Interview Questions

-Even if it's not covered as of now, still read about it and prepare.
----

(1). Define the types of Operating System.

An Operating System performs all the basic tasks like managing files, processes, and memory. Thus, the operating system acts as the manager of all the resources, i.e. **resource manager**. Thus, the operating system becomes an interface between the user and the machine. It is one of the most required software that is present in the device.

1. Batch Operating System:

This type of operating system does not interact with the computer directly. There is an operator which takes similar jobs having the same requirements and groups them into batches. It is the responsibility of the operator to sort jobs with similar needs. Batch Operating System is designed to manage and execute a large number of jobs efficiently by processing them in groups.

2. Multi- programming Operating System:

Multiprogramming Operating Systems can be simply illustrated as more than one program is present in the main memory and any one of them can be kept in execution. This is basically used for better utilization of resources.

3. Multi-Processing Operating System:

Multi-Processing Operating System is a type of Operating System in which more than one CPU is used for the execution of resources. It betters the throughput of the System.

4. Multi-Tasking Operating System:

Multitasking Operating System is simply a multiprogramming Operating System with the facility of a Round-Robin Scheduling Algorithm. It can run multiple programs simultaneously.

There are two types of Multi-Tasking Systems which are listed below.

- Preemptive Multi-Tasking
- Cooperative Multi-Tasking

What is Preemptive Multitasking?

In **preemptive multitasking**, the operating system can initiate a context switching from the running process to another process. In other words, the operating system allows stopping the execution of the currently running process and allocating the CPU to some other process. The OS uses some criteria to decide for how long a process should execute before allowing another process to use the operating system. The mechanism of taking control of the operating system from one process and giving it to another process is called preempting or preemption.

What is Cooperative Multitasking?

In **cooperative multitasking**, the operating system never initiates context switching from the running process to another process. A context switch occurs only when the processes voluntarily yield control periodically or when idle or logically blocked to allow multiple applications to execute simultaneously. Also, in this multitasking, all the processes cooperate for the scheduling scheme to work.

(2). Explain DHCP.

Dynamic Host Configuration Protocol is a network protocol used to automate the process of assigning IP addresses and other network configuration parameters to devices (such as computers, smartphones, and printers) on a network. Instead of manually configuring each device with an IP address, DHCP allows devices to connect to a network and receive all necessary network information, like IP address, subnet mask, default gateway, and DNS server addresses, automatically from a DHCP server.

This makes it easier to manage and maintain large networks, ensuring devices can communicate effectively without conflicts in their network settings. DHCP plays a crucial role in modern networks by simplifying the process of connecting devices and managing network resources efficiently.

What is DHCP?

DHCP stands for Dynamic Host Configuration Protocol. It is the critical feature on which the users of an enterprise network communicate. DHCP helps enterprises to smoothly manage the allocation of IP addresses to the end-user clients' devices such as desktops, laptops, cellphones, etc. is an application layer protocol that is used to provide:

Subnet Mask (Option 1 - e.g., 255.255.255.0)

Router Address (Option 3 - e.g., 192.168.1.1)

DNS Address (Option 6 - e.g., 8.8.8.8)

Vendor Class Identifier (Option 43 - e.g.,

'unifi' = 192.168.1.9 ##where unifi = controller)

DHCP is based on a client-server model and based on discovery, offer, request, and ACK.

Why Do We Use DHCP?

DHCP helps in managing the entire process automatically and centrally. DHCP helps in maintaining a unique IP Address for a host using the server. DHCP servers maintain information on TCP/IP configuration and provide configuration of address to DHCP-enabled clients in the form of a lease offer.

Components of DHCP:

The main components of DHCP include:

- **DHCP Server:** DHCP Server is a server that holds IP Addresses and other information related to configuration.
- **DHCP Client:** It is a device that receives configuration information from the server. It can be a mobile, laptop, computer, or any other electronic device that requires a connection.
- **DHCP Relay:** DHCP relays basically work as a communication channel between DHCP Client and Server.
- **IP Address Pool:** It is the pool or container of IP Addresses possessed by the DHCP Server. It has a range of addresses that can be allocated to devices.
- **Subnets:** Subnets are smaller portions of the IP network partitioned to keep networks under control.

- **Lease:** It is simply the time that how long the information received from the server is valid, in case of expiration of the lease, the tenant must have to re-assign the lease.
- **DNS Servers:** DHCP servers can also provide DNS (Domain Name System) server information to DHCP clients, allowing them to resolve domain names to IP addresses.
- **Default Gateway:** DHCP servers can also provide information about the default gateway, which is the device that packets are sent to when the destination is outside the local network.
- **Options:** DHCP servers can provide additional configuration options to clients, such as the subnet mask, domain name, and time server information.
- **Renewal:** DHCP clients can request to renew their lease before it expires to ensure that they continue to have a valid IP address and configuration information.
- **Failover:** DHCP servers can be configured for failover, where two servers work together to provide redundancy and ensure that clients can always obtain an IP address and configuration information, even if one server goes down.
- **Dynamic Updates:** DHCP servers can also be configured to dynamically update DNS records with the IP address of DHCP clients, allowing for easier management of network resources.
- **Audit Logging:** DHCP servers can keep audit logs of all DHCP transactions, providing administrators with visibility into which devices are using which IP addresses and when leases are being assigned or renewed.

(3). Explain DNS.

A DNS server is a computer with a database containing the public IP addresses associated with the names of the websites an IP address brings a user to. DNS acts like a phonebook for the internet. Whenever people type domain names, like Fortinet.com or Yahoo.com, into the address bar of web browsers, the DNS finds the right IP address. The site’s IP address is what directs the device to go to the correct place to access the site’s data.

Once the DNS server finds the correct IP address, browsers take the address and use it to send data to content delivery network (CDN) edge servers or origin servers. Once this is done, the information on the website can be accessed by the user. The DNS server starts the process by finding the corresponding IP address for a website’s uniform resource locator (URL).

How Does DNS Work?

In a usual DNS query, the URL typed in by the user has to go through four servers for the IP address to be provided. The four servers work with each other to get the correct IP address to the client, and they include:

1. **DNS recursor**: The DNS recursor, which is also referred to as a DNS resolver, receives the query from the DNS client. Then it communicates with other DNS servers to find the right IP address. After the resolver retrieves the request from the client, the resolver acts like a client itself. As it does this, it makes queries that get sent to the other three DNS servers: root name servers, top-level domain (TLD) nameservers, and authoritative nameservers.
2. **Root name servers**: The root nameserver is designated for the internet's DNS root zone. Its job is to answer requests sent to it for records in the root zone. It answers requests by sending back a list of the authoritative nameservers that go with the correct TLD.
3. **TLD nameservers**: A TLD name server keeps the IP address of the second-level domain contained within the TLD name. It then releases the website’s IP address and sends the query to the domain’s nameserver.
4. **Authoritative nameservers**: An authoritative nameserver is what gives you the real answer to your DNS query. There are two types of authoritative nameservers: a master server or primary name server and a slave server or secondary nameserver. The master server keeps the original copies of the zone records, while the slave server is an exact copy of the master server. It shares the DNS server load and acts as a backup if the master server fails.

(4). Explain Paging.

Paging is a memory management scheme that eliminates the need for a contiguous allocation of physical memory. The process of retrieving processes in the form of pages from the secondary storage into the main memory is known as **paging**. The basic purpose of paging is to separate each procedure into pages. Additionally, frames will be used to split the main memory. This scheme permits the physical address space of a process to be non – contiguous.

In paging, the physical memory is divided into fixed-size blocks called **page frames**, which are the same size as the pages used by the process. The process's logical address space is also divided into fixed-size blocks called pages, which are the same size as the page frames. When a process requests memory, the operating system allocates one or more page frames to the process and maps the process's **logical pages** to the physical page frames.

The mapping between logical pages and physical page frames is maintained by the page table, which is used by the **memory management unit** to translate logical addresses into physical addresses. The page table maps each logical page number to a physical page frame number.

Important Points About Paging in Operating Systems:

- **Reduces internal fragmentation:** Paging facilitates lessening internal fragmentation by allocating memory in fixed-size blocks (pages), which might be usually a whole lot smaller than the size of the process's facts segments. This lets in for greater efficient use of memory in view that there are fewer unused bytes in each block.
- **Enables reminiscence to be allotted on call for:** Paging enables memory to be allocated on call for, this means that memory is most effectively allocated when it's far needed. This allows for extra efficient use of memory in view that only the pages that are absolutely used by the manner want to be allocated inside the physical memory.
- **Protection and sharing of memory:** Paging allows for the protection and sharing of reminiscence between methods, as each procedure has its own web page table that maps its logical deal with area to its physical address

space. This permits techniques to proportion facts at the same time as preventing unauthorized get right of entry to every other memory.

- **External fragmentation:** Paging can result in outside fragmentation, wherein memory turns fragmented into small, non-contiguous blocks. This can make it difficult to allocate massive blocks of reminiscence to a method seeing that there may not be enough contiguous free memory to be had.
- **Overhead:** Paging involves overhead because of the renovation of the web page table and the translation of logical addresses to physical addresses. The working device must maintain the page table for each manner and perform a deal with translation whenever a procedure accesses memory, which can slow down the machine.

(5). Explain Segmentation.

A process is divided into Segments. The chunks that a program is divided into which are not necessarily all of the exact sizes are called segments. Segmentation gives the user's view of the process which paging does not provide. Here the user's view is mapped to physical memory.

Types of Segmentation in Operating Systems:

- **Virtual Memory Segmentation:** Each process is divided into a number of segments, but the segmentation is not done all at once. This segmentation may or may not take place at the run time of the program.
- **Simple Segmentation:** Each process is divided into a number of segments, all of which are loaded into memory at run time, though not necessarily contiguously.

There is no simple relationship between logical addresses and physical addresses in segmentation. A table stores the information about all such segments and is called a Segment Table.

Advantages of Segmentation in Operating System:

- **Reduced Internal Fragmentation**: Segmentation can reduce internal fragmentation compared to fixed-size paging, as segments can be sized according to the actual needs of a process. However, internal fragmentation can still occur if a segment is allocated more space than it is actually used.
- Segment Table consumes less space in comparison to Page table in paging.
- As a complete module is loaded all at once, segmentation improves CPU utilization.
- The user’s perception of physical memory is quite similar to segmentation. Users can divide user programs into modules via segmentation. These modules are nothing more than separate processes’ codes.
- The user specifies the segment size, whereas, in paging, the hardware determines the page size.
- Segmentation is a method that can be used to segregate data from security operations.
- **Flexibility:** Segmentation provides a higher degree of flexibility than paging. Segments can be of variable size, and processes can be designed to have multiple segments, allowing for more fine-grained memory allocation.
- **Sharing:** Segmentation allows for sharing of memory segments between processes. This can be useful for inter-process communication or for sharing code libraries.
- **Protection:** Segmentation provides a level of protection between segments, preventing one process from accessing or modifying another process’s memory segment. This can help increase the security and stability of the system.

Disadvantages of Segmentation in Operating System:

- **External Fragmentation**: As processes are loaded and removed from memory, the free memory space is broken into little pieces, causing external fragmentation. This is a notable difference from paging, where external fragmentation is significantly lesser.

- Overhead is associated with keeping a segment table for each activity.
- Due to the need for two memory accesses, one for the segment table and the other for main memory, access time to retrieve the instruction increases.
- **Fragmentation:** As mentioned, segmentation can lead to external fragmentation as memory becomes divided into smaller segments. This can lead to wasted memory and decreased performance.
- **Overhead:** Using a segment table can increase overhead and reduce performance. Each segment table entry requires additional memory, and accessing the table to retrieve memory locations can increase the time needed for memory operations.
- **Complexity:** Segmentation can be more complex to implement and manage than paging. In particular, managing multiple segments per process can be challenging, and the potential for segmentation faults can increase as a result.

(6). Explain Memory management.

In a multiprogramming computer, the Operating System resides in a part of memory, and the rest is used by multiple processes. The task of subdividing the memory among different processes is called Memory Management. Memory management is a method in the operating system to manage operations between main memory and disk during process execution. The main aim of memory management is to achieve efficient utilization of memory.

Why is Memory Management Required?

- Allocate and deallocate memory before and after process execution.
- To keep track of used memory space by processes.
- To minimize fragmentation issues.
- To proper utilization of main memory.

- To maintain data integrity while executing the process.

Now we are discussing the concept of Logical Address Space and Physical Address Space

Logical and Physical Address Space:

- **Logical Address Space:** An address generated by the CPU is known as a "Logical Address". It is also known as a Virtual address. Logical address space can be defined as the size of the process. A logical address can be changed.
- **Physical Address Space:** An address seen by the memory unit (i.e the one loaded into the memory address register of the memory) is commonly known as a "Physical Address". A Physical address is also known as a Real address. The set of all physical addresses corresponding to these logical addresses is known as Physical address space. A physical address is computed by MMU. The run-time mapping from virtual to physical addresses is done by a hardware device Memory Management Unit(MMU). The physical address always remains constant.

(7). Explain the function of the OS.

An operating system (OS) has many functions, including:

- Booting: The OS loads into the computer's main memory when the computer is turned on.
- Managing resources: The OS manages the computer's resources, such as the CPU, memory, storage devices, and printer.
- Managing files: The OS stores, organizes, and manages files, including their creation, deletion, and security.
- Managing processes: The OS manages all running programs, scheduling them, allocating resources, and ensuring they run without interfering with each other.
- Managing devices: The OS handles communication with hardware devices, such as printers, keyboards, and network adapters.

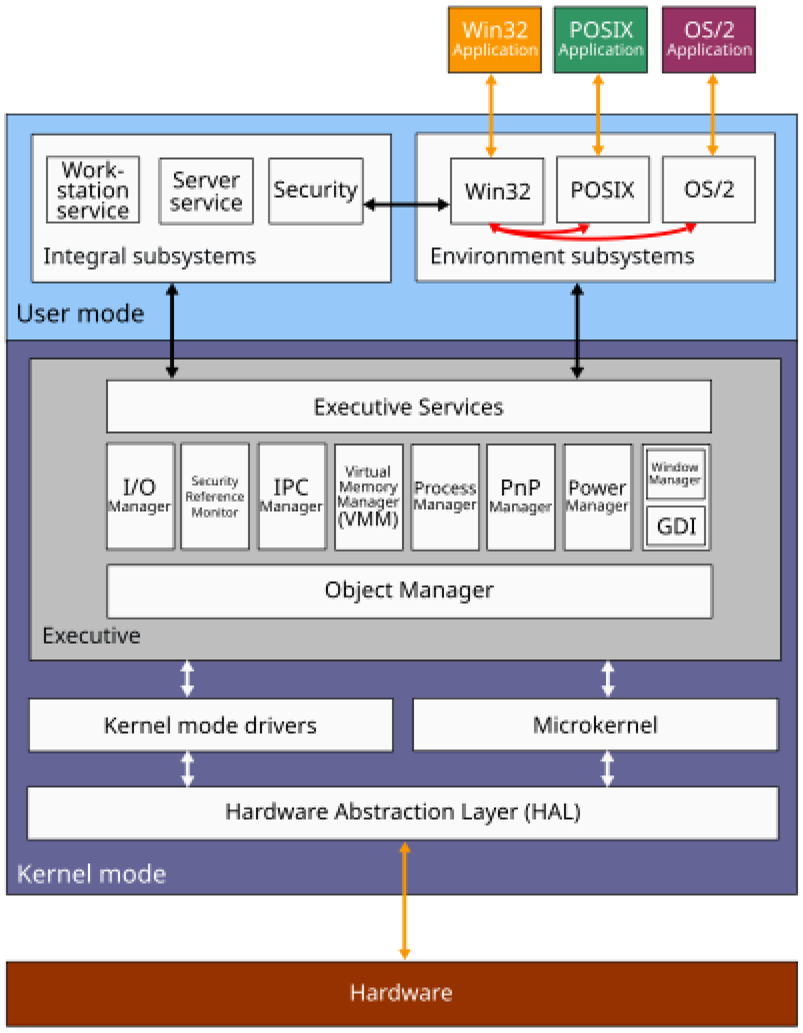
- Providing a user interface: The OS provides a user-friendly interface, such as a graphical user interface (GUI) or command-line interface (CLI).
- Ensuring security: The OS enforces user authentication and access rights, and protects data and files from unauthorized access.
- Monitoring system performance: The OS monitors system health and performance, and records response time between service requests and system response.
- Tracking resources: The OS tracks the time and resources used by users, which can be used to collect individual data.
- Detecting errors: The OS continuously monitors for errors and avoids computer malfunctions.

(8). Explain a kernel, its architecture and working.

A kernel is a central component of an operating system that manages the operations of computers and hardware. It basically manages operations of memory and CPU time. It is a core component of an operating system. Kernel acts as a bridge between applications and data processing performed at the hardware level using inter-process communication and system calls.

What is Kernel?

A kernel is the core part of an operating system. It acts as a bridge between software applications and the hardware of a computer. The kernel manages system resources, such as the CPU, memory, and devices, ensuring everything works together smoothly and efficiently. It handles tasks like running programs, accessing files, and connecting to devices like printers and keyboards.



The architecture of Windows NT, a line of operating systems produced and sold by Microsoft, is a layered design that consists of two main components, user mode and kernel mode. It is a preemptive, reentrant multitasking operating system, which has been designed to work with uniprocessor and symmetrical multiprocessor (SMP)-based computers. To process input/output (I/O) requests, it uses packet-driven I/O, which utilizes I/O request packets (IRPs) and asynchronous I/O. Starting with Windows XP, Microsoft began making 64-bit versions of Windows available; before this, there were only 32-bit versions of these operating systems.

Programs and subsystems in user mode are limited in terms of what system resources they have access to, while the kernel mode has unrestricted access to the system memory and external devices. Kernel mode in Windows NT has full access to the hardware and system resources of the computer. The Windows NT kernel is a hybrid kernel; the architecture comprises a simple kernel, hardware abstraction layer (HAL), drivers, and a range of services (collectively named Executive), which all exist in kernel mode.[1]

User mode in Windows NT is made of subsystems capable of passing I/O requests to the appropriate kernel mode device drivers by using the I/O manager. The user mode layer of Windows NT is made up of the "Environment subsystems", which run applications written for many different types of operating systems, and the "Integral subsystem", which operates system-specific functions on behalf of environment subsystems. The kernel mode stops user mode services and applications from accessing critical areas of the operating system that they should not have access to.

The Executive interfaces, with all the user mode subsystems, deal with I/O, object management, security and process management. The kernel sits between the hardware abstraction layer and the Executive to provide *multiprocessor synchronization*, thread and interrupt scheduling and dispatching, and trap handling and exception dispatching. The kernel is also responsible for initializing device drivers at bootup. Kernel mode drivers exist in three levels: highest level drivers, intermediate drivers and low-level drivers. Windows Driver Model (WDM) exists in the intermediate layer and was mainly designed to be binary and source compatible between Windows 98 and Windows 2000. The lowest level drivers are either legacy Windows NT device drivers that control a device directly or can be a plug and play (PnP) hardware bus.

(9). Explain a shell script.

A shell script is a text file that contains a sequence of commands for a UNIX-based operating system. It is called a shell script because it combines a sequence of commands, that would otherwise have to be typed into the keyboard one at a time, into a single script. The shell is the operating system's command-line interface (CLI) and interpreter for the set of commands that are used to communicate with the system. A shell script is usually created for command sequences in which a user has a need to use repeatedly in order to save time. Like other programs, the shell script can contain parameters, comments and subcommands that the shell must follow. Users initiate the sequence of commands in the shell script by simply entering the file name on a command line.

In the DOS operating system, a shell script is called a batch file. In IBM's mainframe VM operating systems, it's called an EXEC.

How does shell scripting work ?

The basic steps involved with shell scripting are writing the script, making the script accessible to the shell and giving the shell execute permission.

Shell scripts contain ASCII text and are written using a text editor, word processor or graphical user interface (GUI). The content of the script is a series of commands in a language that can be interpreted by the shell. Functions that shell scripts support include loops, variables, if/then/else statements, arrays and shortcuts. Once complete, the file is saved typically with a .txt or .sh extension and in a location that the shell can access.

Types of shells:

In Unix and Linux, the two major types of shell scripts are:

(1). Bourne again shells (BASH)- BASH is the default shell for Unix version 7. The character for prompting a bourne again shell is $.

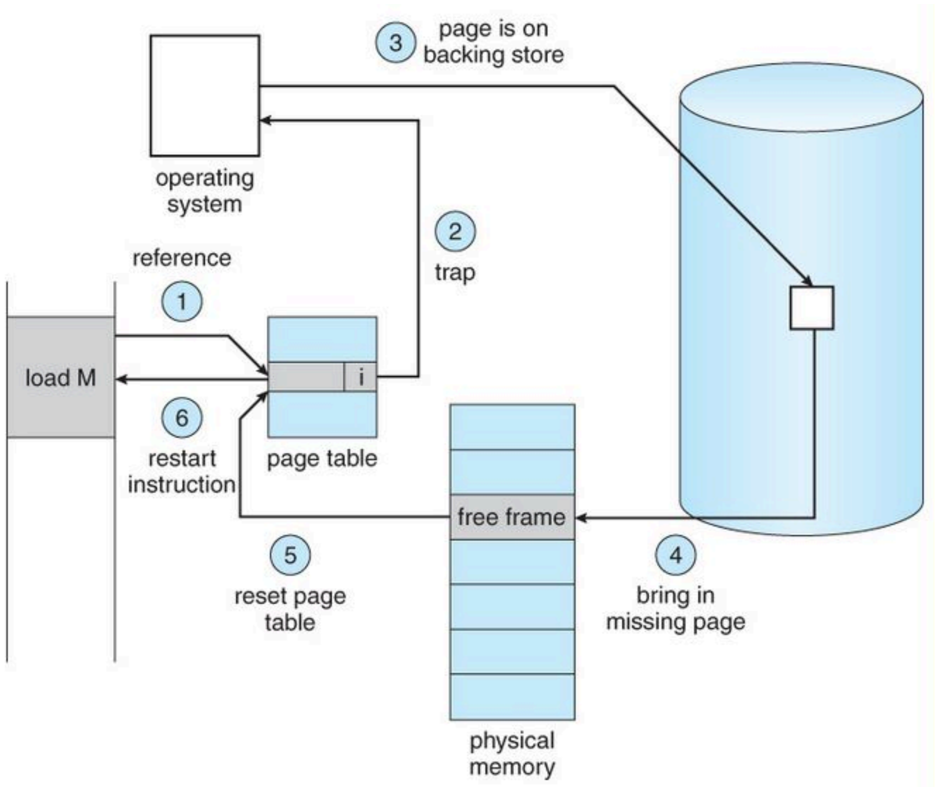
(2). C shells- A C shell is run in a text terminal window and is able to easily read file commands. The character for prompting a C shell is %.

(10). Explain a Page fault.

A page fault occurs when a program attempts to access data or code that is in its address space but is not currently located in the system RAM. This triggers a sequence of events where the operating system must manage the fault by loading the required data from secondary storage into RAM. Page faults are essential for implementing virtual memory systems that provide the illusion of a larger contiguous memory space.

What is Page Fault?

A page fault is a critical event in computer systems in which a program attempts to access data or code that is not currently available in the physical memory (main memory). This occurrence is an integral part of virtual memory management which allows the system to physically fit large amounts of data efficiently in memory.



- The computer hardware traps the kernel and the program counter (PC) is saved on the stack. Current instruction state information is saved in CPU registers.
- An assembly program is started to save the general registers and other volatile information to keep the OS from destroying it.
- Operating system finds that a page fault has occurred and tries to find out which virtual page is needed. Sometimes the hardware register contains this required information. If not, the operating system must retrieve the PC, fetch instructions and find out what it was doing when the fault occurred.
- Once the virtual address caused page fault is known, the system checks to see if the address is valid and checks if there is no protection access problem.

- If the virtual address is valid, the system checks to see if a page frame is free. If no frames are free, the page replacement algorithm is run to remove a page.
- If the frame selected is dirty, the page is scheduled for transfer to disk, context switch takes place, fault process is suspended and another process is made to run until disk transfer is completed.
- As soon as the page frame is clean, the operating system looks up the disk address where the needed page is, schedules disk operation to bring it in.
- When disk interrupt indicates page has arrived, page tables are updated to reflect its position, and frame marked as being in normal state.
- Faulting instruction is backed up to state it had when it began and the PC is reset. Faulting is scheduled, the operating system returns to the routine that called it.
- Assembly Routine reloads register and other state information, returns to user space to continue execution.

Causes of Page Faults:

There are several reason of causing Page faults:

- **Demand Paging:** Accessing the page that is not currently loaded in the memory(RAM).
- **Invalid Memory Access:** occurs when a program tries to access that memory which is beyond access boundaries or not allocated .
- **Process Violation:** when a process tries to write to a read-only page or otherwise violates memory protection rules.

Types of Page Fault:

- **Minor Page Fault:** Occurs when the required page is in memory but not in the current process’s page table.

- **Major Page Fault:** Occurs when the page is not in memory and must be fetched from disk.
- **Invalid Page Fault:** It happens when the process tries to access an invalid memory address.

Page Fault Handling Process:

- **Page Fault Detection:** Through a hardware interrupt the Operating system detects the page fault.
- **Address Validation:** The OS checks if the page is present in the secondary storage or checks whether the memory is valid or not.
- **Page Replacement:** If there is no space for the page that is not in the memory yet, then the OS decides which page is removed or evicted from memory to make room for the new page.
- **Loading the Page:** The required page is loaded from disk into memory.
- **Updating Page Tables:** The OS updates the page table with the new mapping.
- **Restarting the Process:** When any page fault occurs then the process will restart.

Impact of Page Faults or System Performance:

Page Fault impact the system if it occurs frequently

- **Thrashing:** If occurrence of page fault is frequent then the system spends more time to handle it than executing the processes, and because of which overall performance also degrades.
- **Increased Latency:** Fetching pages from disk takes more time than accessing them in memory, which causes more delays.
- **CPU Utilization:** If the Page fault occurs excessively then it can reduce CPU Utilization as the processor waits for memory operations to complete or remain idle which is not efficient.

Conclusion:

In an operating system, page fault handling is an essential process that ensures smooth and efficient memory management. When a program tries to access data that isn't currently in the computer's physical memory (RAM), a page fault occurs. The operating system steps in to retrieve the necessary data from a slower storage (like a hard drive) and loads it into RAM. Although page faults can temporarily slow down a program, they are a normal part of how virtual memory works, allowing computers to run large applications efficiently without needing an excessive amount of physical memory. Effective page fault handling helps the system maintain performance and stability.

(11). Explain a deadlock.

A deadlock is a situation where a set of processes is blocked because each process is holding a resource and waiting for another resource acquired by some other process. In this article, we will discuss deadlock, its necessary conditions, etc. in detail.

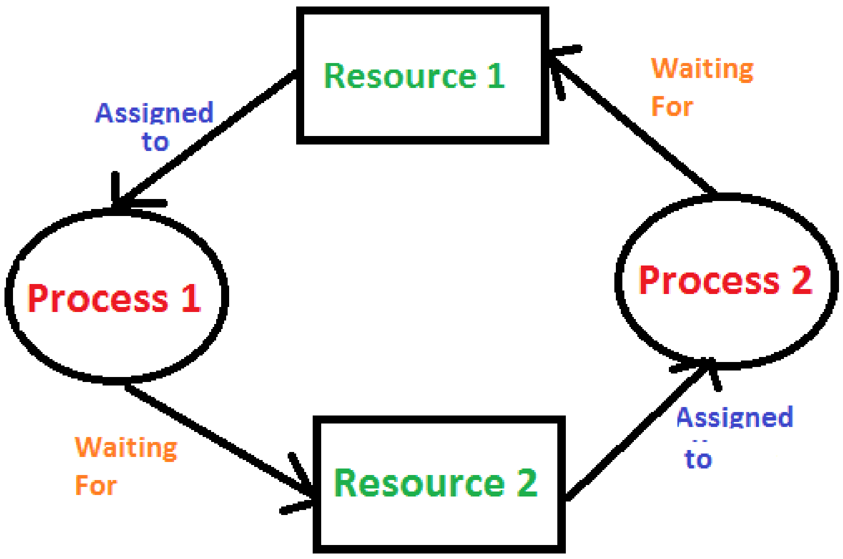
What is Deadlock?

Deadlock is a situation in computing where two or more processes are unable to proceed because each is waiting for the other to release resources. Key concepts include mutual exclusion, resource holding, circular wait, and no preemption.

Consider an example when two trains are coming toward each other on the same track and there is only one track, none of the trains can move once they are in front of each other. This is a practical example of deadlock.

How Does Deadlock occur in the Operating System?
Before going into detail about how deadlock occurs in the Operating System, let's first discuss how the Operating System uses the resources present. A process in an operating system uses resources in the following way.

- Requests a resource
- Use the resource
- Releases the resource



What is Deadlock Detection?

Deadlock detection is a process in computing where the system checks if there are any sets of processes that are stuck waiting for each other indefinitely, preventing them from moving forward. In simple words, deadlock detection is the process of finding out whether any process is stuck in a loop or not. There are several algorithms like

- Resource Allocation Graph
- Banker's Algorithm

These algorithms help in detection of deadlock in Operating Systems.

(12). Define the necessary conditions for deadlock.

Necessary Conditions for Deadlock in OS:

Deadlock can arise if the following four conditions hold simultaneously (Necessary Conditions)

- **Mutual Exclusion:** Two or more resources are non-shareable (Only one process can use at a time).
- **Hold and Wait:** A process is holding at least one resource and waiting for resources.
- **No Preemption:** A resource cannot be taken from a process unless the process releases the resource.
- **Circular Wait:** A set of processes waiting for each other in circular form.

(13). Explain a semaphore.

A semaphore is a variable in an operating system that controls access to a shared resource, such as a file or database, by multiple processes. It works by keeping track of how many units of a resource are available, and allowing processes to check and change the value of the semaphore. Depending on the value, a process can either use the resource or wait for it to become available.

Semaphore type:

Semaphore type	**Description**
Counting semaphore	Allows an arbitrary resource count
Binary semaphore	Restricted to the values 0 and 1, or locked/unlocked, unavailable/availa

	ble

Semaphores are used to: Prevent data corruption or inconsistency, Manage memory resources, Ensure process synchronization, and Protect against race conditions.

Dutch computer scientist Edsger Dijkstra invented the semaphore concept in 1962 or 1963.

(14). Explain a Mutex.

In the Operating System, **Mutexes and Semaphores** are kernel resources that provide synchronization services (also known as synchronization primitives). Synchronization is required when multiple processes are executing concurrently, to avoid conflicts between processes using shared resources.

Mutex:

Mutex is a specific kind of binary semaphore that is used to provide a locking mechanism. It stands for Mutual Exclusion Object. Mutex is mainly used to provide mutual exclusion to a specific portion of the code so that the process can execute and work with a particular section of the code at a particular time.

Mutex uses a priority inheritance mechanism to avoid priority inversion issues. The priority inheritance mechanism keeps higher-priority processes in the blocked state for the minimum possible time. However, this cannot avoid the priority inversion problem, but it can reduce its effect up to an extent.

Advantages of Mutex:

- No race condition arises, as only one process is in the critical section at a time.
- Data remains consistent and it helps in maintaining integrity.

- It’s a simple locking mechanism that can be obtained by a process before entering into a critical section and released while leaving the critical section.

Disadvantages of Mutex:

- If after entering into the critical section, the thread sleeps or gets preempted by a high-priority process, no other thread can enter into the critical section. This can lead to starvation.
- When the previous thread leaves the critical section, then only other processes can enter into it, there is no other mechanism to lock or unlock the critical section.
- Implementation of mutex can lead to busy waiting that leads to the wastage of the CPU cycle.

(15). Difference among kernel space and user space.

Kernel space and user space are different areas of computer memory that serve different purposes:

Kernel space:
This area is reserved for the operating system's core functions, such as the kernel, kernel extensions, and most device drivers. The kernel manages the system's resources, like the CPU, memory, and storage. It also provides system calls that allow user space applications to interact with the kernel. Kernel space is protected by security features that guard against malware and other threats.
User space:
This area is where application software and non-kernel processes run. It includes utilities, programming languages, graphical tools, and applications written in various languages. User space can also be used to communicate between programs, where one program loads information into a user space and then passes it to another program.

The separation between kernel space and user space is important for maintaining system stability and security

(16). Write in brief the ping command.

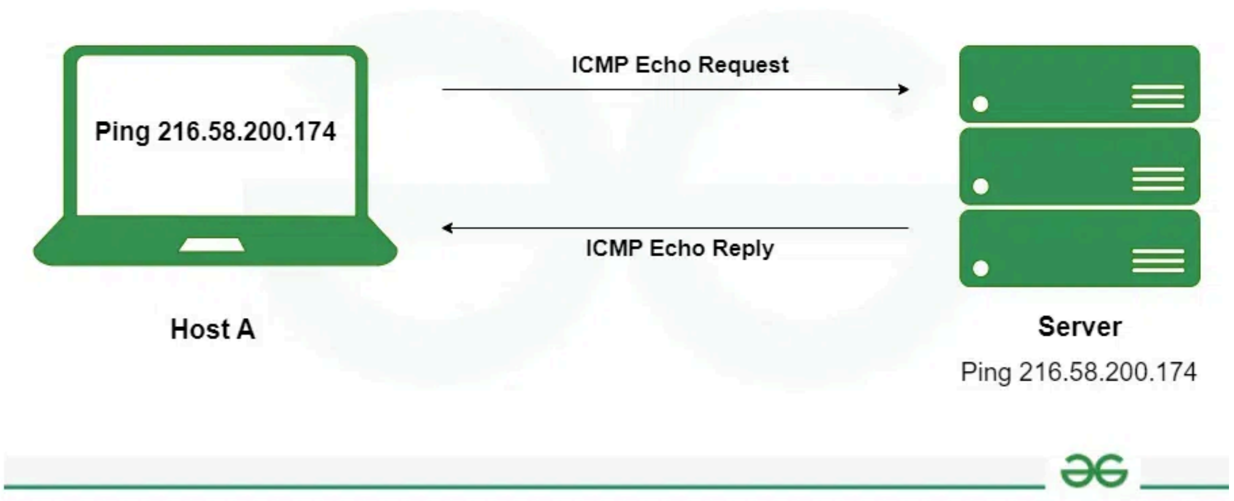
A ping is a basic Internet command that allows a user to test and verify whether a given destination IP address exists and can accept requests in computer network administration. Ping is also used for diagnosis to confirm that the computer the user tries to reach is operational. Ping can be used with any operating system (OS) that supports networking, including the majority of embedded network administration software.

What is Ping?

Ping (Packet Internet Groper) is a method for determining communication latency between two networks or ping is a method of determining the time it takes for data to travel between two devices or across a network. As communication latency decreases, communication effectiveness improves. A low ping time is critical in situations where the timely delivery of data is more important than the quantity and quality of the desired information.

How Does Ping Work?

Ping sends an Internet Control Message Protocol (ICMP) Echo Request to a network interface and then waits for a response. When the ping command is executed, a ping signal is delivered to the provided address. When the target host receives the echo request, it answers with an echo reply packet. This method has two distinct purposes: calculating round-trip time (RTT) or latency and ensuring that the target host is available. RTT is a measure of the time it takes to receive a response. Measured in milliseconds (ms), the process begins when a browser submits a request to a server and concludes when the server responds. RTT is an important performance figure for online applications.



(17). Explain UNIX.

UNIX is an innovative or groundbreaking operating system which was developed in the 1970s by Ken Thompson, Dennis Ritchie, and many others at AT&T Laboratories. It is like a backbone for many modern operating systems like Ubuntu, Solaris, Kali Linux, Arch Linux, and also POSIX. Originally, It was designed for developers only, UNIX played a most important role in the development and creation of the software and computing environments. Its distribution to government and academic institutions led to its widespread adoption across various types of hardware components. The core part of the UNIX system lies in its base Kernel, which is integral to its architecture, structure, and key functionality making it the heart of the operating system.
The basic design philosophy of UNIX is to provide simple, powerful tools that can be combined to perform complex tasks. It features a command-line interface that allows users to interact with the system through a series of commands, rather than through a graphical user interface (GUI).

Some of the Key Features of UNIX Include:

1. **Multiuser support:** UNIX allows multiple users to simultaneously access the same system and share resources.

2. **Multitasking:** UNIX is capable of running multiple processes at the same time.
3. **Shell scripting:** UNIX provides a powerful scripting language that allows users to automate tasks.
4. **Security:** UNIX has a robust security model that includes file permissions, user accounts, and network security features.
5. **Portability:** UNIX can run on a wide variety of hardware platforms, from small embedded systems to large mainframe computers.
6. **Communication:** UNIX supports communication methods using the write command, mail command, etc.
7. **Process Tracking:** UNIX maintains a record of the jobs that the user creates. This function improves system performance by monitoring CPU usage. It also allows you to keep track of how much disk space each user uses, and the use that information to regulate disk space.

Today, UNIX is widely used in enterprise-level computing, scientific research, and web servers. Many modern operating systems, including Linux and macOS, are based on UNIX or its variants.

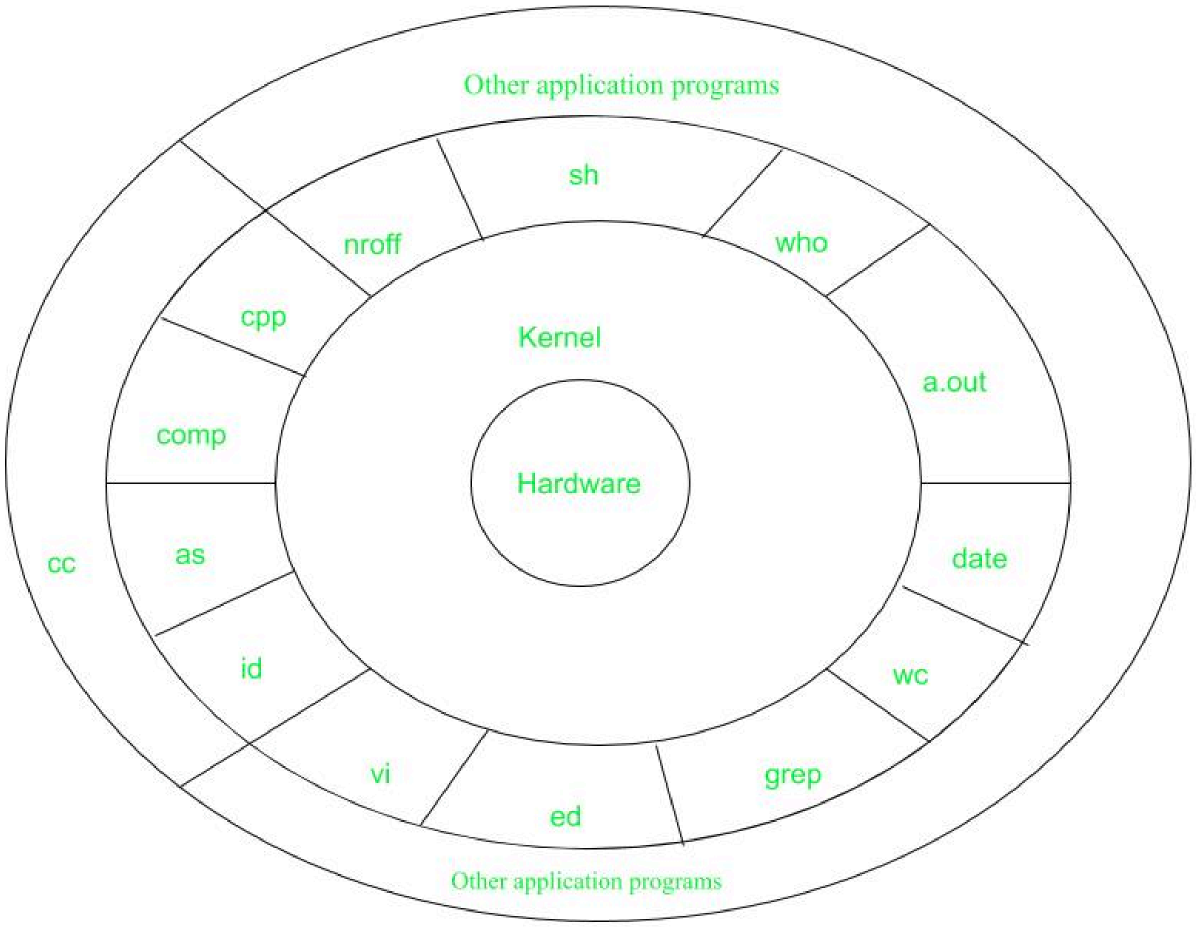


Figure – System Structure

- **Layer-1: Hardware:** It consists of all hardware related information.
- **Layer-2: Kernel:** This is the core of the Operating System. It is a software that acts as the interface between the hardware and the software. Most of the tasks like memory management, file management, network management, process management, etc., are done by the kernel.
- **Layer-3: Shell commands:** This is the interface between the user and the kernel. Shell is the utility that processes your requests. When you type in a command at the terminal, the shell interprets the command and calls the program that you want. There are various commands like cp, mv, cat, grep, id, wc, nroff, a.out and more.
- **Layer-4: Application Layer:** It is the outermost layer that executes the given external applications.

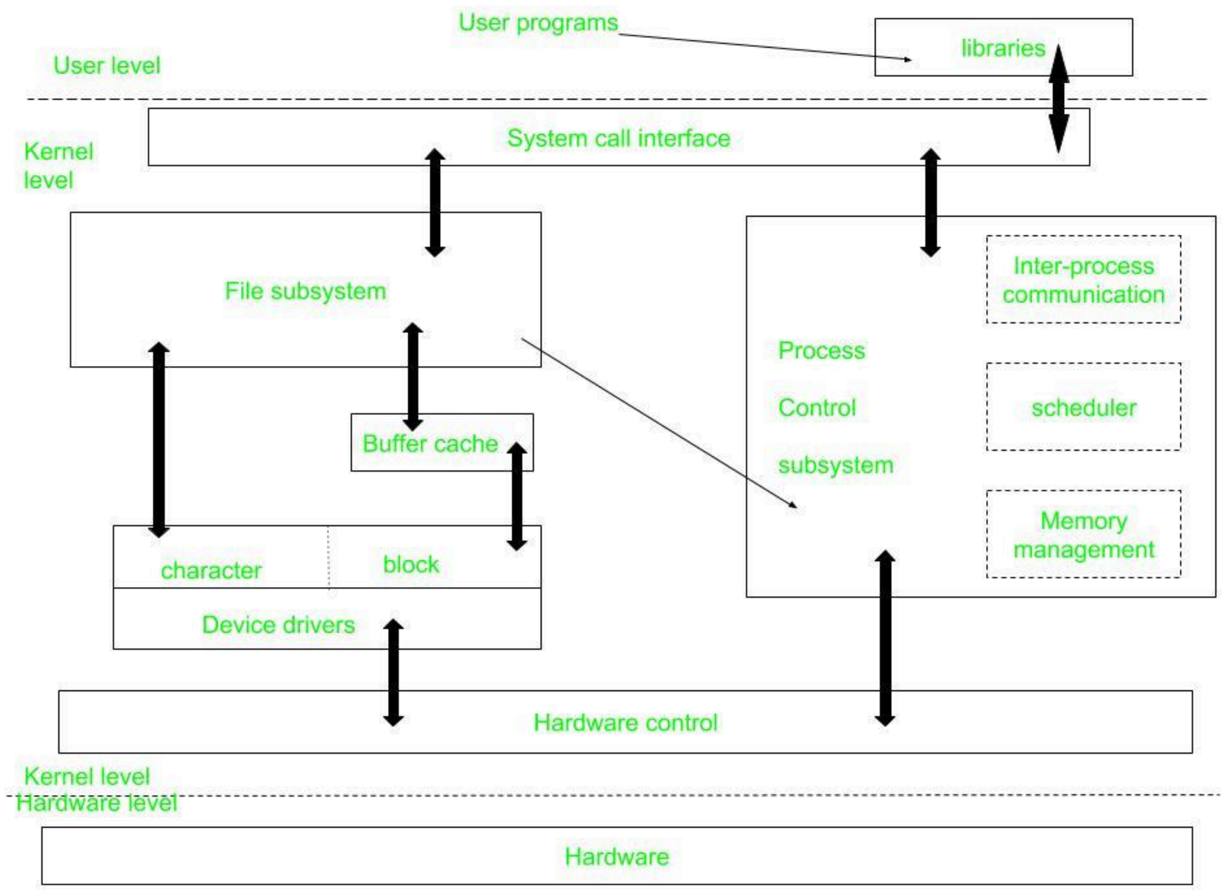


Figure – Kernel and its Block Diagram.

This diagram shows three levels: user, kernel, and hardware.

- The system call and library interface represent the border between user programs and the kernel. System calls look like ordinary function calls in C programs. Assembly language programs may invoke system calls directly without a system call library. The libraries are linked with the programs at compile time.
- The set of system calls into those that interact with the file subsystem and some system calls interact with the process control subsystem. The file subsystem manages files, allocating file space, administering free space, controlling access to files, and retrieving data for users.

- Processes interact with the file subsystem via a specific set of system calls, such as open (to open a file for reading or writing), close, read, write, stat (query the attributes of a file), chown (change the record of who owns the file), and chmod (change the access permissions of a file).
- The file subsystem accesses file data using a buffering mechanism that regulates data flow between the kernel and secondary storage devices. The buffering mechanism interacts with block I/O device drivers to initiate data transfer to and from the kernel.
- Device drivers are the kernel modules that control the operator of peripheral devices. The file subsystem also interacts directly with “raw” I/O device drivers without the intervention of the buffering mechanism. Finally, the hardware control is responsible for handling interrupts and for communicating with the machine. Devices such as disks or terminals may interrupt the CPU while a process is executing. If so, the kernel may resume execution of the interrupted process after servicing the interrupt.
- Interrupts are not serviced by special processes but by special functions in the kernel, called in the context of the currently running process.

(18). Explain grep.

The grep command in Unix/Linux is a powerful tool used for searching and manipulating text patterns within files. Its name is derived from the ed (editor) command g/re/p (globally search for a regular expression and print matching lines), which reflects its core functionality. grep is widely used by programmers, system administrators, and users alike for its efficiency and versatility in handling text data. In this article, we will explore the various aspects of the grep command.

Syntax of grep Command in Unix/Linux:

The basic syntax of the **`grep`** command is as follows:

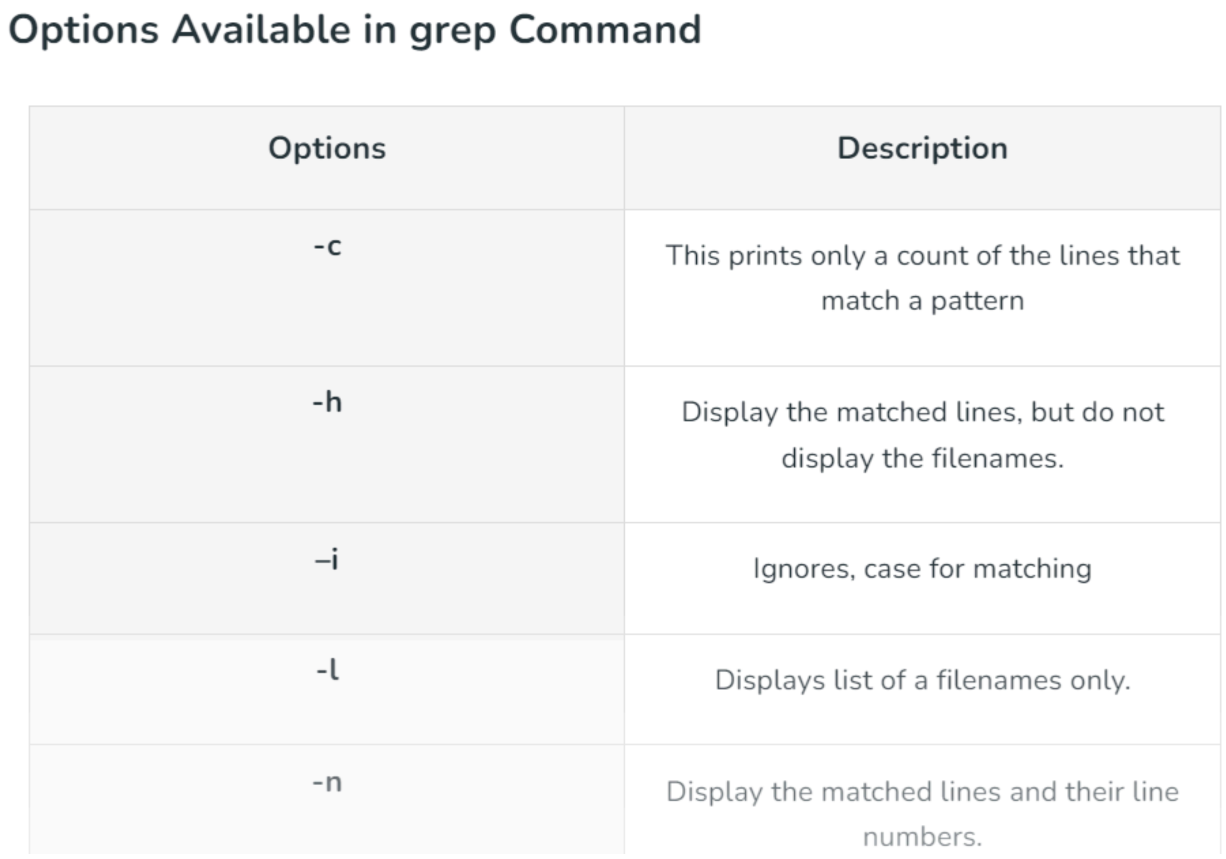
```
grep [options] pattern [files]
```

Here,

[**options**]: These are command-line flags that modify the behavior of grep.

[**pattern**]: This is the regular expression you want to search for.

[**file**]: This is the name of the file(s) you want to search within. You can specify multiple files for simultaneous searching.



Options Available in grep Command

Options	Description
-c	This prints only a count of the lines that match a pattern
-h	Display the matched lines, but do not display the filenames.
–i	Ignores, case for matching
-l	Displays list of a filenames only.
-n	Display the matched lines and their line numbers.

(19). Explain pipe.

A pipe is a form of redirection (transfer of standard output to some other destination) that is used in Linux and other Unix-like operating systems to send the output of one command/program/process to another command/program/process for further processing. The Unix/Linux systems allow the stdout of a command to be connected to the stdin of another command. You can make it do so by using the pipe character **'|'**.

The pipe is used to combine two or more commands, and in this, the output of one command acts as input to another command, and this command's output may act as input to the next command, and so on. It can also be visualized as a temporary connection between two or more commands/ programs/ processes. The command line programs that do the further processing are referred to as filters.

This direct connection between commands/ programs/ processes allows them to operate simultaneously and permits data to be transferred between them continuously rather than having to pass it through temporary text files or through the display screen. Pipes are unidirectional i.e., **data flows from left to right through the pipeline.**

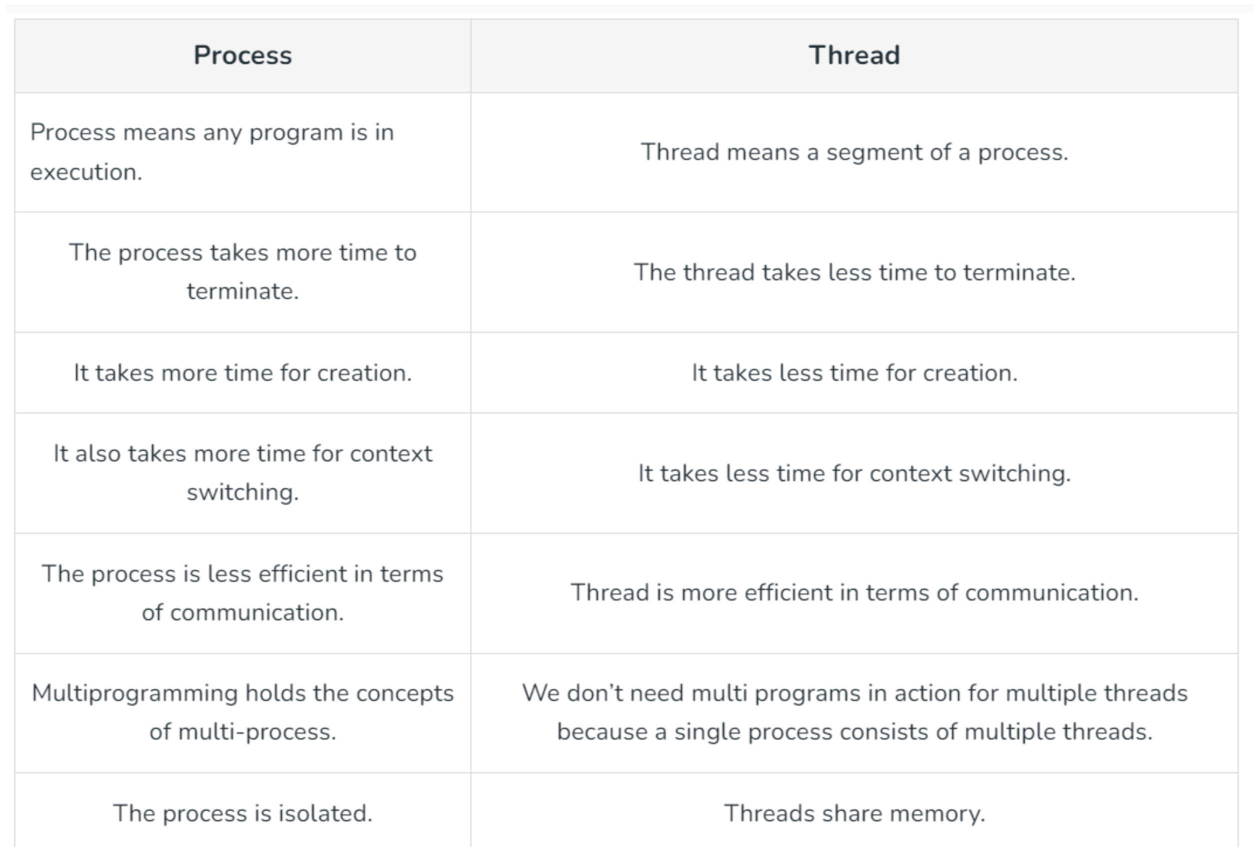
Syntax:

```
command_1 | command_2 | command_3 | .... | command_N
```

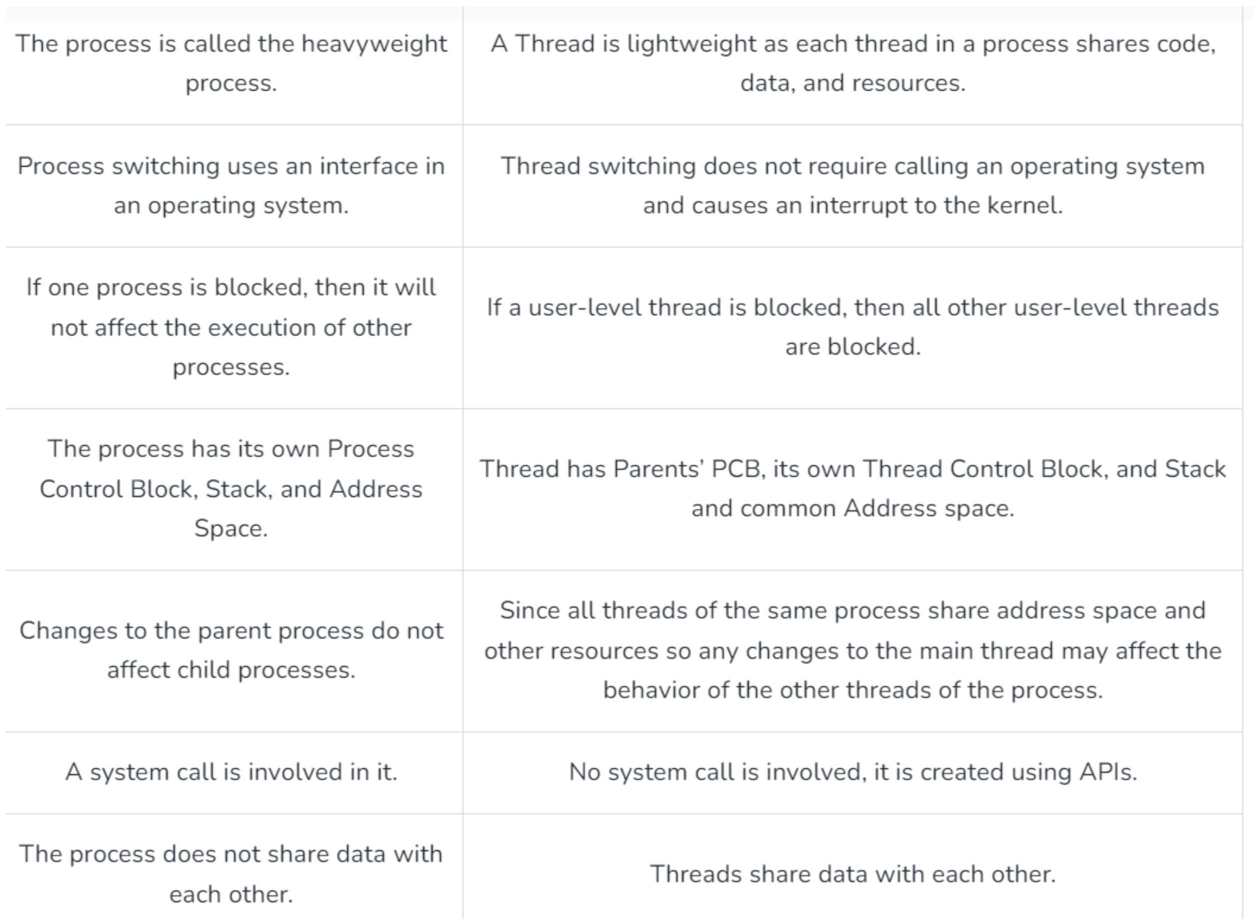
Eg:

```
$ cat sample2.txt | head -7 | tail -5
```

(20). Difference among Thread & Process.



Process	Thread
Process means any program is in execution.	Thread means a segment of a process.
The process takes more time to terminate.	The thread takes less time to terminate.
It takes more time for creation.	It takes less time for creation.
It also takes more time for context switching.	It takes less time for context switching.
The process is less efficient in terms of communication.	Thread is more efficient in terms of communication.
Multiprogramming 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.
The process is isolated.	Threads share memory.



The process is called the heavyweight process.	A Thread is lightweight as each thread in a process shares code, data, and resources.
Process switching uses an interface in an operating system.	Thread switching does not require calling an operating system and causes an interrupt to the kernel.
If one process is blocked, then it will not affect the execution of other processes.	If a user-level thread is blocked, then all other user-level threads are blocked.
The 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.
Changes to the parent process do 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.
A system call is involved in it.	No system call is involved, it is created using APIs.
The process does not share data with each other.	Threads share data with each other.

(21). Explain a scheduling algorithm.

A scheduling algorithm is a set of rules that decides which task to execute at a given time. In computer systems, scheduling algorithms are used to manage and prioritize the execution of tasks, also known as jobs or processes.

Here are some things to know about scheduling algorithms:

- Purpose: Scheduling algorithms are used to maximize the efficiency of a system by ensuring that jobs are completed on time.
- How they work: Scheduling algorithms determine which process gets to run on the CPU, for how long, and in what order.
- Types: There are two types of scheduling algorithms: preemptive and non-preemptive. Preemptive algorithms allow low-priority processes to be preempted

in favor of high-priority processes. Non-preemptive algorithms do not stop a process until it is completed.

- Impact: Scheduling algorithms can affect the performance, efficiency, and fairness of the system, as well as the user experience.
- Real-world example: One example of a scheduling algorithm is earliest deadline first (EDF), which is used in real-time operating systems to place processes in a priority queue.

(22). Explain pre-emptive and non-preemptive scheduling.

In operating systems, scheduling is the method by which processes are given access to system resources, primarily the CPU. Efficient scheduling is essential for optimal system performance and user satisfaction. There are two primary types of CPU scheduling: preemptive and non-preemptive.

Understanding the differences between preemptive and non-preemptive scheduling helps in designing and choosing the right scheduling algorithms for various types of operating systems. You will discover the distinction between preemptive and non-preemptive scheduling in this article. But first, you need to understand preemptive and non-preemptive scheduling before going over the differences.

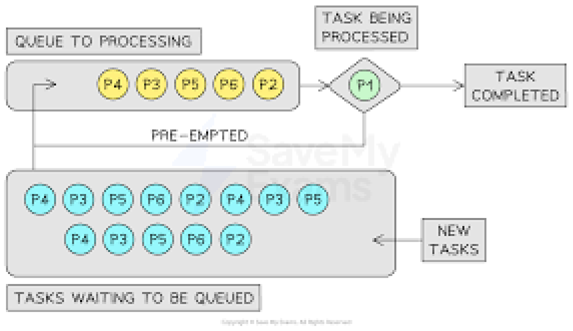
What is Preemptive Scheduling?

Preemptive scheduling is used when a process switches from the running state to the ready state or from the waiting state to the ready state. The resources (mainly CPU cycles) are allocated to the process for a limited amount of time and then taken away, and the process is again placed back in the ready queue if that process still has CPU burst time remaining. That process stays in the ready queue till it gets its next chance to execute.

What is Non-Preemptive Scheduling?

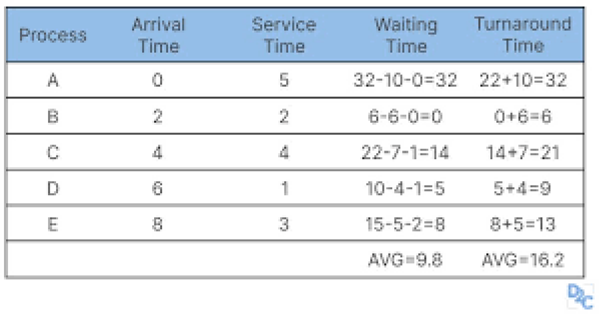
Non-preemptive Scheduling is used when a process terminates, or a process switches from running to the waiting state. In this scheduling, once the resources (CPU cycles) are allocated to a process, the process holds the CPU till it gets terminated or reaches a waiting state. In the case of non-preemptive scheduling it does not interrupt a process running CPU in the middle of the execution. Instead, it waits till the process completes its CPU burst time, and then it can allocate the CPU to another process.

(23). Define the different scheduling algorithms.



Round Robin:

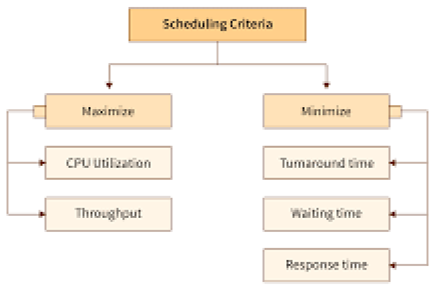
A simple and widely used algorithm that allocates a time slice to each process in the queue.



Process	Arrival Time	Service Time	Waiting Time	Turnaround Time
A	0	5	32-10-0=32	22+10=32
B	2	2	6-6-0=0	0+6=6
C	4	4	22-7-1=14	14+7=21
D	6	1	10-4-1=5	5+4=9
E	8	3	15-5-2=8	8+5=13
			AVG=9.8	AVG=16.2

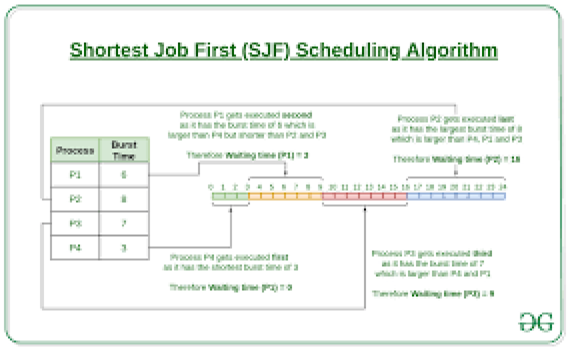
Priority scheduling:

Assigns different priorities to tasks based on how urgent they are. Higher priority tasks are processed before lower priority tasks.



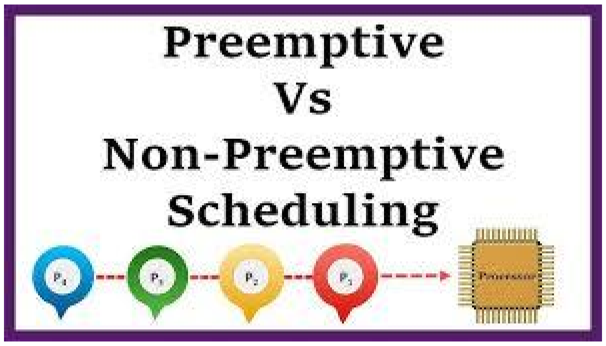
Multilevel queue scheduling:

Similar to multilevel feedback queues scheduling, but processes can change their queue after partial execution.


Shortest Job First (SJF) Scheduling Algorithm
Process
Burst Time
P1
6
P2
8
P3
7
P4
3
Process P1 gets executed second as it has the burst time of 6 which is larger than P4 but shorter than P2 and P3
Therefore Waiting time (P1) = 3
Process P2 gets executed last as it has the largest burst time of 8 which is larger than P4, P1 and P3
Therefore Waiting time (P2) = 16
Process P4 gets executed first as it has the shortest burst time of 3
Therefore Waiting time (P4) = 0
Process P3 gets executed third as it has the burst time of 7 which is larger than P4 and P1
Therefore Waiting time (P3) = 9

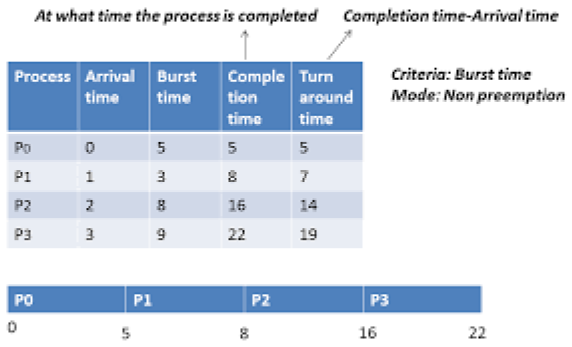
Shortest Job First scheduling:

The waiting process with the shortest execution time is always allocated to the CPU first.



Preemptive scheduling:

Each process is given a fixed time to execute, called a quantum. Once a process has executed for the given time period, it is preempted and another process is executed.


At what time the process is completed   *Completion time-Arrival time*

Criteria: Burst time
Mode: Non preemption

Process	Arrival time	Burst time	Completion time	Turn around time
P0	0	5	5	5
P1	1	3	8	7
P2	2	8	16	14
P3	3	9	22	19

P0	P1	P2	P3

0   5   8   16   22

First come, first serve:

Also known as FCFS, this algorithm runs processes in the order they are added to the queue.

(24). Explain booting process.

The booting process is the process of starting a computer and loading the operating system. It includes the following steps:

- Power on: The computer is turned on by pressing the power button.
- Power On Self Test (POST): The bootloader in the cache memory performs a POST to check that everything is working.
- BIOS: If the POST is successful, the Basic Input Output System (BIOS) is activated.
- Operating system load: The BIOS finds and loads the operating system.
- System configuration: The system configures all devices and initializes other software.
- System utility loads: System utilities are loaded.
- User authentication: The user is authenticated.
- File systems mounted: The file systems are mounted and ready to use.

There are two types of booting: cold booting and warm booting. Cold booting is when a computer is first turned on, while warm booting is when a computer is reset.

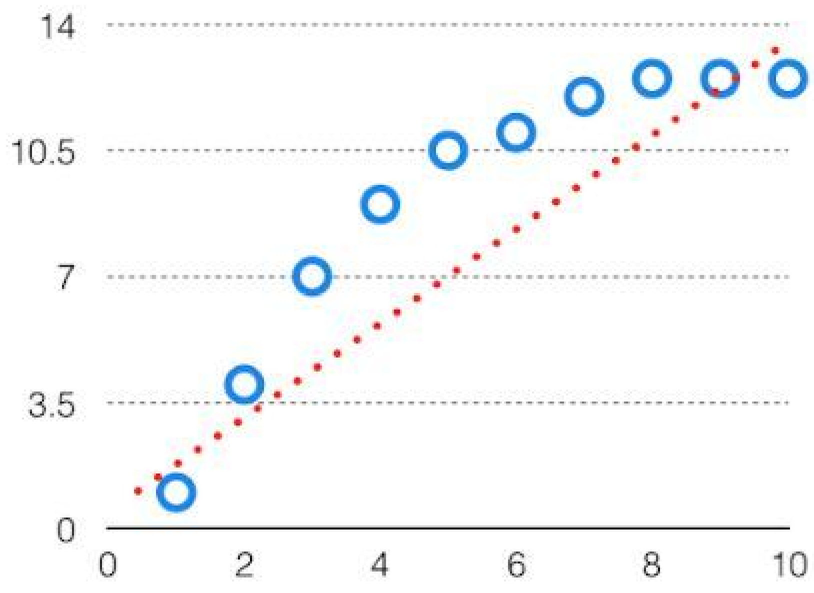
If there is a problem during the booting process, the computer can be started in safe mode. Safe mode allows the user to perform basic functions to diagnose and fix the problem.

(25). Explain bias.

It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine-learning algorithm. There is a tradeoff between a model's ability to minimize bias and variance which is referred to as the best solution for selecting a value of **Regularization** constant. A proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

What is Bias?

The bias is known as the difference between the prediction of the values by the Machine Learning model and the correct value. Being high in biasing gives a large error in training as well as testing data. It recommended that an algorithm should always be low-biased to avoid the problem of underfitting. By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as the **Underfitting of Data**. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation.



In such a problem, a hypothesis looks like follows:

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

(26). Explain the difference among static memory allocation and dynamic memory allocation.

Static memory allocation and dynamic memory allocation differ in how memory is allocated and how it can be changed while a program is running:

Static memory allocation:
Memory is assigned to variables before a program is executed, and remains constant throughout the program's execution. This method is best for variables with known lifetimes and fixed sizes, and is often used in embedded systems and small-scale applications.

Dynamic memory allocation:
Memory is allocated for variables and data structures while a program is running, when the program requests it. This allows for flexibility and efficiency, as the size and location of memory blocks can be changed based on the program's logic and data size.

(27). UNIX commands like touch, sed, grep.

Touch command:

Run the **touch** command to create a new empty file in a specific directory. The syntax is as follows:

```
touch [options] [path_and_file_name]
```

If you omit the path, the **touch** command will create a new file in your current working directory. Here’s an example:

```
touch file.txt
```

Sed command:

Use the **sed** command to search and replace patterns in files quickly. The basic syntax looks like this:

```
sed [options] 'subcommand/new_pattern/target_pattern' input_file
```

You can replace a string in multiple files simultaneously by listing them. Here’s an example of a sed command that changes **red** in **colors.txt** and **hue.txt** with **blue**:

```
sed 's/red/blue' colors.txt hue.txt
```

Grep command:

Global regular expression print or **grep** lets you search specific lines from a file using keywords. It is useful for filtering large data like logs. The syntax looks as follows:

```
grep [options] keyword [file]
```

You can also filter data from another utility by piping it to the **grep** command. For example, the following searches **file.txt** from the **ls** command's output:

```
ls | grep "file.txt"
```

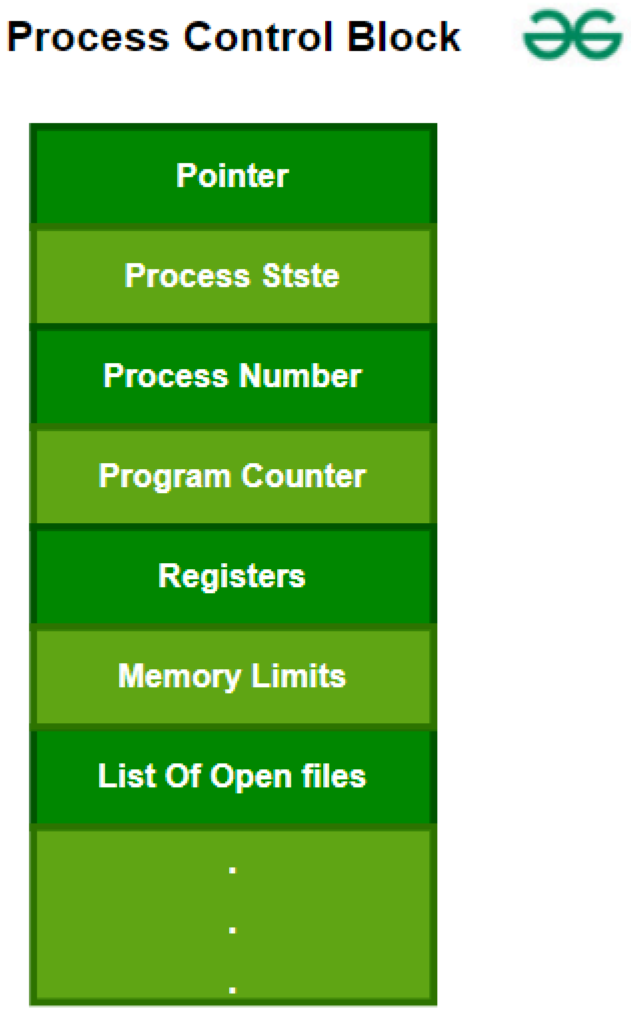
(28). Explain a process and process table? Define different states of process.

While creating a process, the operating system performs several operations. To identify the processes, it assigns a process identification number (PID) to each process. As the operating system supports multi-programming, it needs to keep track of all the processes. For this task, the process control block (PCB) is used to track the process's execution status. Each block of memory contains information about the process state, program counter, stack pointer, status of opened files, scheduling algorithms, etc.

All this information is required and must be saved when the process is switched from one state to another. When the process makes a transition from one state to another, the operating system must update information in the process's PCB. A process control block (PCB) contains information about the process, i.e. registers, quantum, priority, etc. The process table is an array of PCBs, which logically contains a PCB for all of the current processes in the system.

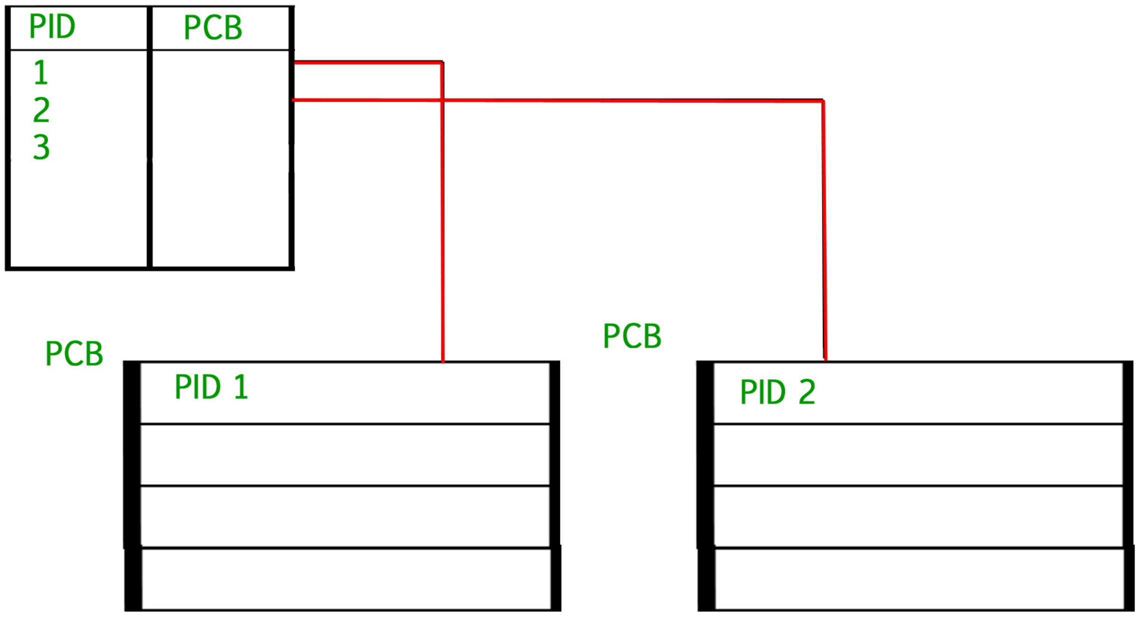
Structure of the Process Control Block:

A Process Control Block (PCB) is a data structure used by the operating system to manage information about a process. The process control keeps track of many important pieces of information needed to manage processes efficiently. The diagram helps explain some of these key data items.



- **Pointer:** It is a stack pointer that is required to be saved when the process is switched from one state to another to retain the current position of the process.
- **Process state:** It stores the respective state of the process.
- **Process number:** Every process is assigned a unique id known as process ID or PID which stores the process identifier.
- **Program counter:** Program Counter stores the counter, which contains the address of the next instruction that is to be executed for the process.

- **Register:** Registers in the PCB, it is a data structure. When a process is running and it's time slice expires, the current value of process specific registers would be stored in the PCB and the process would be swapped out. When the process is scheduled to be run, the register value is read from the PCB and written to the CPU registers. This is the main purpose of the registers in the PCB.
- **Memory limits:** This field contains the information about memory management system used by the operating system. This may include page tables, segment tables, etc.
- **List of Open files:** This information includes the list of files opened for a process.



Process table and process control block

Additional Points to Consider for Process Control Block (PCB):

- **Interrupt Handling:** The PCB also contains information about the interrupts that a process may have generated and how they were handled by the operating system.

- **Context Switching:** The process of switching from one process to another is called context switching. The PCB plays a crucial role in context switching by saving the state of the current process and restoring the state of the next process.
- **Real-Time Systems:** Real-time operating systems may require additional information in the PCB, such as deadlines and priorities, to ensure that time-critical processes are executed in a timely manner.
- **Virtual Memory Management:** The PCB may contain information about a process's virtual memory management, such as page tables and page fault handling.
- **Inter-Process Communication:** The PCB can be used to facilitate inter-process communication by storing information about shared resources and communication channels between processes.
- **Fault Tolerance:** Some operating systems may use multiple copies of the PCB to provide fault tolerance in case of hardware failures or software errors.

Location of The Process Control Block
The Process Control Block (PCB) is stored in a special part of memory that normal users can't access. This is because it holds important information about the process. Some operating systems place the PCB at the start of the kernel stack for the process, as this is a safe and secure spot

(29). Define the benefits of multithreaded programming.

Multithreading has many benefits for operating systems, including:

- Improved performance: Multithreading can take advantage of multiple processors in a computer, allowing for parallel task execution and improved system performance.
- Better responsiveness: Multithreading allows programs to continue running even if part of the program is blocked or performing a lengthy operation.

- Resource sharing: Multithreading allows multiple tasks to share resources like CPU time and memory.
- Simplified program structure: Multithreading can simplify the structure of complex applications by making it easier to design and code.
- Reduced context switching time: Multithreading minimizes the time required for context switching.
- Increased throughput: Multithreading allows for many concurrent compute operations and I/O requests within a single process.

However, multithreading can also introduce complexity and potential issues related to synchronization and concurrency.

(30). Explain Thrashing. What is Thrash?

In computer science, *thrash* is the poor performance of a virtual memory (or paging) system when the same pages are being loaded repeatedly due to a lack of main memory to keep them in memory. Depending on the configuration and algorithm, the actual throughput of a system can degrade by multiple orders of magnitude.

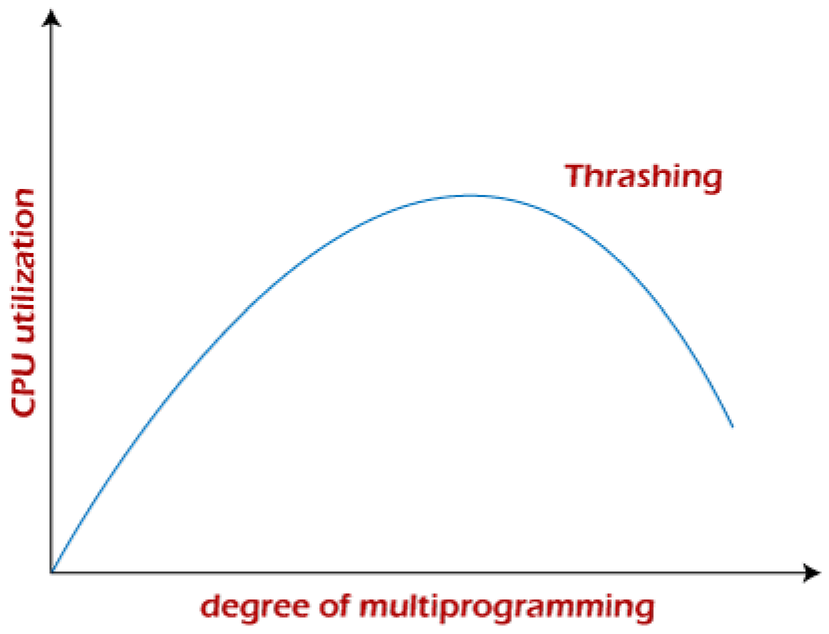
In computer science, *thrashing* occurs when a computer's virtual memory resources are overused, leading to a constant state of paging and page faults, inhibiting most application-level processing. It causes the performance of the computer to degrade or collapse. The situation can continue indefinitely until the user closes some running applications or the active processes free up additional virtual memory resources.

To know more clearly about thrashing, first, we need to know about page fault and swapping.

- ○ Page fault: We know every program is divided into some pages. A page fault occurs when a program attempts to access data or code in its address space but is not currently located in the system RAM.
- ○ Swapping: Whenever a page fault happens, the operating system will try to fetch that page from secondary memory and try to swap it with one of the pages in RAM. This process is called swapping.

Thrashing is when the page fault and swapping happens very frequently at a higher rate, and then the operating system has to spend more time swapping these pages.

This state in the operating system is known as thrashing. Because of thrashing, the CPU utilization is going to be reduced or negligible.



The basic concept involved is that if a process is allocated too few frames, then there will be too many and too frequent page faults. As a result, no valuable work would be done by the CPU, and the CPU utilization would fall drastically.

The long-term scheduler would then try to improve the CPU utilization by loading some more processes into the memory, thereby increasing the degree of multiprogramming. Unfortunately, this would result in a further decrease in the CPU utilization, triggering a chained reaction of higher page faults followed by an increase in the degree of multiprogramming, called thrashing.

Algorithms during Thrashing:

Whenever thrashing starts, the operating system tries to apply either the Global page replacement Algorithm or the Local page replacement algorithm.

1. Global Page Replacement

Since global page replacement can bring any page, it tries to bring more pages whenever thrashing is found. But what actually will happen is that no process gets enough frames, and as a result, the thrashing will increase more and more. Therefore, the global page replacement algorithm is not suitable when thrashing happens.

2. Local Page Replacement

Unlike the global page replacement algorithm, local page replacement will select pages which only belong to that process. So there is a chance to reduce the thrashing. But it is proven that there are many disadvantages if we use local page replacement. Therefore, local page replacement is just an alternative to global page replacement in a thrashing scenario.

Causes of Thrashing:

Programs or workloads may cause thrashing, and it results in severe performance problems, such as:

- If CPU utilization is too low, we increase the degree of multiprogramming by introducing a new system. A global page replacement algorithm is used. The CPU scheduler sees the decreasing CPU utilization and increases the degree of multiprogramming.
- CPU utilization is plotted against the degree of multiprogramming.
- As the degree of multiprogramming increases, CPU utilization also increases.
- If the degree of multiprogramming is increased further, thrashing sets in, and CPU utilization drops sharply.
- So, at this point, to increase CPU utilization and to stop thrashing, we must decrease the degree of multiprogramming.

How to Eliminate Thrashing ?

Thrashing has some negative impacts on hard drive health and system performance. Therefore, it is necessary to take some actions to avoid it. To resolve the problem of thrashing, here are the following methods, such as:

- Adjust the swap file size:If the system swap file is not configured correctly, disk thrashing can also happen to you.
- Increase the amount of RAM: As insufficient memory can cause disk thrashing, one solution is to add more RAM to the laptop. With more memory, your computer can handle tasks easily and don't have to work excessively. Generally, it is the best long-term solution.
- Decrease the number of applications running on the computer: If there are too many applications running in the background, your system resource will consume a lot. And the remaining system resource is slow, which can result in thrashing. So while closing, some applications will release some resources so that you can avoid thrashing to some extent.
- Replace programs: Replace those programs that are heavy memory occupied with equivalents that use less memory.

Techniques to Prevent Thrashing:

The Local Page replacement is better than the Global Page replacement, but local page replacement has many disadvantages, so it is sometimes not helpful. Therefore below are some other techniques that are used to handle thrashing:

1. Locality Model:

A locality is a set of pages that are actively used together. The locality model states that as a process executes, it moves from one locality to another. Thus, a program is generally composed of several different localities which may overlap.

For example, when a function is called, it defines a new locality where memory references are made to the function call instructions, local and global variables, etc. Similarly, when the function is exited, the process leaves this locality.

2. Working-Set Model:

This model is based on the above-stated concept of the Locality Model.

The basic principle states that if we allocate enough frames to a process to accommodate its current locality, it will only fail whenever it moves to some new locality. But if the allocated frames are less than the size of the current locality, the process is bound to thrash.

According to this model, based on parameter A, the working set is defined as the set of pages in the most recent 'A' page references. Hence, all the actively used pages would always end up being a part of the working set.

The accuracy of the working set is dependent on the value of parameter A. If A is too large, then working sets may overlap. On the other hand, for smaller values of A, the locality might not be covered entirely.

If D is the total demand for frames and WSSi is the working set size for process i,

$D = \sum WSSi$

Now, if 'm' is the number of frames available in the memory, there are two possibilities:

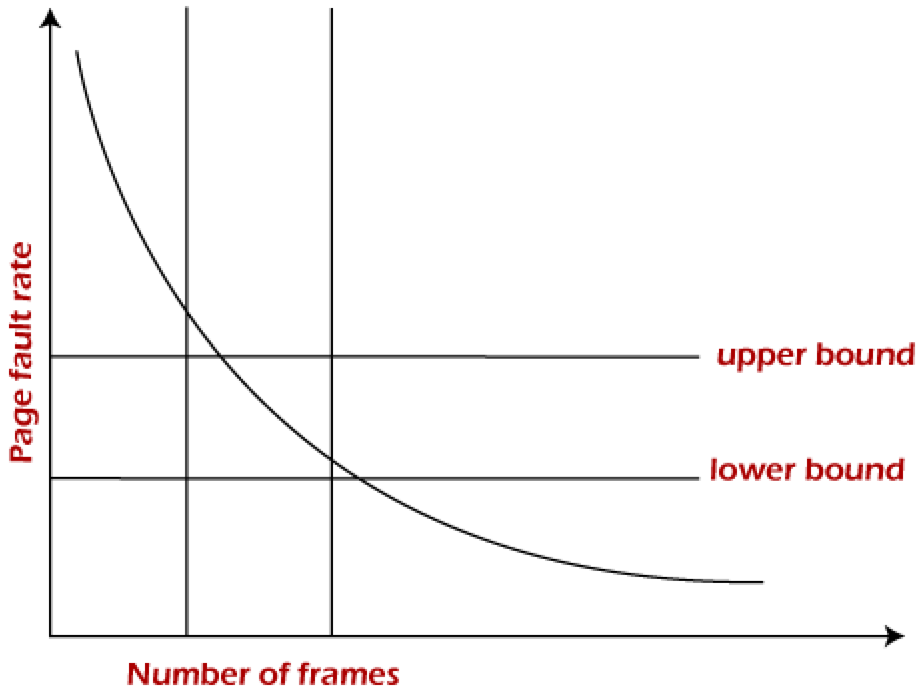
- D>m, i.e., total demand exceeds the number of frames, then thrashing will occur as some processes would not get enough frames.
- D<=m, then there would be no thrashing.

If there are enough extra frames, then some more processes can be loaded into the memory. On the other hand, if the summation of working set sizes exceeds the frames' availability, some of the processes have to be suspended (swapped out of memory).

This technique prevents thrashing along with ensuring the highest degree of multiprogramming possible. Thus, it optimizes CPU utilization.

3. Page Fault Frequency

A more direct approach to handle thrashing is the one that uses the Page-Fault Frequency concept.



The problem associated with thrashing is the high page fault rate, and thus, the concept here is to control the page fault rate.

If the page fault rate is too high, it indicates that the process has too few frames allocated to it. On the contrary, a low page fault rate indicates that the process has too many frames.

Upper and lower limits can be established on the desired page fault rate, as shown in the diagram.

If the page fault rate falls below the lower limit, frames can be removed from the process. Similarly, if the page faults rate exceeds the upper limit, more frames can be allocated to the process.

In other words, the graphical state of the system should be kept limited to the rectangular region formed in the given diagram.

If the page fault rate is high with no free frames, some of the processes can be suspended and allocated to them can be reallocated to other processes. The suspended processes can restart later.

(31). Explain Belady’s Anomaly.

In the Operating System, process data is loaded in fixed-sized chunks and each chunk is referred to as a page. The processor loads these pages in fixed-sized chunks of memory called frames. Typically the size of each page is always equal to the frame size.

A page fault occurs when a page is not found in the memory and needs to be loaded from the disk. If a page fault occurs and all memory frames have been already allocated, then the replacement of a page in memory is required at the request of a new page. This is referred to as demand-paging. The choice of which page to replace is specified by page replacement algorithms. The commonly used page replacement algorithms are FIFO, LRU, optimal page replacement algorithms, etc.

What is a Page Fault?

A page fault is a type of interrupt or exception that occurs in a computer’s operating system when a program attempts to access a page of memory that is not currently loaded into physical RAM (Random Access Memory). Instead, the page is stored on disk in a storage space called the page file or swap space.

Generally, on increasing the number of frames to a process' virtual memory, its execution becomes faster as fewer page faults occur. Sometimes the reverse happens, i.e. more page faults occur when more frames are allocated to a process. This most unexpected result is termed **Belady's Anomaly**.

What is a Belady's Anomaly?

Bélády's anomaly is the name given to the phenomenon where increasing the number of page frames results in an increase in the number of page faults for a given memory access pattern.

This phenomenon is commonly experienced in the following page replacement algorithms:

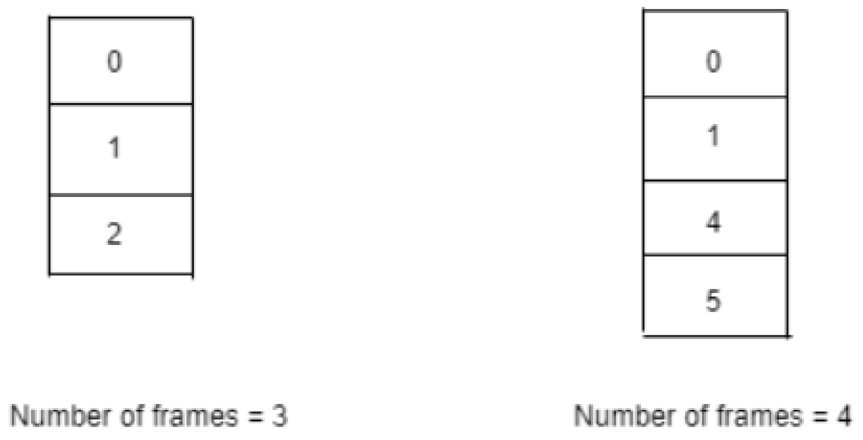
- First in first out (FIFO)
- Second chance algorithm
- Random page replacement algorithm

Reason for Belady's Anomaly:

The other two commonly used page replacement algorithms are Optimal and LRU, but Belady's Anomaly can never occur in these algorithms for any reference string as they belong to a class of stack-based page replacement algorithms.

A **stack-based algorithm** is one for which it can be shown that the set of pages in memory for *N* frames is always a subset of the set of pages that would be in memory with *N + 1* frames. For LRU replacement, the set of pages in memory would be the *n* most recently referenced pages. If the number of frames increases then these *n* pages will still be the most recently referenced and so, will still be in the memory. While in FIFO, if a page named *b* came into physical memory before a page – *a* then priority of replacement of *b* is greater than that of *a*, but this is not independent of the number of page frames and hence, FIFO does not follow a stack page replacement policy and therefore suffers Belady's Anomaly.

Example: Consider the following diagram to understand the behavior of a stack-based page replacement algorithm



As the set of pages in memory with 4 frames is not a subset of memory with 3 frames, the property of stack algorithm fails in FIFO.

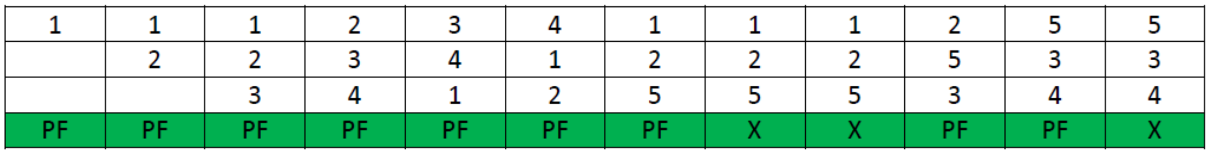
The diagram illustrates that given the set of pages i.e. {0, 1, 2} in 3 frames of memory is not a subset of the pages in memory – {0, 1, 4, 5} with 4 frames and it is a violation in the property of stack-based algorithms. This situation can be frequently seen in the FIFO algorithm.

Belady's Anomaly in FIFO:

Assuming a system that has no pages loaded in the memory and uses the FIFO Page replacement algorithm. Consider the following reference string:

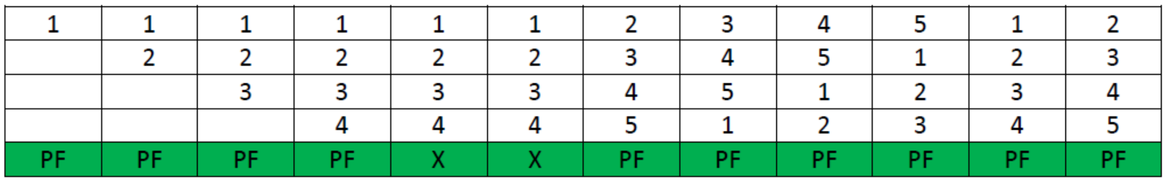
1, 2, 3, 4, 1, 2, 5, 1, 2, 3, 4, 5

Case-1: If the system has 3 frames, the given reference string using the FIFO page replacement algorithm yields a total of 9 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.



1	1	1	2	3	4	1	1	1	2	5	5
	2	2	3	4	1	2	2	2	5	3	3
		3	4	1	2	5	5	5	3	4	4
PF	PF	PF	PF	PF	PF	PF	X	X	PF	PF	X

Case-2: If the system has 4 frames, the given reference string using the FIFO page replacement algorithm yields a total of 10 page faults. The diagram below illustrates the pattern of the page faults occurring in the example.



1	1	1	1	1	1	2	3	4	5	1	2
	2	2	2	2	2	3	4	5	1	2	3
		3	3	3	3	4	5	1	2	3	4
			4	4	4	5	1	2	3	4	5
PF	PF	PF	PF	X	X	PF	PF	PF	PF	PF	PF

It can be seen from the above example that on increasing the number of frames while using the FIFO page replacement algorithm, the number of **page faults increased** from 9 to 10.

Note – It is not necessary that every string reference pattern cause Belady anomaly in FIFO but there are certain kinds of string references that worsen the FIFO performance on increasing the number of frames.

Why Do Stack Based Algorithms Do not Suffer Anomaly?

All the stack based algorithms never suffer Belady Anomaly because these type of algorithms assign a priority to a page (for replacement) that is independent of the number of page frames. Examples of such policies are Optimal, LRU and LFU. Additionally these algorithms also have a good property for simulation, i.e. the miss (or hit) ratio can be computed for any number of page frames with a single pass through the reference string.

In LRU algorithm every time a page is referenced it is moved at the top of the stack, so, the top *n* pages of the stack are the *n* most recently used pages. Even if the number of frames is incremented to *n+1*, the top of the stack will have *n+1* most recently used pages.

(32). Explain starvation and aging.

https://www.geeksforgeeks.org/starvation-and-aging-in-operating-systems/

(33). Explain a trap and trapdoor.

1. Trojan Horse:

- A standalone malicious program that may give full control of an infected PC to another PC is called a Trojan horse.

- This is actually a code segment that tries to misuse its own environment.
- They somehow look attractive but on the other hand, they are really harmful and they actually serve as virus carriers.
- 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, a 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 the system creating a backdoor on your computer that gives unauthorized users access to your system.
- Unlike Trojans do not self-replicate or reproduce by infecting other files nor do they self-replicate which means Trojan horse viruses differ from other computer viruses and do not spread themselves.
- The most popular Trojan horses are Beast, Zeus, The Blackhole Exploit Kit, Flashback Trojan, Netbus, Subseven, Y3K Remote Administration Tool, and Back Orifice.

2.Trap Door:

- A trap door is kind of a secret entry point into a program that allows anyone to gain access to any system without going through the usual security access procedures.
- Another definition of a trap door is it is a method of bypassing normal authentication methods. Therefore it is also known as a back door.
- Trap Doors are quite difficult to detect and also in order to find them the programmers or the developers have to go through the components of the system.
- Programmers use Trap door legally to debug and test programs. Trap doors turn to threats when any dishonest programmers gain illegal access.

- Program development and software update activities should be the first focus of security measures. The operating system that controls the trap doors is difficult to implement.

(34). Explain a daemon.

Daemon processes start working when the system will be bootstrapped and terminate only when the system is shutdown .It does not have a controlling terminal. It always runs in the background.

Characteristics of Daemon Processes:

- Background Operation: Daemons run in the background, performing system-level tasks without direct user intervention.
- Long-lived: Daemons are designed to run for extended periods, handling tasks such as system maintenance, network services, or hardware management.

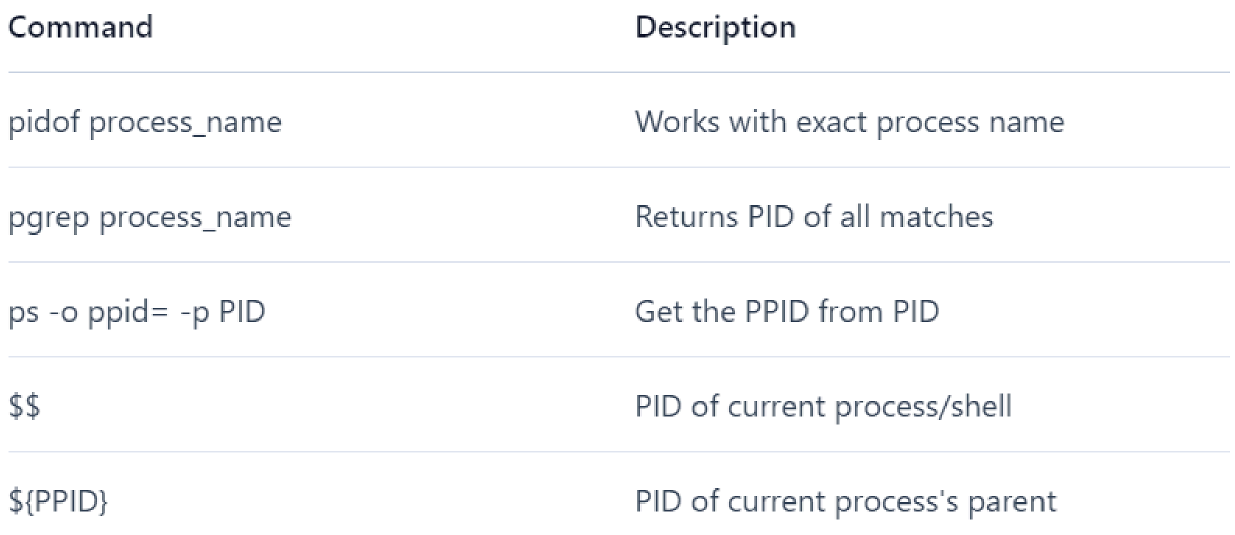
(35). Which application software's executed on the OS ?

Application software, software designed to handle specific tasks for users. Such software directs the computer to execute commands given by the user and may be said to include any program that processes data for a user. Application software thus includes word processors, spreadsheets, database management, inventory and payroll programs, and many other “applications.” Application software is distinguished from system software, which controls a computer’s internal functioning, chiefly through an operating system, and also controls such peripherals as monitors, printers, and storage devices.

(36). Define daemon objects and thread objects.

https://www.geeksforgeeks.org/difference-between-daemon-threads-and-user-threads-in-java/

(37). Give commands for finding process ID.



Command	Description
pidof process_name	Works with exact process name
pgrep process_name	Returns PID of all matches
ps -o ppid= -p PID	Get the PPID from PID
$$	PID of current process/shell
${PPID}	PID of current process's parent

(38). How to edit, rename and move files in Linux ?

https://cyfuture.cloud/kb/linux/how-to-open-edit-move-and-copy-a-file-in-linux

(39). Give 5 commands in Linux with explanation.

https://www.geeksforgeeks.org/linux-commands-cheat-sheet/

(40). Which are deadlock handling situations ?

https://www.geeksforgeeks.org/handling-deadlocks/