosen by the CPU. RD and WR are two-way inputs. When BG (bus grant) input is 0, the CPU can communicate with DMA registers. When BG (bus grant) input is 1, the CPU has relinquished the buses and DMA can communicate directly with the memory.

DMA controller registers :
The DMA controller has three registers as follows.

- **Address register –** It contains the address to specify the desired location in memory.
- **Word count register –** It contains the number of words to be transferred.
- **Control register –** It specifies the transfer mode.

Note –
All registers in the DMA appear to the CPU as I/O interface registers. Therefore, the CPU can both read and write into the DMA registers under program control via the data bus.



Explanation :
The CPU initializes the DMA by sending the given information through the data bus.

- The starting address of the memory block where the data is available (to read) or where data are to be stored (to write).
- It also sends word count which is the number of words in the memory block to be read or write.
- Control to define the mode of transfer such as read or write.

- A control to begin the DMA transfer.

UNIT – 8

1 Implement Protection Mechanism illustrating use of Protection Domain and Access Control List.

Domain of Protection :

- The protection policies limit the access of each process with respect to their resource handling. A process is bound to use only those resources which it requires to complete its task, in the time limit that it requires and also the mode in which it is required. That is the protected domain of a process.
- A computer system has processes and objects, which are treated as abstract data types, and these objects have operations specific to them. A domain element is described as <object, {set of operations on object}>.
- Each domain consists of a set of objects and the operations that can be performed on them. A domain can consist of either only a process or a procedure or a user. Then, if a domain corresponds to a procedure, then changing domain would mean changing procedure ID. Objects may share a common operation or two. Then the domains overlap.

An access control list (ACL) contains rules that grant or deny access to certain digital environments. There are two types of ACLs:

- **Filesystem ACLs—**filter access to files and/or directories. Filesystem ACLs tell operating systems which users can access the system, and what privileges the users are allowed.
- **Networking ACLs—**filter access to the network. Networking ACLs tell routers and switches which type of traffic can access the network, and which activity is allowed.

Originally, ACLs were the only way to achieve firewall protection. Today, there are many types of firewalls and alternatives to ACLs. However, organizations continue to use ACLs in conjunction with technologies like virtual private networks (VPNs) that specify which traffic should be encrypted and transferred through a VPN tunnel.

2 Create the ways for the user authentication? Explain each in brief

Authentication is the process of identifying users that request access to a system, network, or device. Access control often determines user identity according to credentials like

username and password. Other authentication technologies like biometrics and authentication apps are also used to authenticate user identity.

User authentication is a method that keeps unauthorized users from accessing sensitive information. For example, User A only has access to relevant information and cannot see the sensitive information of User B.

Cybercriminals can gain access to a system and steal information when user authentication is not secure. The data breaches companies like Adobe, Equifax, and Yahoo faced are examples of what happens when organizations fail to secure their user authentication.

Hackers gained access to Yahoo user accounts to steal contacts, calendars and private emails between 2012 and 2016. The Equifax data breach in 2017 exposed credit card data of more than 147 million consumers. Without a secure authentication process, any organization could be at risk.

3 Explain domain protection mechanism in brief

Ans – (Question-1)

4 Explain the Trojan Horse and Trap doors program threats

1. Trojan Horse :
   A standalone malicious program which may give full control of infected PC to another PC is called Trojan horse. It may make copies of them, harm the host computer systems, or steal information.
   The Trojan horse will actually do damage once installed or run on your computer but at first glance will appear to be useful software. Trojans are designed as they can cause serious damage by deleting files and destroying information on your system.

   Trojans allow confidential or personal information to be compromised by system creating a backdoor on your computer that gives unauthorized users access to your system. Unlike Trojans do not self replicate reproduce by infecting other files nor do they self-replicate. Means Trojan horse viruses differ from other computer viruses and do not spread themselves.

   Most popular Trojan horses are Beast, Zeus, The Blackhole Exploit Kit, Flashback Trojan, Netbus, Subseven, Y3K Remote Administration Tool, Back Orifice.

2. Trap Door :
   A trap door is kind of a secret entry point into a program that allows anyone gain access to any system without going through the usual security access procedures. Other definition of trap door is it is a

method of bypassing normal authentication methods. Therefore it is also known as back door.
Programmers use Trap door legally to debug and test programs. Trap doors turns to threats when any dishonest programmers to gain illegal access. Program development and software update activities should be first focus of security measures. Operating system that controls the trap doors is difficult to implement.

5 Discuss some security goals.

The objective of Cybersecurity is to protect information from being stolen, compromised or attacked. Cybersecurity can be measured by at least one of three goals-

1. Protect the confidentiality of data.
2. Preserve the integrity of data.
3. Promote the availability of data for authorized users.

These goals form the confidentiality, integrity, availability (CIA) triad, the basis of all security programs. The CIA triad is a security model that is designed to guide policies for information security within the premises of an organization or company. This model is also referred to as the **AIC (Availability, Integrity, and Confidentiality)** triad to avoid the confusion with the Central Intelligence Agency. The elements of the triad are considered the three most crucial components of security.

The CIA criteria are one that most of the organizations and companies use when they have installed a new application, creates a database or when guaranteeing access to some data. For data to be completely secure, all of these security goals must come into effect. These are security policies that all work together, and therefore it can be wrong to overlook one policy.

1. Confidentiality

Confidentiality is roughly equivalent to privacy and avoids the unauthorized disclosure of information. It involves the protection of data, providing access for those who are allowed to see it while disallowing others from learning anything about its content. It prevents essential information from reaching the wrong people while making sure that the right people can get it. Data encryption is a good example to ensure confidentiality.

2. Integrity

Integrity refers to the methods for ensuring that data is real, accurate and safeguarded from unauthorized user modification. It is the property that information has not be altered in an unauthorized way, and that source of the information is genuine.

3. Availability

Availability is the property in which information is accessible and modifiable in a timely fashion by those authorized to do so. It is the guarantee of reliable and constant access to our sensitive data by authorized people.

6 Explain the goals of Operating System Security.

Authentication

Authentication refers to identifying each user of the system and associating the executing programs with those users. It is the responsibility of the Operating System to create a protection system which ensures that a user who is running a particular program is authentic. Operating Systems generally identifies/authenticates users using following three ways −

- **Username / Password** − User need to enter a registered username and password with Operating system to login into the system.
- **User card/key** − User need to punch card in card slot, or enter key generated by key generator in option provided by operating system to login into the system.
- **User attribute - fingerprint/ eye retina pattern/ signature** − User need to pass his/her attribute via designated input device used by operating system to login into the system.

One Time passwords

One-time passwords provide additional security along with normal authentication. In One-Time Password system, a unique password is required every time user tries to login into the system. Once a one-time password is used, then it cannot be used again. One-time password are implemented in various ways.

- **Random numbers** − Users are provided cards having numbers printed along with corresponding alphabets. System asks for numbers corresponding to few alphabets randomly chosen.
- **Secret key** − User are provided a hardware device which can create a secret id mapped with user id. System asks for such secret id which is to be generated every time prior to login.
- **Network password** − Some commercial applications send one-time passwords to user on registered mobile/ email which is required to be entered prior to login.

Program Threats

Operating system's processes and kernel do the designated task as instructed. If a user program made these process do malicious tasks, then it is known as **Program Threats**. One of the common example of program threat is a program installed in a computer which can store and send user credentials via network to some hacker. Following is the list of some well-known program threats.

- **Trojan Horse** − Such program traps user login credentials and stores them to send to malicious user who can later on login to computer and can access system resources.
- **Trap Door** − If a program which is designed to work as required, have a security hole in its code and perform illegal action without knowledge of user then it is called to have a trap door.
- **Logic Bomb** − Logic bomb is a situation when a program misbehaves only when certain conditions met otherwise it works as a genuine program. It is harder to detect.
- **Virus** − Virus as name suggest can replicate themselves on computer system. They are highly dangerous and can modify/delete user files, crash systems. A virus is generatlly a small code embedded in a program. As user accesses the program, the virus starts getting embedded in other files/ programs and can make system unusable for user

System Threats

System threats refers to misuse of system services and network connections to put user in trouble. System threats can be used to launch program threats on a complete network called as program attack. System threats creates such an environment that operating system resources/ user files are misused. Following is the list of some well-known system threats.

- **Worm** − Worm is a process which can choked down a system performance by using system resources to extreme levels. A Worm process generates its multiple copies where each copy uses system resources, prevents all other processes to get required resources. Worms processes can even shut down an entire network.
- **Port Scanning** − Port scanning is a mechanism or means by which a hacker can detects system vulnerabilities to make an attack on the system.
- **Denial of Service** − Denial of service attacks normally prevents user to make legitimate use of the system. For example, a user may not be able to use internet if denial of service attacks browser's content settings.

Computer Security Classifications

As per the U.S. Department of Defense Trusted Computer System's Evaluation Criteria there are four security classifications in computer systems: A, B, C, and D.

This is widely used specifications to determine and model the security of systems and of security solutions. Following is the brief description of each classification.

S.N.	Classification Type & Description
1	**Type A** Highest Level. Uses formal design specifications and verification techniques. Grants a high degree of assurance of process security.
2	**Type B** Provides mandatory protection system. Have all the properties of a class C2 system. Attaches a sensitivity label to each object. It is of three types. • **B1** − Maintains the security label of each object in the system. Label is used • for making decisions to access control. • **B2** − Extends the sensitivity labels to each system resource, such as • storage objects, supports covert channels and auditing of events. • **B3** − Allows creating lists or user groups for access-control to grant access • or revoke access to a given named object.
3	**Type C** Provides protection and user accountability using audit capabilities. It is of two types. • **C1** − Incorporates controls so that users can protect their • private information and keep other users from accidentally reading / deleting their data. • UNIX versions are mostly Cl class. • **C2** − Adds an individual-level access control to the capabilities of a Cl level system.
4	**Type D** Lowest level. Minimum protection. MS-DOS, Window 3.1 fall in this category.

UNIT – 9

1 Give the functions of following UNIX commands:grep, cat, cmp, chmod ,finger,man Explain the following commands in UNIX: 7 suid, wall, man,finger,ls,cat,ps

cat

This command outputs the contents of a text file. You can use it to read brief files or to concatenate files together.

To append `file1` onto the end of `file2`, enter:

```
cat file1 >> file2
```

To view the contents of a file named `myfile`, enter:

```
cat myfile
```

Because `cat` displays text without pausing, its output may quickly scroll off your screen. Use the `less` command (described below) or an editor for reading longer text files.

chmod

This command changes the permission information associated with a file. Every file (including directories, which Unix treats as files) on a Unix system is stored with records indicating who has permission to read, write, or execute the file, abbreviated as r, w, and x. These permissions are broken down for three categories of user: first, the owner of the file; second, a group with which both the user and the file may be associated; and third, all other users. These categories are abbreviated as u for owner (or user), g for group, and o for other.

To allow yourself to execute a file that you own named `myfile`, enter:

```
chmod u+x myfile
```

To allow anyone who has access to the directory in which `myfile` is stored to read or execute `myfile`, enter:

```
chmod o+rx myfile
```

man

This command displays the manual page for a particular command. If you are unsure how to use a command or want to find out all its options, you might want to try using `man` to view the manual page.

For example, to learn more about the `ls` command, enter:

```
man ls
```

To learn more about `man`, enter:

```
man man
```

If you are not sure of the exact command name, you can use `man` with the `-k` option to help you find the command you need. To see one line summaries of each reference page that contains the keyword you specify, enter:

```
man -k keyword
```

ls

This command will list the files stored in a directory. To see a brief, multi-column list of the files in the current directory, enter:

```
ls
```

To also see "dot" files (configuration files that begin with a period, such as `.login`), enter:

```
ls -a
```

To see the file permissions, owners, and sizes of all files, enter:

```
ls -la
```

If the listing is long and scrolls off your screen before you can read it, combine `ls` with the `less` utility, for example:

```
ls -la | less
```

ps

The `ps` command displays information about programs (that is, processes) that are currently running. Entered without arguments, it lists basic information about interactive processes you own. However, it also has many options for determining what processes to display, as well as the amount of information about each. Like `lp` and `lpr`, the options available differ between BSD and System V implementations. For example, to view detailed information about all running processes, in a BSD system, you would use `ps` with the following arguments:

```
ps -alxww
```

To display similar information in System V, use the arguments:

```
ps -elf
```

For more about `ps` refer to the `ps` `man` page on your system. Also see About the output fields of the ps command in Unix.

finger command is a user information lookup command which gives details of all the users logged in. This tool is generally used by system administrators. It provides details like login name, user name, idle time, login time, and in some cases their email address even. This tool is similar to the Pinky tool but the Pinky tool is just the lightweight version of this tool.

Installing finger User Information Lookup Tool

To install *finger* tool use the following commands as per your Linux distribution.

In case of Debian/Ubuntu

```
$sudo apt-get install finger
```

In case of CentOS/RedHat

```
$sudo yum install finger
```

In case of Fedora OS

```
$sudo dnf install finger
```

2 What is “inode”? Explain File and Directory Management of Unix Operating System

The **inode** (index node) is a data structure in a Unix-style file system that describes a file-system object such as a file or a directory. Each inode stores the attributes and disk block locations of the object's data.

All data in Unix is organized into files. All files are organized into directories. These directories are organized into a tree-like structure called the filesystem. When you work with Unix, one way or another, you spend most of your time working with files.

3 Explain Unix Commands: cat, sort, grep

Unix Command: Cat

It is used to create, display and concatenate file contents.

Syntax: `cat [options] [FILE]...`

Example: `$ cat file1.c`
`Above syntax will display the content of file1.c`
Example: `$ cat > file1.c`
`Above syntax creates file1.c and allow us to insert content for this file.`

- **After inserting content you can use ctrl+d to exit the file.**
- **If file with same name exist then it will overwrite that file.**

Example: `$ cat file1.c >> file2.c`
`It can concatenate the contents of two files. For this you have to use append output redirection operator.`
`The contents of file2.cwill be appended to file1.c.`

Unix Command: Sort

It is used to sort the lines in a text file.

Syntax: `sort [FILE]...`

Example:
$ cat > data.txt
```
apples
oranges
pears
kiwis
bananas
```

$ sort data.txt `apples`
```
bananas
kiwis
oranges
pears
```

Unix Command: grep

It selects and prints the lines from a file which matches a given string or pattern.

Syntax: `grep [options] pattern [file]`

Description

- **This command searches the specified input fully for a match with the supplied pattern and displays it.**
- **While forming the patterns to be searched we can use shell match characters, or regular expressions.**

Example: `$ grep "Error" logfile.txt`
`This searches for the string "Error" in the log file and prints all the lines that has the word "Error".`

4 Explain Linux kernel and its functions in brief.

The Linux® kernel is the main component of a Linux operating system (OS) and is the core interface between a computer's hardware and its processes. It communicates between the 2, managing resources as efficiently as possible.

What the kernel does

The kernel has 4 jobs:

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

The kernel, if implemented properly, is invisible to the user, working in its own little world known as kernel space, where it allocates memory and keeps track of where everything is stored. What the user sees—like web browsers and files—are known as the user space. These applications interact with the kernel through a system call interface (SCI).

Think about it like this. The kernel is a busy personal assistant for a powerful executive (the hardware). It's the assistant's job to relay messages and requests (processes) from employees and the public (users) to the executive, to remember what is stored where (memory), and to determine who has access to the executive at any given time and for how long.

To put the kernel in context, you can think of a Linux machine as having 3 layers:

1. **The hardware:** The physical machine—the bottom or base of the system, made up of memory (RAM) and the processor or central processing unit (CPU), as well as input/output (I/O) devices such as storage, networking, and graphics. The CPU performs computations and reads from, and writes to, memory.
2. **The Linux kernel:** The core of the OS. (See? It's right in the middle.) It's software residing in memory that tells the CPU what to do.
3. **User processes:** These are the running programs that the kernel manages. User processes are what collectively make up user space. User processes are also known as just *processes*. The kernel also allows these processes and servers to communicate with each other (known as inter-process communication, or IPC).

Code executed by the system runs on CPUs in 1 of 2 modes: kernel mode or user mode. Code running in the kernel mode has unrestricted access to the hardware, while user mode restricts access to the CPU and memory to the SCI. A similar separation exists for memory (kernel space and user space). These 2 small details form the base for some complicated operations like privilege separation for security, building containers, and virtual machines.

This also means that if a process fails in user mode, the damage is limited and can be recovered by the kernel. However, because of its access to memory and the processor, a kernel process crash can crash the entire system. Since there are safeguards in place and permissions required to cross boundaries, user process crashes usually can't cause too many problems.