Submission by: Ashish Kumar

DevOps Master's

ASSIGNMENT 2

Ques. 1. What are inode and process id?

Answer: Inode (Index Node) is a data structure used in Unix-based file systems to store information about a file or directory such as its permissions, ownership, timestamps, and location of the data blocks that make up the file's contents.

Process ID (PID) is a unique identifier assigned to every process in an operating system. It is used to track and control individual processes, as well as to differentiate between processes with the same name. The PID is used by many system calls and system utilities to manipulate or obtain information about the process.

Ques. 2. Which are the Linux Directory Commands?

Answer: Here is a list of commonly used Linux directory commands:

ls - list the contents of a directory

pwd - print the current working directory

cd- change the current working directory

mkdir- create a new directory

rmdir- remove an empty directory

cp- copy files and directories

mv- move or rename files and directories

rm- remove files and directories

find- search for files and directories

du- estimate file space usage.

These are just a few of the many directory commands available in Linux.

Ques. 3. What is Virtual Desktop?

Answer: A virtual desktop is a software-based simulation of a physical desktop environment that runs on a computer. It allows users to create multiple, separate desktop environments within a single operating system instance, and switch between them as needed. Each virtual desktop can have its own set of open windows, applications, and settings, and operates independently of the others.

Virtual desktops are useful for organizing tasks and reducing clutter, and are especially helpful for power users who need to keep multiple applications and windows open at the same time. Some popular examples of virtual desktop software include Gnome Boxes, VirtualBox, and VMware Workstation.

Ques. 4. Which are the different modes of vi editor?

Answer: The vi editor has three modes of operation:

Command mode: This is the default mode when you open a file in vi. In this mode, you can perform various file-level operations, such as saving changes, quitting, or moving the cursor.

Insert mode: In this mode, you can enter text into the file. To switch to insert mode, press the **i** key while in command mode.

Visual mode: In this mode, you can select and manipulate text in the file. There are two sub-modes in visual mode: character-based selection (press **v** to enter) and line-based selection (press **V** to enter).

It's important to understand the difference between these modes, as the available commands and actions you can perform in vi depend on which mode you are in.

Ques. 5. What are daemons?

Answer: A daemon is a background process in a Unix-like operating system that runs independently of the terminal and performs a specific task or set of tasks. Daemons typically start when the system boots up and continue to run in the background until the system is shut down.

Examples of daemons include the Apache web server daemon, the SSH daemon for remote logins, and the printing daemon for managing print jobs. Daemons are essential components of a Unix-like system, as they provide many of the underlying services that make the system usable.

Daemons can be managed using various tools and techniques, such as process management utilities like **init, system, or upstart**, and can be configured to start automatically or be launched on demand.

Ques. 6. What are the process states in Linux?

Answer: In Linux, every process goes through different states during its lifetime. The following are the main process states in Linux:

- Running: The process is currently executing and using CPU time.
- Sleeping: The process is waiting for an event to occur, such as the completion of an I/O operation or a signal from another process.
- Stopped: The process has been stopped, typically by a user or by receiving a signal. A stopped process can be restarted.
- Zombie: The process has completed execution but its status has not yet been reported to its parent process. A zombie process consumes no system resources and will eventually be removed from the system.

- Dead: The process has terminated and its resources have been freed by the operating system.

These states are tracked by the Linux kernel, and the state of a process can be determined using various tools, such as the **ps** and **top** commands. Understanding the different process states is important for managing and monitoring the performance of a Linux system.

Ques. 7. Explain grep command.

Answer: The **grep** command is a powerful tool in Linux for searching and filtering text in one or more files. It stands for "global regular expression print," and it can be used to search for specific patterns of text, such as keywords, phrases, or regular expressions.

Here's the basic syntax of the **grep** command:

grep PATTEN FILE

The **PATTERN** argument is a string or a regular expression that you want to search for, and **FILE** is the name of the file you want to search in. For example, the following command will search for the word "error" in the file "log.txt":

grep error log.txt

The **grep** command can also search for patterns in multiple files by specifying multiple file names:

grep PATTERN FILE1 FILE2 FILE3

Additionally, you can use the **-r** or **–recursive** option to search for patterns in all files in a directory and its subdirectories:

grep -r PATTERN DIRECTORY

Ques. 8. Explain Process Management System Calls in Linux

Answer: Process management system calls in Linux are a set of functions provided by the operating system that allow a program to create, manage, and control processes. The following are some of the most commonly used process management system calls in Linux:

fork(): This system call creates a new process by duplicating the calling process. The new process is known as the child process, while the original process is known as the parent process.

exec(): This system call replaces the current process image with a new process image. It's commonly used to execute new programs or scripts within a process.

wait(): This system call blocks the calling process until one of its child processes terminates. The wait() system call is typically used by parent processes to wait for the completion of their child processes.

kill():This system call sends a signal to a process, causing it to terminate or change its behavior. The kill(): system call is often used to stop processes that are not responding or to terminate processes that are running in the background.

exit():This system call terminates the calling process. The process's resources, including memory and open file descriptors, are freed by the operating system.

These system calls are the basic building blocks of process management in Linux, and they form the foundation for more advanced process management tools, such as the **init** process and the **systemd** service manager. Understanding how to use these system calls is important for managing processes and building scalable and reliable software systems in Linux.

Ques. 9. Explain the ‘ls’ command

Answer: The ls command is a basic but essential tool in Linux for listing the contents of a directory. The ls command stands for "list," and it is used to display the names of the files and subdirectories in a specified directory.

Here's the basic syntax of the ls command:

ls OPTIONS DIRECTORY

The **OPTIONS** argument is a list of optional flags that modify the behavior of the ls command. For example, the -l option will display the contents of a directory in a long format, showing details such as file permissions, owner, size, and modification time. The -a option will show hidden files and directories, which are typically prefixed with a dot (.).

The **DIRECTORY** argument is the name of the directory you want to list. If no directory is specified, the ls command will list the contents of the current directory. For example, the following command will list the contents of the **/etc** directory:

ls /etc

The **ls** command is a basic but versatile tool that can be used in many ways, such as for exploring the file system, checking the contents of a directory, or verifying the existence of a file. It's an essential tool for any Linux user to have in their toolkit.

Ques. 10. Explain the redirection operator

Answer: The redirection operator is a shell feature in Linux that allows you to redirect input and output from and to files, or to other commands. Redirection is a powerful feature that enables you to manipulate the flow of data in your shell scripts and command-line tools.

Here are some of the most commonly used redirection operators in Linux:

>: This operator redirects the standard output of a command to a file, overwriting the contents of the file if it already exists. For example, the following command will redirect the output of the ls command to a file named "file_list.txt":

ls > file_list.txt

>>: This operator is similar to >, but it appends the output to the file instead of overwriting it. For example, the following command will append the output of the **ls** command to the file "file_list.txt":

ls >> file_list.txt

<: This operator redirects the standard input of a command from a file. For example, the following command will sort the contents of a file named "file_list.txt" and write the sorted output to the screen:

sort < file_list.txt

|: This operator pipes the output of one command to the input of another command. It allows you to chain multiple commands together and create complex pipelines of data processing. For example, the following command will sort the contents of a file named "file_list.txt" and write the sorted output to a file named "sorted_list.txt":

sort < file_list.txt > sorted_list.txt

These redirection operators are fundamental to shell scripting and command-line tools in Linux, and they provide an easy and flexible way to manipulate and process data. Understanding how to use these operators is important for automating tasks, performing data processing, and building complex systems in Linux.