1. What is OS? What are its important functions?

OS (Operating System) is a software program that manages computer hardware and software resources and provides common services for computer programs. Its important functions include:

Process management: allocating system resources, scheduling processes, and providing inter-process communication. Memory management: managing the system's main memory, including virtual memory and memory allocation to running processes. File management: organizing, storing, and retrieving files and directories on storage devices. Device management: managing I/O devices such as printers, scanners, and keyboards. Security: protecting the system from unauthorized access and ensuring data integrity.

2. What is system call? How it is executed?

System call is a programming interface between a user process and the operating system. It is executed by the CPU when a process makes a request to the operating system to perform a privileged operation, such as reading or writing to a file, creating a new process, or allocating memory. The system call interface is provided by the kernel, and its execution involves switching from user mode to kernel mode.

3. Explain terms: Multi-programming, Multi-tasking, Multi-threading, Multi-processing and Mutli-user?

Multi-programming: running multiple programs on a single processor by time-sharing. Multi-tasking: running multiple tasks within a single program or across multiple programs, allowing them to run concurrently. Multi-threading: running multiple threads within a single program, allowing them to run concurrently and share resources. Multi-processing: running multiple processes across multiple processors or cores, allowing them to run concurrently. Multi-user: supporting multiple users to run their own programs and access system resources concurrently.

4. Explain Linux kernel design (monolithic or modular).

Linux kernel design is modular, meaning that it is built from separate components or modules that can be loaded or unloaded at runtime. The kernel includes a core set of functions and device drivers, and additional functionality can be added through loadable kernel modules (LKMs). This design allows the kernel to be customized and optimized for specific use cases.

5. Explain process life cycle.

Process life cycle consists of several stages:

New: a new process is created by the operating system. Ready: the process is waiting to be assigned to a processor. Running: the process is being executed on a processor. Waiting: the process is waiting for a certain event, such as I/O completion. Terminated: the process has finished execution and is terminated.

6. What are Linux IPC mechanisms? Explain any three with diagram.

Linux IPC (Inter-Process Communication) mechanisms include: Pipes: allow communication between two processes by creating a unidirectional data channel. Message queues: allow multiple processes to send and receive messages through a queue. Shared memory: allows multiple processes to access the same region of memory.

7. What is difference between process and thread? How can you create them in Linux program?

A process is a program in execution, while a thread is a lightweight process that can run concurrently with other threads in the same process. In Linux, processes are created with the fork() system call, while threads can be created with the pthread_create() function.

8. What is difference between semaphore and mutex? How can you create them in Linux program?

Semaphore and mutex are both synchronization primitives used to control access to shared resources in multi-threaded or multi-process applications. The main difference between them is that a semaphore can be used to allow multiple threads/processes to access the shared resource simultaneously, while a mutex allows only one thread/process at a time. In Linux, semaphores can be created with the sem_init() function, while mutexes can be created with the pthread_mutex_init() function.

9. What is file? Which system calls are used to manipulate files in Linux? Write their prototypes and explain.

A file is a collection of related data stored on a storage device. System calls used to manipulate files in Linux include:

open(): opens a file and returns a file descriptor. read(): reads data from a file into a buffer. write(): writes data from a buffer to a file. close(): closes a file and releases the file descriptor.

10. What is paging? What is page fault and how it is handled?

Paging is a memory management technique used by the operating system to allow processes to access more memory than is physically available. In paging, the main memory is divided into fixed-size blocks called pages, and each page can be allocated to a process. When a process accesses a page that is not currently in the main memory, a page fault occurs. Page fault is an exception raised by the hardware when a process tries to access a page that is not present in the main memory. When a page fault occurs, the operating system performs the following steps:

The CPU traps to the kernel, which saves the state of the process. The operating system determines the location of the required page on disk and reads it into the main memory. The operating system updates the page table to reflect the new location of the page in the main memory. The operating system resumes the process from where it was interrupted.

11. What is difference between starvation and deadlock?

Starvation and deadlock are both problems that can occur in multi-process or multi-threaded systems. Starvation occurs when a process or thread is prevented from making progress due to a lack of resources, such as CPU time or memory. Deadlock occurs when two or more processes or threads are waiting for each other to release resources, resulting in a situation where no progress can be made.

12. How vfork() system call works? What is copy-on-write?

The fork() system call creates a new process by duplicating the calling process. The new process, called the child process, is an exact copy of the parent process, except for the process ID and a few other details. The child process runs in its own address space and has its own copy of the parent's data and code.

Copy-on-write is a technique used by some operating systems, including Linux, to optimize the fork() system call. Instead of immediately duplicating all of the parent's data and code, the child process shares the same address space with the parent. When either process tries to modify a shared page, the operating system creates a copy of the page and assigns it to the modifying process, while the other process continues to use the original page.

13. Explain OS booting. Explain Linux booting.

OS booting is the process of starting up the operating system on a computer. The boot process typically involves the following steps: Power-on self-test (POST): the computer's hardware is checked to ensure it is functioning properly. BIOS/UEFI initialization: the Basic Input/Output System (BIOS) or Unified Extensible Firmware Interface (UEFI) is initialized to prepare the system for booting. Boot loader: a boot loader program is loaded from disk into memory and executed. Kernel initialization: the operating system kernel is loaded into memory and initialized. Init process: the init process is started, which initializes system services and runs startup scripts. Linux booting follows a similar process, with some differences in the boot loader and initialization stages. The boot loader on Linux systems is typically GRUB (Grand Unified Bootloader), which allows the user to select which operating system to boot. The kernel initialization involves loading and initializing device drivers, while the init process on Linux systems can be either systemd or SysV init.

14. How many Linux run-levels are there? Which features are enabled in each runlevel?

Linux has 7 run-levels, numbered from 0 to 6. Each run-level has a specific purpose and enables or disables certain system services. The features enabled in each run level are: Run-level 0: shut down the system. Run-level 1: single-user mode, with minimal system services. Run-level 2: multi-user mode, with no network services. Run-level 3: multi-user mode, with full networking. Run-level 4: reserved for custom use. Run-level 5: multi-user mode with graphical interface. Run-level 6: reboot the system.

15. Explain regex commands and wildcard characters.

Regular expressions, or regex, is a sequence of characters that define a search pattern. In Linux, regex commands are often used in conjunction with other commands to search for and manipulate data. Some common regex commands include: grep: searches for a pattern in a file sed: performs text transformations on a file awk: processes and analyzes text files Wildcard characters, such as * and ?, can also be used in conjunction with regex commands to match patterns in filenames or directories.

16. Which Linux command is used to kill all running instances of the same program.

The Linux command used to kill all running instances of the same program is pkill. For example, to kill all running instances of the firefox browser, you would use the command pkill firefox.

17. How security is implemented in Linux file systems? Tell commands related to it.

Linux file systems implement security through file permissions, which specify the access rights of users and groups to files and directories. The chmod command is used to change file permissions, while the chown command is used to change the ownership of a file or directory. Additionally, Linux file systems also support access control lists (ACLs), which provide more fine-grained access control to files and directories.

18. How redirection and pipe in Linux commands work?

In Linux, redirection and pipes are used to redirect the input and output of commands. Redirection is used to send the output of a command to a file or to read input from a file. The > symbol is used to redirect output to a file, while the < symbol is used to redirect input from a file. For example, the command ls > output.txt would redirect the output of the ls command to a file called output.txt.

Pipes, on the other hand, are used to send the output of one command as input to another command. The | symbol is used to pipe output from one command to another. For example, the command ls | grep "example" would list all files in the current directory and pipe the output to the grep command, which would search for files containing the string "example".