Linux Interview Q&A for 3+ years

LINUX OS-based Questions

1. **What is the booting process in detail? Explain**

Read: https://www.tecmint.com/linux-boot-process/

---

2. **Benefits of a multiprocessor system.**

Read: https://www.tutorialspoint.com/Multiprocessor-Systems

These systems have multiple processors working in parallel that share the computer clock, memory, bus, peripheral devices, etc.

More reliable Systems

In a multiprocessor system, even if one processor fails, the system will not halt. This ability to continue working despite hardware failure is known as graceful degradation. For example, if there are 5 processors in a multiprocessor system and one of them fails, then 4 processors are still working. So the system only becomes slower and does not ground to a halt.

Enhanced Throughput

If multiple processors are working in tandem, then the throughput of the system increases, i.e. number of processes getting executed per unit of time increases. If there are N processors, then the throughput increases by an amount just under N.

More Economic Systems

Multiprocessor systems are cheaper than single-processor systems in the long run because they share the data storage, peripheral devices, power supplies, etc. If there are multiple processes that share data, it is better to schedule them on multiprocessor systems with shared data than to have different computer systems with multiple copies of the data.

---

3. What is the RAID structure in the OS? What are the different levels of RAID configuration?

Data redundancy refers to the practice of storing the same piece of data in multiple locations or in multiple instances within a system.

Different RAID Levels

1. RAID-0 (Stripping) (data ko tod kar pieces me rakhta hai, parallelly data ko read, write kar sakte hai)
2. RAID-1 (Mirroring) (disk ki multiple copies rakh rhe for data availability)
3. RAID-2 (Bit-Level Stripping with Dedicated Parity) (obsolete, don't use)
4. RAID-3 (Byte-Level Stripping with Dedicated Parity) (obsolete, don't use)
5. RAID-4 (Block-Level Stripping with Dedicated Parity) (parity is in a separate disk in order to restore the data if one disk fails)
6. RAID-5 (Block-Level Stripping with Distributed Parity) (parity distributed, now parity is in every disk. Now every disk has data and parity)
7. RAID-6 (Block-Level Stripping with two Parity Bits) (create 2 parities each dedicatedly, so even if 2 disks fail, we can recover data from RAID 6)

RAID (Redundant Array of Independent Disks) is a data storage technology that combines multiple physical disk drives into a single logical unit for the purpose of data redundancy (duplicacy), performance improvement, or both. RAID structures are

implemented at the hardware level using RAID controllers or at the software level within the operating system.

In the context of the operating system (OS), RAID structures typically refer to software RAID, where the RAID functionality is managed by the OS itself rather than by a dedicated RAID controller. Software RAID provides flexibility and cost-effectiveness, as it utilizes the CPU and memory resources of the host system to manage disk arrays.

The most common RAID structures implemented in the OS include:

1. **RAID 0 (Striping):** This RAID level distributes data evenly across multiple disks (striping) to improve performance by allowing data to be read from and written to multiple disks simultaneously. However, RAID 0 does not provide redundancy, so the failure of any disk in the array results in data loss.
2. **RAID 1 (Mirroring):** RAID 1 creates an exact copy (mirror) of data on two or more disks. Each disk in the array contains the same data, providing redundancy against disk failures. RAID 1 offers data protection but does not improve performance since data is written to all disks in the array.
3. **RAID 5 (Striping with Parity):** RAID 5 combines striping with distributed parity across multiple disks. Parity information is used to reconstruct data in the event of a disk failure. RAID 5 provides a balance of performance and redundancy, as it offers both data striping for improved read/write performance and parity for fault tolerance.
4. **RAID 6 (Striping with Two Parity):** RAID 6 is similar to RAID 5 but includes an additional parity block (double parity) to provide fault tolerance against the

simultaneous failure of two disks in the array. This level of redundancy comes at the cost of reduced usable storage capacity compared to RAID 5.

5. **RAID 10 OR 1+0 (Striping and Mirroring):** Also known as RAID 1+0, RAID 10 combines striping and mirroring to provide both performance and redundancy benefits. Data is striped across multiple mirrored pairs of disks, offering improved read/write performance and redundancy against disk failures.

---

4. **How to check CPU utilization.**
   top, htop, mpstat, sar, vmstat, uptime

---

5. **Paging concepts.** (Eliminates External fragmentation)

Paging is a memory management technique used to efficiently manage the allocation and use of physical memory (RAM). We divide the process into various chunks, also we divide the main memory into smaller chunks named Frames. The size of both should be the same, i.e., your Page size should be equal to the Frame size.

1. **Virtual Memory:** Virtual memory is a memory management technique that allows processes to use more memory than physically available by utilizing disk space as an extension of RAM. Each process has its own virtual address space, which is divided into pages.
2. **Page Table:** The page table is a data structure used by the operating system to map virtual addresses to physical addresses. It contains entries that associate each virtual page with its corresponding physical page frame in RAM.
3. **Page Fault:** A page fault occurs when a process attempts to access a virtual page that is not currently present in physical memory. The Linux kernel handles page faults by loading the required page from the disk into a free page frame in RAM, updating the page table, and resuming the interrupted process.

4. **Page Replacement:** When physical memory becomes full, the Linux kernel uses page replacement algorithms to select pages to evict from memory and make room for new pages. Common page replacement algorithms include Least Recently Used (LRU), First-In-First-Out (FIFO), and Clock (also known as Second Chance).
5. **Swapping:** Swapping is the process of moving entire processes or parts of processes between physical memory and disk storage. When physical memory is insufficient to hold all active processes, the Linux kernel swaps out inactive pages to disk to free up space for more critical processes.
6. **Memory Management Unit (MMU):** The MMU is a hardware component responsible for translating virtual addresses generated by the CPU into physical addresses in RAM. It performs address translation based on the page table maintained by the operating system.

---

6. What are System Calls?

System calls in Linux are the interface between the user-space applications and the kernel. They provide a way for user-level processes to request services and functionality from the operating system kernel. When a user-level process needs to perform privileged operations or interact with hardware devices, it makes a system call to the kernel, which executes the requested operation on behalf of the process.

---

7. Explain about the fork().

In Linux, fork() is a system call used for process creation. It creates a new process by duplicating the existing process (called the parent process). After the fork() call, two processes are created: the parent process and the child process. These processes are identical in almost every aspect, including the program code, memory, open files, and more. However, they have different process IDs (PIDs) and parent process IDs (PPIDs).

8. **Explain the Process Life Cycle of Process States.**

Process States

The states of a process are as follows:

- **New (Create):** In this step, the process is about to be created, but not yet created. It is the program that is present in secondary memory that will be picked up by the OS to create the process.
- **Ready: New** -> Ready to run. After the creation of a process, the process enters the ready state, i.e., the process is loaded into the main memory. The process here is ready to run and is waiting to get the CPU time for its execution. Processes that are ready for execution by the CPU are maintained in a queue called the ready queue for ready processes.
- **Run:** The process is chosen from the ready queue by the CPU for execution, and the instructions within the process are executed by any one of the available CPU cores.
- **Blocked or Wait:** Whenever the process requests access to I/O or needs input from the user, it enters the blocked or wait state. The process continues to wait in the main memory and does not require the CPU. Once the I/O operation is completed, the process goes to the ready state.
- **Terminated or Completed:** The Process is killed, as well as PCB (process control block) is deleted. The resources allocated to the process will be released or deallocated.
- **Suspend Ready:** A Process that was initially in the ready state but was swapped out of main memory(refer to Virtual Memory topic) and placed onto external storage by the scheduler is said to be in suspend ready state. The process will transition back to a ready state whenever the process is again brought into the main memory.

- **Suspend wait or suspend blocked:** Similar to suspend ready, but uses the process which was performing an I/O operation, and a lack of main memory caused them to move to secondary memory. When work is finished, it may go into suspended readiness.

How Process Moves (Process LifeCycle)

A process can move between different states in an operating system based on its execution status and resource availability. Here are some examples of how a process can move between different states:

- **New to ready:** When a process is created, it is in a new state. It moves to the ready state when the operating system has allocated resources to it and it is ready to be executed.
- **Ready to run:** When the CPU becomes available, the operating system selects a process from the ready queue depending on various scheduling algorithms and moves it to the running state.
- **Running to blocked:** When a process needs to wait for an event to occur (I/O operation or system call), it moves to the blocked state. For example, if a process needs to wait for user input, it moves to the blocked state until the user provides the input.
- **Running to ready:** When a running process is preempted by the operating system, it moves to the ready state. For example, if a higher-priority process becomes ready, the operating system may preempt the running process and move it to the ready state.
- **Blocked to ready:** When the event a blocked process was waiting for occurs, the process moves to the ready state. For example, if a process is waiting for user input and the input is provided, it moves to the ready state.

- **Running to terminate:** When a process completes its execution or is terminated by the operating system, it moves to the terminated state.

---

9. What is the difference b/w multitasking and multiprocessing OS?

1. **Multitasking Operating System:**
    - **Definition:** A multitasking operating system allows multiple tasks or processes to run concurrently on a single CPU by rapidly switching between them.
    - **Concurrency:** Tasks are interleaved or time-shared on the CPU, with each task being allocated CPU time slices. The operating system's scheduler manages the scheduling of tasks, ensuring fair allocation of CPU resources.
    - **Example:** In a multitasking OS, such as modern versions of Windows, macOS, and Linux, multiple applications can run simultaneously on a single CPU. Each application appears to run concurrently, but in reality, the CPU rapidly switches between them.

2. **Multiprocessing Operating System:**
    - **Definition:** A multiprocessing operating system supports the execution of multiple tasks or processes simultaneously across multiple CPUs or processor cores.
    - **Concurrency:** Each CPU or core executes its own set of tasks independently of the others. Processes can run truly concurrently, with each CPU performing its own computations.
    - **Example:** In a multiprocessing OS, such as Linux running on a multi-core processor or a server with multiple CPUs, tasks can be distributed across

multiple processing units. Each CPU or core can execute its own set of instructions, allowing for true parallelism and increased throughput.

Key Differences:

- **Resource Utilization:** In a multitasking OS, multiple tasks share the same CPU, while in a multiprocessing OS, tasks can run on separate CPUs or cores simultaneously, leading to potentially higher overall resource utilization and throughput.
- **Concurrency Level:** Multitasking OS achieves concurrency by time-sharing the CPU among multiple tasks, while multiprocessing OS achieves concurrency by running tasks simultaneously on multiple CPUs or cores.
- **Scalability:** A Multiprocessing OS typically offers better scalability, as additional processing units can be added to increase system capacity and performance. A multitasking OS may have limitations on scalability due to resource contention on a single CPU.
- **Performance:** In general, a multiprocessing OS can provide better performance for computationally intensive tasks that can be parallelized across multiple processors or cores. A multitasking OS may suffer from overhead associated with context switching and resource contention on a single CPU.

---

10. What is a scheduling algorithm? Name different types of scheduling algorithms.

A scheduling algorithm is a set of rules or policies used by the operating system to determine the order in which tasks or processes are executed on a CPU. The goal of a scheduling algorithm is to optimize system performance, resource utilization, and responsiveness by efficiently allocating CPU time to different processes.

1. **First-Come, First-Served (FCFS):**

- **Description:** Processes are executed in the order they arrive. The CPU is allocated to the first process in the ready queue, and subsequent processes are executed in the order of their arrival.
- **Characteristics:** Simple and easy to implement, but may lead to long waiting times for processes with shorter CPU bursts.

2. **Shortest Job Next (SJN) or Shortest Job First (SJF):**
    - **Description:** Select the process with the shortest CPU burst time for execution. If multiple processes have the same shortest burst time, FCFS is used to break ties.
    - **Characteristics:** Minimizes average waiting time and turnaround time, but requires knowledge of process CPU burst times.
3. **Priority Scheduling:**
    - **Description:** Each process is assigned a priority value, and the CPU is allocated to the process with the highest priority. Processes with equal priority are scheduled using a predefined policy (e.g., FCFS).
    - **Characteristics:** Allows for the implementation of different scheduling policies (e.g., preemptive or non-preemptive), but may lead to starvation if lower-priority processes never get CPU time.
4. **Round Robin (RR):**
    - **Description:** Allocates CPU time to processes in fixed-size time slices (time quantum). Each process is allowed to execute for one time quantum before being preempted and placed at the end of the ready queue.
    - **Characteristics:** Provides fair allocation of CPU time among processes and ensures responsiveness, but may suffer from high context-switching overhead.
        - Adapts to changes in process behavior over time, allowing for efficient scheduling of both CPU-bound and I/O-bound processes.

---

11. How do PING and TRACERT commands work? Also, DIG, CURL, MPSTAT, etc.

The PING and TRACERT (or traceroute in Linux) commands are network diagnostic tools used to troubleshoot and analyze network connectivity issues. They work by sending special packets called ICMP (Internet Control Message Protocol) packets to a target host and analyzing the responses received.

- **PING (Packet Internet Groper)** is used to test the reachability of a host on an Internet Protocol (IP) network and measure the round-trip time (RTT) for packets sent to the host.
- **TRACERT (Windows) or traceroute (Linux, macOS)** is used to trace the route taken by packets from the source host to the destination host and identify any network hops (routers) along the path.
- **DIG (Domain Information Groper)** is a command-line tool used for querying Domain Name System (DNS) servers to retrieve DNS-related information, such as IP addresses, domain records, and DNS server configurations.
    - DIG sends a DNS query to the configured DNS server. The DNS server responds with the requested information, such as the IP address associated with the domain or the DNS records
- **CURL (Client URL)** is a command-line tool used for transferring data from or to a server using various protocols, such as HTTP, HTTPS, FTP, and more. It supports a wide range of options and features for making HTTP requests and handling responses.
    - When you execute a CURL command with a URL (e.g., curl https://example.com), CURL establishes a connection to the specified server using the specified protocol (e.g., HTTP or HTTPS) and sends an HTTP request. It then retrieves the response from the server, which may include HTML content, files, or other data.
- **MPSTAT (Multi-Processor Statistics)** is a command-line tool used for monitoring CPU usage and performance on multi-processor systems. It provides detailed statistics about CPU utilization, interrupts, context switches, and more.
    - MPSTAT gathers information about CPU usage and performance by querying the kernel and system resources. It collects data

about each CPU core or processor and displays statistics in a tabular format.

---

12. Difference b/w Paging and Segmentation.

Paging and segmentation are both memory management techniques used in operating systems to organize and manage memory resources efficiently, but they differ in their approach to memory organization, address translation, memory protection, and fragmentation handling. Paging provides a linear and contiguous address space, while segmentation provides a hierarchical and non-contiguous address space.

---

13. What is a Process? What are the different states of a process?

[Check question 8]

---

14. Importance of Inodes.

They play a crucial role in file system management and are essential for organizing and accessing files efficiently.

1. **File Metadata Storage:** Each inode stores metadata information about a file, including its permissions, owner, group, size, timestamps (creation, modification, access), and pointers to data blocks containing the actual file contents. This metadata is crucial for file system operations, such as file access, modification, and permission management.
2. **Efficient File Access:** Inodes enable efficient access to files by providing a direct mapping between file names and file data. When a file is accessed by its name, the file system uses the inode associated with that file to locate its data blocks, reducing the need for time-consuming directory searches.

3. **File System Consistency:** Inodes help maintain the integrity and consistency of the file system by tracking the allocation of disk blocks to files. Each inode contains pointers to the data blocks allocated to the file, allowing the file system to track and manage disk space usage effectively.
4. **File System Performance:** Inodes contribute to file system performance by reducing fragmentation and improving disk access times. The use of inodes allows for efficient allocation and management of disk blocks, minimizing disk seek times and improving overall file system throughput.
5. **File System Security:** Inodes play a crucial role in enforcing file system security by storing file permissions, ownership information, and access control lists (ACLs). This information is used by the operating system to enforce file access permissions and protect sensitive data from unauthorized access.
6. **File System Recovery:** Inodes facilitate file system recovery and maintenance operations by providing a centralized data structure for tracking file metadata and disk block allocation. Inodes can be used to recover lost files, repair file system inconsistencies, and perform disk maintenance tasks.

---

15. How to check the Disk Free space & Disk Usage.

- df -Th (disk free) [shows free & usage of all mounted filesystems]
- du -sh * (disk usage) [shows size of current dir & its content]

---

16. What is a thread in an OS?

A thread is the smallest unit of execution within a process. Threads are independent sequences of instructions that can be scheduled and executed by the CPU. Multiple threads can exist within a single process, and they share the same memory space and resources of that process.

Threads are commonly used in applications to perform concurrent tasks and improve performance. They are particularly useful for tasks that involve I/O

operations, such as network communication or disk access, as well as tasks that can be parallelized, such as image processing or computational tasks.

In summary, a thread is a lightweight unit of execution within a process, capable of concurrent execution, sharing resources with other threads within the same process, and communicating with them through shared memory and synchronization mechanisms. Threads provide a flexible and efficient mechanism for multitasking and parallelism in modern computing environments.

---

17. What is Cache? What are the different types? Explain the entire process of searching in memory using hit and miss.

A cache is a high-speed data storage layer that stores frequently accessed or recently used data to improve access times and reduce latency. It serves as a buffer between the main memory (RAM) and the CPU, holding copies of frequently accessed data to speed up data retrieval operations.

There are several types of caches, including:

1. **CPU Cache:**
    - L1 Cache: A small, fast cache located directly on the CPU chip. It typically consists of separate instruction and data caches.
    - **L2 Cache:** A larger cache that sits between the L1 cache and main memory. It serves as a secondary cache and has higher latency than L1 cache, but is still faster than main memory.
    - **L3 Cache:** A cache that is shared among multiple CPU cores within a processor or across multiple processor cores. It provides additional cache capacity and helps reduce contention for cache resources.
2. **Disk Cache:**
    - Disk caches are used to temporarily store data read from or written to a disk. They help improve disk I/O performance by reducing the number of physical disk accesses.
3. **Web Cache:**

- Web caches (e.g., proxy servers and content delivery networks) store copies of web pages and multimedia content to serve requests from clients more quickly and reduce bandwidth usage.

The process of searching in memory using cache involves the concepts of cache hit and cache miss:

1. **Cache Hit:**
    - When the CPU requests data from memory, the cache controller first checks if the requested data is already present in the cache.
    - If the data is found in the cache (i.e., cache hit), it is retrieved directly from the cache, avoiding the longer latency of accessing the main memory.
    - A cache hit results in faster data access and improves overall system performance.
2. **Cache Miss:**
    - If the requested data is not found in the cache (i.e., cache miss), the cache controller retrieves the data from the main memory and stores a copy of it in the cache for future access.
    - Cache misses incur higher latency compared to cache hits because the data must be fetched from the main memory, which has longer access times.
    - After fetching the data from the main memory, the cache controller updates the cache with the newly retrieved data, replacing older cache entries if necessary.

The efficiency of a cache is measured by its hit rate, which is the percentage of memory accesses that result in cache hits. A higher hit rate indicates that the cache is effectively storing frequently accessed data and reducing the need to access main memory, leading to improved performance.

18. **Explain the concept of Virtual memory. If it's present in hardware, how does it store data?**

Virtual Memory is a storage allocation scheme in which secondary memory can be addressed as though it were part of the main memory.

---

19. **Explain framing, segmentation, and paging.**

In Linux, framing, segmentation, and paging are concepts related to memory management, how the OS manages access to RAM for different programs. Here's a clear breakdown of each.

Framing

- ***Framing* refers to the division of physical memory into fixed-size blocks called frames.** It is part of the paging system, allowing memory to be allocated in manageable, equal-sized chunks.

- **How it works:**
    - The size of a frame is typically 4 KB (but can vary).
    - Physical memory (RAM) is divided into these equal-sized frames.
    - Each frame can hold one page of a process.

- Framing happens in physical memory.

***Analogy*:** Think of RAM as a bookshelf where each shelf (frame) holds a fixed-size book (page).

Segmentation

- *Segmentation* divides a program's memory into logical units called segments like code, data, stack, and heap. Helps in logical memory division and protection.

- **How it works:**

    - Each segment has a base address and a limit (size).

    - Unlike paging, segments can be variable in size.

    - The OS keeps a segment table for each process.

- **Modern Linux note:** Linux uses a flat memory model for user space, meaning segmentation is largely minimal and mostly legacy. However, the concept still exists in hardware (like x86's segment registers).

***Analogy*:** If a program is a book, segmentation divides it into chapters (code, data, etc.), each with a different size.

Paging

- *Paging* divides virtual memory into fixed-size blocks called pages that map to physical frames in RAM. Provides virtual memory, isolation, and efficient memory usage.

- How it works:

- Each process gets its own virtual address space.
- The virtual memory is broken into pages (typically 4 KB).
- A page table maps virtual pages to physical frames.
- Pages that are not in RAM can be stored in swap space on disk.

- Key features:
  - Supports demand paging (load pages only when needed).
  - Enables memory protection and isolation between processes.

***Analogy*:** Think of paging like a post office forwarding system the address you see (virtual) is redirected behind the scenes to the actual location (physical frame).

---

20. What is a bootstrap program in OS?

A bootstrap program, also known as a boot loader or bootstrap loader, is a small program that initializes the operating system (OS) and prepares the computer system for normal operation when it is powered on or restarted.

The primary function of the bootstrap program is to load the kernel into memory and transfer control to it so that the OS can take over and manage the system's hardware and resources.

Power-On Self-Test (POST):

- The bootstrap program often begins by performing a series of diagnostic tests called the Power-On Self-Test (POST). These tests check the hardware components of the computer, such as the CPU, memory

(RAM), storage devices, and peripherals, to ensure they are functioning properly.

Bootstrapping the Operating System:

- After completing the POST, the bootstrap program locates and loads the kernel into main memory. This involves reading the necessary files (such as the kernel image) from the storage device (e.g., hard disk, solid-state drive) and copying them into RAM.

Initializing the Operating System:

- Once the operating system kernel is loaded into memory, the bootstrap program transfers control to the kernel. At this point, the operating system takes over and begins its initialization process, initializing device drivers and establishing system services.

Handing Control to the Operating System:

- Finally, the bootstrap program hands control of the system to the operating system kernel, allowing it to manage the system's hardware resources and provide services to applications and users.

The bootstrap program is a critical component of the boot process in a computer system, as it ensures that the operating system is properly loaded and initialized, allowing the system to become operational.

---

21. Explain demand paging.

A memory management technique where pages are loaded into physical memory only when accessed by a process, instead of loading the entire program into RAM at launch. It improves performance and memory efficiency by reducing initial memory load.

How It Works (Step-by-Step)

1. **Program starts**

- Only minimal parts (like headers or entry points) are loaded.
- Code, data, heap, etc., are mapped but not yet loaded into physical memory.

2. **Page access by the process**
    - Process tries to access a page not currently in RAM.
    - Triggers a page fault.
3. **Page fault handling**
    - OS checks if the access is valid.
    - If valid:
        - A free frame is allocated.
        - Page is loaded from disk or zeroed.
        - Page table is updated.
4. **Process resumes**
    - Execution continues as if nothing happened.

Sources of Demand-Paged Data

- **Executable files** – Program code, static data.
- **Shared libraries** – Dynamically linked .so files.
- **Swap space** – When memory is full, pages may be swapped out.

- **Anonymous memory** – Pages created using `malloc`, `brk`, or `mmap` with no file backing.

---

22. What do you mean by RTOS?

Real-time operating systems (RTOS) are used in environments where a large number of events, mostly external to the computer system, must be accepted and processed in a short time or within certain deadlines. Such applications are industrial control, telephone switching equipment, flight control, and real-time simulations. With an RTOS, the processing time is measured in tenths of seconds. This system is time-bound and has a fixed deadline. The processing in this type of system must occur within the specified constraints. Otherwise, this will lead to system failure.
Examples of real-time operating systems are airline traffic control systems, Command Control Systems, airline reservation systems, Heart pacemakers, Network Multimedia Systems, robots, etc.

---

23. What is virtual memory?

Virtual memory is a memory management technique used by Linux (and all modern operating systems) to give processes the illusion of having more memory than what is physically available (RAM), while also providing isolation, security, and efficiency. Virtual memory is an abstraction layer that provides each process with its own private address space, mapping virtual addresses to physical memory or disk storage.

Key Concepts

1. Virtual Address Space

- Every process thinks it has access to a full range of memory (e.g., 0x00000000 to 0xFFFFFFFF on 32-bit).

- This space is isolated — one process cannot access another's memory.

2. Page Tables

- The kernel uses page tables to map virtual addresses → physical addresses.

- Pages are typically 4 KB in size.

- Mapping is handled by hardware (Memory Management Unit or MMU) and the OS.

3. Swap Space

- When RAM is full, Linux can move inactive pages to swap space on disk.

- This extends available memory but is much slower than RAM.

4. Demand Paging

- Pages are only loaded into RAM on demand (when accessed).

- Unused parts of memory stay on disk or unallocated until needed.

---

24. Explain the Zombie process.

When a process completes its execution, it becomes a zombie process until its parent process performs the necessary cleanup steps.

A child process enters after it has terminated but before its parent process has acknowledged its termination and collected its exit status.

In the zombie state, the process still exists in the process table, but it no longer executes any code. It retains its process ID (PID) and exit status, allowing the parent process to retrieve this information later.

Once the parent process collects the exit status of the child process, the zombie process is removed from the process table, and its resources are released.

---

25. What is thrashing in the OS?

Thrashing in operating systems refers to a situation where the system is excessively busy swapping data between main memory (RAM) and disk storage, resulting in a significant decrease in system performance. Thrashing occurs when the system's memory resources are overwhelmed by the demands of running processes, leading to frequent and continuous paging or swapping activity.

---

26. What are starvation and aging in the OS?

Starvation and aging are concepts related to process scheduling in operating systems.

Starvation:

- Starvation occurs when a low-priority process is prevented from making progress or accessing resources indefinitely because higher-priority processes continuously consume the CPU.
- In other words, a process is said to be starving if it is unable to execute despite being ready to run, often due to higher-priority processes consuming all available CPU time.
- Starvation can lead to unfairness and inefficiency in the system, as low-priority processes may experience significant delays or may never get a chance to execute, even though they are technically ready to run.

- To mitigate starvation, scheduling algorithms may incorporate mechanisms to ensure that all processes, regardless of priority, have the opportunity to execute over time. This may involve periodically boosting the priorities of low-priority processes or implementing aging mechanisms to gradually increase the priority of waiting processes.

2. **Aging:**
    - Aging is a technique used in priority-based scheduling algorithms to prevent starvation by gradually increasing the priority of waiting processes over time.
    - When a process waits for an extended period in the ready queue without being selected for execution, its priority is increased incrementally.
    - By gradually boosting the priorities of waiting processes, aging ensures that lower-priority processes eventually receive an opportunity to execute, even if higher-priority processes are continuously being scheduled.
    - Aging helps maintain fairness and prevent starvation by ensuring that all processes have a chance to run, regardless of their initial priority.

In summary, starvation occurs when low-priority processes are indefinitely delayed or prevented from executing due to higher-priority processes monopolizing system resources. Aging is a technique used to mitigate starvation by gradually increasing the priority of waiting processes over time, ensuring fairness and preventing low-priority processes from being starved.

---

27. What is the Kernel? Write its main functions.

The kernel is the core component of an operating system that provides essential services and manages system resources, allowing applications to run and interact with the hardware.

---

28. I'm trying to transfer media over FTP from one device to another. The rate of transfer is very slow. Troubleshoot the scenario.

Troubleshooting Slow FTP Transfer

1. Check Network Performance

- Use ping to check latency.
- Use iperf to test bandwidth between devices.

2. Try Switching FTP Modes

- Toggle between Passive and Active mode.
- Firewalls/NATs may affect Passive mode.

3. Monitor CPU & Disk Usage

- Use top and iostat to check resource bottlenecks on both devices.

4. Test Different FTP Clients

- Try lftp, FileZilla, or ncftp for better performance.
- lftp supports parallel and resumed transfers.

5. Use Binary Transfer Mode

- Run ftp> binary to avoid ASCII mode issues.

6. Check FTP Server Rate Limits

- Review /etc/vsftpd.conf:

```
 ini
CopyEdit
local_max_rate=0
anon_max_rate=0
```

- Restart FTP server after changes.

7. Compare with SFTP or SCP

- Test transfer using scp or sftp to isolate the issue.

8. Check Server Logs & Load

- Check /var/log/vsftpd.log
- Monitor system load with uptime, vmstat

---

29. Memory Management: Memory pages, Buffer, and Caches.

1. Memory Pages

- Memory is divided into fixed-size blocks called pages (typically 4 KB).
- Each process gets its own virtual memory, which is divided into pages.
- Linux uses paging to map virtual pages to physical frames via the page table.
- Benefits:
    - Isolates process memory
      Enables demand paging
    - Supports swap

2. Buffers

- Buffers are memory areas used to temporarily store data during I/O operations.
- Example: When writing to a disk, data is first stored in a buffer.
- Purpose:
    - Improves write performance
    - Minimizes direct disk I/O

Can be monitored using:

bash
CopyEdit
free -m

cat /proc/meminfo

3. Caches

- Caches store recently or frequently accessed data to speed up reads.
- Linux keeps portions of files and data in page cache, not just for one app, but for the entire system.
- Types:
    - Page cache: File data cached in RAM
    - Dentry/inode cache: File system metadata
- Caches are automatically reclaimed by the kernel if memory is needed.

Clear cache manually (not recommended in production):

bash
CopyEdit
echo 3 > /proc/sys/vm/drop_caches

Check Memory Usage

bash
CopyEdit
free -h

Output fields:

- used: Actual used memory
- buffers/cache: Memory used for buffers and caches
- available: How much memory is really free to use

---

30. Basic troubleshooting commands.

These are essential commands every Linux administrator or engineer should know for quick diagnosis and issue resolution.

1. System Status & Load

Command	Purpose
`uptime`	Shows system load and uptime
`top` / `htop`	Real-time CPU, memory, and process usage
`vmstat 1`	Memory, process, I/O, and system stats
`free -h`	Shows available and used memory
`dmesg`	Displays kernel logs

2. Network Troubleshooting

Command	Purpose
`ping <host>`	Test network connectivity
`traceroute <host>`	Show path packets take to destination
`netstat -tunlp`	Show open ports and listening services
`ss -tuln`	Modern alternative to netstat
`curl -I <url>`	Check HTTP response headers
`dig` / `nslookup`	DNS resolution testing
`ip a` / `ifconfig`	Show IP addresses of interfaces
`ethtool eth0`	View/change NIC settings
`nmap <host>`	Scan for open ports on remote machine

3. Disk & File System

Command	Purpose
`df -h`	Check disk space usage
`du -sh *`	Check space used by files/folders
`lsblk`	List block devices (disks/partitions)
`mount` / `umount`	View or manage mounted file systems
`fdisk -l`	View partition details
`iostat`	I/O stats (needs sysstat package)

4. Process & Service Management

Command	Purpose
`ps aux`	List running processes
`kill <PID>`	Terminate a process
`systemctl status <service>`	View service status
`systemctl restart <service>`	Restart a service
`journalctl -xe`	Check system logs (systemd)

5. File & Permission Issues

Command	Purpose
`ls -l`	Check file permissions
`chmod` / `chown`	Change permissions or ownership
`getfacl` / `setfacl`	View/set ACL permissions
`chattr` / `lsattr`	View/set immutable attributes

6. Package & System Info

Command	Purpose
`uname -a`	System info (kernel, architecture)
`hostnamectl`	View/set hostname and OS details
`rpm -qa` / `dpkg -l`	List installed packages
`which <command>`	Show command location
`lsof -i :<port>`	Check which process uses a port

7. Log Monitoring

Command	Purpose
`tail -f /var/log/messages`	Real-time system log monitoring
`less /var/log/syslog`	

or

`/var/log/secure`	View logs securely
`grep "pattern" file`	Search within logs
