Assignment-2 Linux

1. **In Linux FHS (Filesystem Hierarchy Standard) what is the /?**

In the Linux FHS (Filesystem Hierarchy Standard), the / (root) directory is the top-level directory in the filesystem hierarchy. It is the base of the entire directory tree and contains all other directories and files on the system. All directories and files on the system are located either directly or indirectly under the root directory.

2. **What is stored in each of the following paths?**
   /bin, /sbin, /usr/bin and /usr/sbin
   /etc
   /home
   /var
   /tmp
   /bin: This directory contains essential command binaries that are required for the system to boot and run. Example:ls, cp, mv, and rm.
   /sbin: This directory contains system binaries that are used for system administration tasks, such as mounting file systems, setting up networking, and managing user accounts. Example:ifconfig, fdisk, and iptables.
   /usr/bin: This directory contains non-essential command binaries that are used by normal users, as well as some system utilities that are not required for booting the system. Example:text editors, image viewers, and media players.
   /usr/sbin: This directory contains system administration binaries that are not required for normal user operations. Example: dhcpd, sshd, and httpd.
   /etc: This directory contains system configuration files, which are used to configure various aspects of the system and its applications. Example:/etc/passwd, which contains user account information, and /etc/fstab, which specifies file systems to be mounted at boot time.
   /home: This directory contains user home directories, which are where user-specific files and settings are stored. Each user on the system has a corresponding directory in /home.
   /var: This directory contains variable files, which are files that change frequently during normal system operation. Example:log files in /var/log, mail spools in /var/spool/mail, and temporary files in /var/tmp.

/tmp: This directory contains temporary files, which are used for storage of data that is only needed temporarily. Files in /tmp may be deleted automatically by the system or by the user, and should not be relied upon for long-term storage.

3. **What is special about the /tmp directory when compared to other directories?**

   The /tmp directory is a special directory in the Linux filesystem hierarchy that is used for temporary file storage. There are a few things that make the /tmp directory unique when compared to other directories:

   1.Data in /tmp may be deleted automatically: Many Linux distributions are configured to periodically clean out the /tmp directory, deleting any files that have not been accessed in a certain amount of time or that are older than a certain age. This means that files stored in /tmp should not be relied upon for long-term storage.

   2./tmp is often mounted as a separate filesystem: On some Linux distributions, the /tmp directory is actually a separate filesystem that is mounted at boot time. This allows the system administrator to set different filesystem parameters for /tmp, such as limiting the amount of disk space that can be used for temporary files.

   3./tmp is typically writable by all users: By default, the /tmp directory is configured to be writable by all users on the system. This allows any user to create temporary files in /tmp, which can be useful for a variety of purposes such as compiling software or storing temporary data for a script. However, this can also be a security risk if users are allowed to create and execute malicious scripts in /tmp.

4. **What kind of information one can find in /proc?**

   The /proc directory in the Linux filesystem hierarchy is a virtual filesystem that provides a view into the kernel's internal data structures and information about running processes. Here are some examples of the types of information that can be found in /proc:

   1.System information: The /proc directory contains a number of files and subdirectories that provide information about the system, such as /proc/cpuinfo, which contains information about the CPU, and /proc/meminfo, which contains information about memory usage.

2.Process information: Each running process on the system has a corresponding directory in /proc with a numerical name, such as /proc/1234. These directories contain information about the process, such as its status, memory usage, file descriptors, and command line arguments.
3.Kernel information: The /proc directory contains files and directories that provide information about the running kernel, such as /proc/cmdline, which contains the kernel command line arguments, and /proc/sys, which contains various system parameters that can be read or modified by the user.
4.Hardware information: The /proc directory contains information about hardware devices on the system, such as /proc/interrupts, which lists interrupt requests and their handlers, and /proc/devices, which lists the available device drivers.

5. **What makes /proc different from other filesystems?**
   The /proc filesystem is different from other filesystems because it is a virtual filesystem that does not exist on a physical disk. Instead, it is created by the Linux kernel on-the-fly as a way to provide access to kernel data structures and system information in a hierarchical file-like structure. This means that the files and directories in the /proc filesystem do not correspond to actual files and directories on a physical disk, but rather they provide an interface to kernel data structures and other system information.
   Another key difference is that files in the /proc filesystem are not static, but rather they are dynamically generated in real-time as the system runs. For example, the /proc/cpuinfo file provides information about the system's CPUs, but the contents of the file will change as new processes are started or stopped.

6. **True or False? only root can create files in /proc**
   False.
   In general, any user can create files in the /proc directory, including non-root users. However, some files and directories in /proc may be restricted to root user only, depending on the system configuration and permissions. The /proc directory is a virtual filesystem that provides an interface to kernel data structures and system information. It is dynamically generated

by the kernel and contains various system information files, such as process information, hardware and system configuration details, network statistics, and more.

7. **What can be found in /proc/cmdline?**
   The /proc/cmdline file contains the command line arguments passed to the kernel at the time of booting the system. These arguments are used to configure the kernel and to specify various options, such as the root device, kernel parameters, boot options, and more.

8. **In which path can you find the system devices (e.g. block storage)?**
   In Linux, the system devices, including block storage devices, are typically represented as device files that can be found in the /dev directory.
   Block storage devices, such as hard drives, solid-state drives, USB drives, and other storage media, are represented as block special files, which are accessed using the "raw" device driver.
   The device files for block storage devices in Linux have names that follow a specific naming convention. The first two letters indicate the device type, and the remaining letters and numbers indicate the specific device. For example, the device file for the first hard drive on a Linux system is typically named "/dev/sda".
   Other types of system devices, such as character devices, network devices, and input/output devices, are also represented as device files and can be found in the /dev directory.
   It's important to note that the devices in the /dev directory are just special files that represent the physical or virtual devices on the system, and they are not the devices themselves. The device files allow applications and users to interact with the devices using standard file operations, such as reading and writing data to the device.

Permissions

9. **How to change the permissions of a file?**
   We can change the permissions of a file in Linux using the chmod command. The chmod command allows you to modify the read, write, and execute permissions of a file for the owner, group, and others.
   The basic syntax of the chmod command is as follows:
   chmod [options] mode file

Here, mode specifies the permissions to be set, and file specifies the file for which the permissions are to be changed.

There are several ways to specify the mode argument, but the most common way is to use a numeric mode, which represents the permissions as a three-digit octal number. The first digit represents the permissions for the owner, the second digit represents the permissions for the group, and the third digit represents the permissions for others.

The permission values are calculated as follows:

4 = read permission
2 = write permission
1   = execute permission

10.What does the following permissions mean?
777
644
750

Sol:777: This file has read, write, and execute permissions for the owner, group, and others. This is the highest level of permissions and provides full access to the file.

644: This file has read and write permissions for the owner, and read-only permissions for the group and others. This is a typical permissions setting for a file that should be readable by everyone but only writable by the owner.

750: This file has read, write, and execute permissions for the owner, read and execute permissions for the group, and no permissions for others. This is a typical permissions setting for a file that should be accessible and editable by the owner and the members of a specific group, but not by others.

11.What this command does? chmod +x some_file

The command chmod +x some_file adds the execute permission to the file "some_file" for the owner, group, and others.

The chmod command is used to change the permissions of a file or directory in Linux, and the +x option adds the execute permission to the file. The x permission allows the file to be executed as a program or script.

So, by running the chmod +x some_file command, you are allowing the owner, group, and others to execute the file "some_file" as a program or script. This command does not modify any other existing permissions, such as read or write permissions, it just adds the execute permission to the file

12.Explain what is setgid and setuid

The setgid permission is a special permission that is applied to directories in Linux. When the setgid bit is set on a directory, it means that any new files or subdirectories created within that directory inherit the group ownership of the parent directory, instead of the group ownership of the user who created them. This is useful for shared directories where multiple users need to access and modify the same files.

For example, if a directory has the setgid bit set and is owned by the "developers" group, any new files or directories created within that directory will also be owned by the "developers" group. This ensures that all members of the "developers" group can access and modify those files and directories.

The setuid permission is a special permission that can be applied to executable files in Linux. When the setuid bit is set on an executable file, it means that the program is executed with the privileges of the owner of the file, instead of the privileges of the user who is executing the program. This can be used to allow users to execute certain programs with elevated privileges, such as a system administrator or root user.

For example, if the setuid bit is set on an executable file owned by the root user, and a non-root user executes the program, the program will be executed with root privileges, giving the user access to certain system resources that would not normally be accessible to them.

13.What is the purpose of sticky bit?

The sticky bit is a special permission that can be applied to directories in Linux and Unix systems. When the sticky bit is set on a directory, it means that only the owner of a file or directory can delete or rename that file or directory, even if other users have write permissions on that directory.

The purpose of the sticky bit is to prevent accidental or malicious deletion of files in shared directories. For example, if a directory is shared among multiple users, and one user accidentally deletes a file owned by another user, it can cause problems and conflicts.

By setting the sticky bit on the directory, the owner of the file or directory can control who has permission to delete or rename it, even if other users have write permissions on that directory. This ensures that important files are not accidentally deleted or modified, and helps to prevent unauthorized access to sensitive information.

14.What the following commands do?
chmod
chown
Chgrp

The following commands are used to manage file and directory permissions, ownership, and group membership in Linux and Unix systems:

chmod: This command is used to change the permissions of a file or directory. It allows you to add or remove read, write, or execute permissions for the owner, group, or other users. You can also set permissions using numeric values or symbolic notation. For example, chmod 755 file.txt sets the permissions of file.txt to read, write, and execute for the owner, and read and execute for the group and other users.

chown: This command is used to change the ownership of a file or directory. It allows you to change the owner and group of a file or directory. For example, chown root:users file.txt changes the owner of file.txt to root and the group to users.

chgrp: This command is used to change the group ownership of a file or directory. It allows you to change the group ownership of a file or directory to a specific group. For example, chgrp users file.txt changes the group ownership of file.txt to the group "users".

15.What is sudo? How do you set it up?

sudo is a command in Linux and Unix systems that allows users to execute commands with elevated privileges or as another user, such as the root user. It provides a way for authorized users to perform administrative tasks without logging in as the root user, which can be dangerous and potentially harmful to the system.

Setting up sudo involves a few steps:

Install sudo: sudo is typically installed by default on most Linux and Unix systems, but if it is not installed, you can install it using your system's package manager.

Configure sudo: Once sudo is installed, you need to configure it by editing the /etc/sudoers file. This file defines which users or groups are allowed to use sudo, and what commands they are allowed to execute. You should always use the visudo command to edit the /etc/sudoers file, which provides syntax checking and ensures that multiple users do not edit the file simultaneously.

Add users to the sudo group: By default, most Linux systems create a sudo group during installation. You can add users to this group using the usermod command. For example, to add the user "johndoe" to the sudo group, you would use the command sudo usermod -aG sudo johndoe.

16.True or False? In order to install packages on the system one must be the root user or use the sudo command

True. By default, only the root user has permission to install packages on Linux and Unix systems, as this requires access to critical system files and directories. However, authorized users can also use the sudo command to execute package management commands with elevated privileges, allowing them to install and manage packages without logging in as the root user.

17.Explain what are ACLs. For what use cases would you recommend to use them?

Access Control Lists (ACLs) are a type of permission system used in Linux and Unix systems that allow for fine-grained control over file and directory permissions. ACLs provide an additional layer of security beyond the traditional Unix permissions system, which only allows for the assignment of permissions to the owner, group, and others.

ACLs allow you to set permissions for individual users or groups, providing more control over who can access, modify, or delete files and directories. This can be particularly useful in scenarios where multiple users or groups need access to the same file or directory, but with different levels of permissions.

Some use cases where ACLs may be recommended include:

Shared directories: In scenarios where multiple users or groups need access to a shared directory, ACLs can be used to set individual permissions for each user or group.

Sensitive files: For sensitive files that need to be protected from unauthorized access, ACLs can be used to restrict access to only authorized users or groups.

Collaboration: In collaborative environments where multiple users need to work on the same file, ACLs can be used to control access to specific parts of the file, or to restrict access to certain users or groups.

18.You try to create a file but it fails. Name at least three different reason as to why it could happen

There are several reasons why creating a file could fail in a Linux or Unix system. Here are three possible reasons:

Permission denied: The user trying to create the file may not have the necessary permissions to create files in the target directory, or may not have permission to create files with the desired permissions. This can be due to file ownership, directory ownership, or permission settings on the directory or file.

Disk full: The disk where the user is trying to create the file may be full or nearly full, which can prevent the creation of new files. This can be due to insufficient disk space or quotas on the user's account.

File already exists: The file name that the user is trying to use may already be in use by another file, which can prevent the creation of the new file. This can occur if the user tries to create a file with a name that already exists in the same directory, or if the user is trying to create a file with the same name as a file in a different directory that is included in the system's PATH variable.

19.A user accidentally executed the following chmod -x $(which chmod). How to fix it?

If a user accidentally runs the command chmod -x $(which chmod), it removes the executable permission from the chmod command, making it impossible to modify file permissions using the chmod command.

To fix this, the user will need to restore the executable permission to the chmod command. There are a few ways to do this, depending on the user's level of access:

If the user has root access, they can use the su command to become the root user and then restore the executable permission using the command chmod +x $(which chmod).

If the user has sudo access, they can run the command sudo chmod +x $(which chmod) to restore the executable permission.

If the user does not have root or sudo access, they will need to contact the system administrator to restore the executable permission.

Scenarios

20.You would like to copy a file to a remote Linux host. How would you do?

There are several ways to copy a file to a remote Linux host. Here are a few options:

SCP (Secure Copy): SCP is a secure file transfer protocol that is included with most Linux distributions. To use SCP to copy a file to a remote host, you can use the following command:

```
scp /path/to/local/file username@remote_host:/path/to/destination
```

Replace /path/to/local/file with the path to the file you want to copy, username with the username of the remote host, remote_host with the IP address or hostname of the remote host, and /path/to/destination with the destination path on the remote host.

SFTP (Secure FTP): SFTP is a secure file transfer protocol that allows you to transfer files between systems using the SSH protocol. To use SFTP to copy a file to a remote host, you can use the following command:

```
sftp username@remote_host:/path/to/destination
```

This will open an SFTP session on the remote host. You can then use the put command to upload files from your local system to the remote host.

Rsync: Rsync is a powerful file synchronization tool that is commonly used for transferring files between Linux hosts. To use rsync to copy a file to a remote host, you can use the following command:

```
rsync /path/to/local/file username@remote_host:/path/to/destination
```

This will copy the file to the destination on the remote host, while preserving file permissions and other metadata.

21.How to generate a random string?

There are several ways to generate a random string in Linux. Here are a few options:

Using the openssl command:

```
openssl rand -base64 12
```

This will generate a random string of 12 characters using the base64 encoding.

Using the tr command with the /dev/urandom device:

```
tr -dc 'a-zA-Z0-9' </dev/urandom | head -c 12
```

This will generate a random string of 12 alphanumeric characters.

Using the uuidgen command:

uuidgen | tr -d '-'

This will generate a random UUID and remove the hyphens to create a string of 32 alphanumeric characters.

Using the pwgen command:

pwgen 12 1

This will generate a random password of 12 characters.

22.How to generate a random string of 7 characters?

To generate a random string of 7 characters in Linux using the openssl command:

openssl rand -base64 5 | tr -dc 'a-zA-Z0-9' | head -c 7

This command generates a random string of 5 bytes using the base64 encoding, then removes any non-alphanumeric characters using tr, and finally uses head to keep only the first 7 characters.

Systemd

23.What is systemd?

Systemd is a system and service manager for Linux operating systems. It is designed to provide a centralized management system for services and processes, and to simplify the process of booting and managing a Linux system.

Systemd replaces traditional init systems and provides a wide range of features, including:

Parallel startup of system services
On-demand starting of services
Resource management and limiting
Dependency-based service control
Support for socket and D-Bus activation
Event logging and system state management
User session management
Integration with containerization technologies

24.How to start or stop a service?

The process for starting or stopping a service can vary depending on the Linux distribution and the specific service you are working with. However, here are some general steps that we can follow:

Identify the name of the service you want to start or stop. This can often be done by looking at the name of the service file in the /etc/init.d directory or by using the systemctl command to list all available services.

To start a service, use the following command:

```
sudo systemctl start SERVICE_NAME
```

Replace SERVICE_NAME with the name of the service you want to start.

To stop a service, use the following command:

```
sudo systemctl stop SERVICE_NAME
```

Replace SERVICE_NAME with the name of the service you want to stop.

25.How to check the status of a service?

To check the status of a service in Linux, you can use the systemctl command. Here are the steps to follow:

Open a terminal on your Linux system.

Type the following command:

```
systemctl status SERVICE_NAME
```

Replace SERVICE_NAME with the name of the service you want to check.

Press Enter to run the command.

The output of the command will show you the current status of the service, as well as any recent log messages related to the service.

If the service is running, the output will indicate that it is active (running).

If the service is not running, the output will indicate that it is inactive (stopped).

If there is an error with the service, the output will indicate that it is failed.

26.On a system which uses systemd, how would you display the logs?

On a Linux system that uses systemd, you can use the journalctl command to display system logs. Here are the steps to follow:

Open a terminal on your Linux system.

Type the following command to display all logs:

```
journalctl
```

Press Enter to run the command.

The output of the command will show you all system logs in reverse chronological order (i.e., newest logs first). You can use the arrow keys to scroll through the logs.
To exit the log viewer, press q.

27.Describe how to make a certain process/app a service

In order to make a process or application a service on a Linux system, you can create a systemd service file. Here are the general steps:
Create a new systemd service file in the /etc/systemd/system/ directory. The filename should end in .service. For example, to create a service file for the myapp process, you could create a file called **/etc/systemd/system/myapp.service.**
In the service file, add the following sections:

```
[Unit]
Description=MyApp

[Service]
Type=simple
ExecStart=/path/to/myapp

[Install]
WantedBy=multi-user.target
```

Replace MyApp with a description of the application or process, and replace /path/to/myapp with the actual path to the executable file.
Save the service file and exit the editor.
Run the following command to reload the systemd daemon and update the list of available services:

```
systemctl daemon-reload
```

Start the service using the following command:

```
systemctl start myapp.service
```

Replace myapp.service with the name of the service file you created.
Verify that the service is running using the following command:

```
systemctl status myapp.service
```

If the service is running, the output will indicate that it is active (running).
To make the service start automatically at boot time, run the following command:

systemctl enable myapp.service
Now, the process or application is running as a systemd service, and will start automatically at boot time.

28.Troubleshooting and Debugging

Troubleshooting and debugging are essential skills for Linux system administrators, as it allows them to identify and resolve issues with the system. Here are some techniques for troubleshooting and debugging in Linux:
Check system logs: The system logs contain valuable information about the system's operations and can provide insight into issues with the system. Use the journalctl command to view system logs.
Check service logs: If an application or service is not functioning correctly, check its logs to identify the issue. Service logs are typically located in the /var/log/ directory.
Use debugging tools: Linux provides a variety of debugging tools, including strace, lsof, and tcpdump. These tools allow you to monitor system calls, identify open files, and capture network traffic, respectively.
Check system resources: If an application is running slowly or not responding, check the system's resource usage to identify any bottlenecks. Use the top or htop command to view CPU and memory usage.
Check network connectivity: If you are experiencing network issues, use the ping command to check connectivity to other systems or websites. Use traceroute to identify network hops and latency.
Check file permissions: If an application is not functioning correctly, check that the correct file permissions are set for any files or directories it requires.
Check system updates: Ensure that the system is up to date and all updates have been applied. Use the apt-get update and apt-get upgrade commands to update the system.
Review configuration files: If an application is not functioning correctly, review its configuration files to ensure they are set correctly.

By following these troubleshooting and debugging techniques, you can identify and resolve issues with the Linux system, applications, and services.

29.Where system logs are located?

System logs in Linux are typically located in the /var/log/ directory. This directory contains a variety of log files that record system events, including kernel messages, authentication logs, system service logs, and application logs. The most commonly used log file is the syslog file, which contains messages from all system components.

30.How to follow file's content as it being appended without opening the file every time?

You can use the tail command with the -f option to follow a file's content as it is being appended without opening the file every time. Here is the syntax:

```
tail -f /path/to/file
```

This command will display the last 10 lines of the file, and then continue to display any new lines that are added to the file in real-time. You can stop following the file by pressing Ctrl + C. If you want to display more or fewer lines, you can use the -n option followed by the number of lines you want to display, like this:

```
tail -f -n 20 /path/to/file
```

This will display the last 20 lines of the file, and then continue to display any new lines that are added to the file in real-time.

31.What are you using for troubleshooting and debugging network issues?

Ping: A basic network troubleshooting tool that sends ICMP packets to a target device to check for connectivity. In Linux, you can use the ping command.

Traceroute: A tool that traces the route taken by a network packet from the source device to the destination, allowing network administrators to identify any intermediate hops where issues may be occurring. In Linux, you can use the traceroute or tracepath command.

Network analyzers: Tools like Wireshark capture network traffic and allow network administrators to analyze packet data and diagnose issues. Wireshark has a Linux version that can be installed and used.

Log analysis: Examining logs generated by network devices can provide valuable insights into network issues. In Linux, log files can be found in the /var/log/ directory.
Configuration analysis: Examining network device configurations can identify issues with routing, firewall rules, and other configuration settings. In Linux, network configurations can be found in files under the /etc/network/ directory.
IP command: Linux has a versatile command-line tool called ip that can be used to configure and troubleshoot networking. It provides information about network interfaces, routing tables, and other networking details.
Netcat: A simple command-line utility that can be used to test network connections and ports. It can be installed on Linux systems using the package manager and used to troubleshoot network issues.

These are the few examples of tools and techniques that can be used for network troubleshooting on Linux systems, and the specific tools and commands may vary depending on the distribution and version of Linux being used.

32.What are you using for troubleshooting and debugging disk & file system issues?

Disk health monitoring: Tools like smartmontools, which can be installed on Linux and other systems, can monitor the health of disks and notify administrators of any issues.
Disk space analysis: Tools like du, df, and ncdu can be used to analyze disk usage and identify files or directories that may be taking up too much space.
File system checks: Tools like fsck and chkdsk can be used to check the integrity of file systems and repair any issues.
RAID monitoring: Tools like mdadm can be used to monitor and manage RAID arrays, and notify administrators of any issues with disk redundancy.
Log analysis: Examining system logs can provide valuable information about disk and file system issues, including errors or warnings related to disk hardware or file system corruption.
File recovery: Tools like Photorec and TestDisk can be used to recover deleted or damaged files from disks and file systems.

Disk benchmarking: Tools like hdparm and fio can be used to benchmark disk performance and identify issues with disk read/write speeds or throughput.
It's important to note that troubleshooting disk and file system issues can be complex and may require a combination of tools and techniques. In addition, some issues may require the assistance of a trained IT professional or data recovery specialist.

33.What are you using for troubleshooting and debugging process issues?

Process monitoring: Tools like top, htop, and ps can be used to monitor running processes and identify any that may be consuming excessive resources or causing issues.
Process management: Tools like kill and pkill can be used to stop or restart processes that are causing issues or consuming excessive resources.
Log analysis: Examining system logs can provide valuable information about process issues, including errors or warnings related to process crashes or failures.
Resource monitoring: Tools like sar and sysstat can be used to monitor system resource usage over time, and identify any trends or anomalies that may be related to process issues.
System profiling: Tools like strace and ltrace can be used to trace the system calls and library calls made by a process, and identify any issues related to these calls.
Debugging tools: Tools like gdb and valgrind can be used to debug and analyze the behavior of running processes, and identify any issues related to memory usage, code execution, or other factors.
Application-specific tools: Many applications and services come with their own tools and diagnostic utilities that can be used to troubleshoot issues related to specific processes or services.

34.What are you using for debugging CPU related issues?

CPU monitoring: Tools like top, htop, and mpstat can be used to monitor CPU usage in real-time and identify any processes or services that may be consuming excessive CPU resources.

Performance profiling: Tools like perf and sysstat can be used to collect system-wide performance data and identify any CPU-related bottlenecks or issues.
System profiling: Tools like strace and ltrace can be used to trace the system calls and library calls made by a process, and identify any issues related to CPU usage.
Debugging tools: Tools like gdb and valgrind can be used to debug and analyze the behavior of running processes, and identify any issues related to CPU usage, such as excessive loops or inefficient algorithms.
Power management tools: Many modern CPUs come with power management features that can be configured to optimize CPU performance and reduce power consumption. Tools like cpufreq and powertop can be used to monitor and configure these features.
Kernel tuning: Some CPU-related issues may be related to kernel configuration settings. Tools like sysctl and tuning-advisor can be used to identify and adjust kernel settings related to CPU performance.
Hardware diagnostics: In some cases, CPU-related issues may be caused by hardware problems. Tools like memtest and mprime can be used to diagnose and test hardware components, including CPUs.

35.You get a call from someone claiming "my system is SLOW". What do you do?

If I received a call from someone claiming that their system is slow, I would follow these steps:
Ask the user to provide more information: I would start by asking the user to describe in detail what they mean by "slow". Is the system slow to start up? Are applications running slowly? Is the internet connection slow? This information will help me to narrow down the possible causes of the issue.
Check system resource usage: I would then check the system resource usage using tools like top or htop. This will help me to identify any processes or services that may be consuming excessive CPU, memory, or disk resources.
Check for malware or viruses: If the system is running slowly, it's possible that it has been infected with malware or viruses. I would run a full system scan using an anti-virus program or malware removal tool to check for any infections.

Check for disk space: If the system is low on disk space, it can slow down performance. I would check the available disk space using tools like df or du, and identify any files or directories that may be taking up too much space.
Check for updates: I would check for any pending updates to the operating system or applications that may be causing performance issues. Applying updates can often improve performance and fix bugs.
Check hardware components: If the system is still running slowly, I would check the hardware components, such as the hard drive, RAM, or CPU, to ensure that they are functioning properly. Tools like memtest and mprime can be used to diagnose hardware problems.
Provide recommendations and solutions: Once I have identified the cause of the performance issue, I would provide recommendations and solutions to the user. This may include removing unnecessary programs, upgrading hardware components, or optimizing system settings.
Overall, the goal would be to identify the root cause of the performance issue and provide the user with a solution that addresses the problem.

36.Explain iostat output

iostat is a command-line utility that can be used to monitor and report input/output (I/O) statistics for storage devices on a Linux system. The iostat command generates a report that includes a number of different metrics related to I/O performance, including the following:
Device: This column lists the name of the storage device that is being monitored, such as a hard drive or solid-state drive (SSD).
tps (transactions per second): This column shows the number of I/O operations that are occurring on the device per second, including both read and write operations.
kB_read/s and kB_wrtn/s: These columns show the amount of data that is being read from or written to the device per second, in kilobytes.
kB_read and kB_wrtn: These columns show the total amount of data that has been read from or written to the device, in kilobytes.
%util: This column shows the percentage of time that the device is busy handling I/O requests. A high %util value may indicate that the device is experiencing performance issues or may be reaching its maximum capacity.

In addition to these columns, the iostat output may also include information about CPU usage, memory usage, and other system-level statistics.

37.How to debug binaries?

Debugging binaries involves using tools and techniques to analyze and troubleshoot issues with compiled code. Here are some common steps that can be taken to debug binaries:

Use a debugger: A debugger is a tool that allows you to step through code line by line, set breakpoints, and examine the state of the program as it runs. Popular debuggers include gdb, lldb, and Visual Studio Code. Using a debugger can help you identify issues with code logic or memory management.

Enable debug symbols: When compiling code, it's important to enable debug symbols so that the binary includes information about the source code and variable names. This makes it easier to identify where issues are occurring and helps the debugger provide more meaningful information. To enable debug symbols, use compiler flags like -g.

Use logging and tracing: Adding logging statements or tracing code to the binary can help you identify where issues are occurring, what values are being passed to functions, and how the program is executing. This can be useful when trying to reproduce issues or understand unexpected behavior.

Use profiling tools: Profiling tools, such as valgrind or perf, can help you identify performance issues or memory leaks in the binary. These tools can provide insights into how the program is using system resources and help you optimize code.

Check system libraries and dependencies: Issues with the binary may be related to system libraries or dependencies that are missing or out of date. Use tools like ldd or strace to identify missing dependencies or issues with system calls.

Use static analysis tools: Static analysis tools, such as clang-tidy or cppcheck, can help you identify issues with code quality or security vulnerabilities. These tools analyze the code statically, without actually running it, and can help you identify issues that may be difficult to catch with a debugger.

Overall, debugging binaries can be a complex and time-consuming process. By using a combination of tools and techniques, you can gain insights into how the program is executing and identify issues that may be impacting performance or reliability.

38.What is the difference between CPU load and utilization?

CPU load and CPU utilization are both measures of the amount of work that the CPU is currently performing, but they represent different aspects of CPU performance.

CPU load refers to the number of processes or threads that are currently running on the CPU or waiting to run. A high CPU load indicates that there are many processes competing for CPU resources, which can cause the system to become slow or unresponsive. CPU load is typically measured as an average over a period of time, such as 1 minute or 5 minutes.

CPU utilization, on the other hand, refers to the percentage of time that the CPU is actively processing data. This includes both user processes and system processes such as interrupts and kernel operations. A high CPU utilization indicates that the CPU is busy and actively processing data. CPU utilization is typically measured as a percentage over a period of time, such as 1 second or 1 minute.

In summary, CPU load measures the number of processes that are waiting to run, while CPU utilization measures the percentage of time that the CPU is actively processing data. Both measures can be useful for understanding CPU performance and identifying potential issues, but they provide different insights into the behavior of the system.

39.How you measure time execution of a program?

There are several ways to measure the execution time of a program. Here are a few common methods:

Use the "time" command: On Linux or Unix systems, you can use the "time" command to measure the execution time of a program. Simply type "time" followed by the name of the program, like this:

```
$ time my_program
```

This will run "my_program" and display the execution time when it completes.

Use a profiler: Profiling tools, such as gprof or perf, can provide detailed information about the execution time of a program, including how much time is spent in each function or line of code. These tools can help you identify performance bottlenecks and optimize your code.

Use clock functions: Many programming languages provide functions for measuring the execution time of a program. For example, in C or C++, you can use the "clock" function to measure the CPU time used by a program. Here's an example:

```
#include <stdio.h>
#include <time.h>
 int main() {
clock_t start = clock();
// run some code here
 clock_t end = clock();
double elapsed_time = (double)(end - start) / CLOCKS_PER_SEC;
printf("Elapsed time: %f seconds\n", elapsed_time);
 return 0;
}
```

This program uses the "clock" function to measure the time between the start and end of the program, and then calculates the elapsed time in seconds.

Use performance counters: Modern CPUs often provide performance counters that can be used to measure various aspects of program execution, such as cache misses, branch mispredictions, or instruction retirements. Tools like perf or Intel VTune can be used to access these performance counters and provide detailed information about program performance.

Overall, the method you choose to measure the execution time of a program will depend on the programming language, operating system, and level of detail you need. By using a combination of methods, you can gain insights into how your program is performing and identify opportunities for optimization.

Scenarios

40.You have a process writing to a file. You don't know which process exactly, you just know the path of the file. You would like to kill the process as it's no longer needed. How would you achieve it?

To find the process writing to a specific file and then kill it, you can use the "fuser" command and the "kill" command in Linux. Here's how:

First, use the "fuser" command to find the process ID (PID) of the process writing to the file. For example, if the file path is "/path/to/file.txt", you would run:

```
$ fuser -v /path/to/file.txt
```

This will display a list of processes that are currently using the file, along with their PIDs.

Once you have the PID of the process writing to the file, you can use the "kill" command to terminate the process. For example, if the PID is 1234, you would run:

```
$ kill 1234
```

This will send a SIGTERM signal to the process, asking it to terminate gracefully. If the process does not respond to the SIGTERM signal, you can send a SIGKILL signal to force it to terminate:

```
$ kill -9 1234
```

This will send a SIGKILL signal to the process, which will immediately terminate it.

Kernel

41.What is a kernel, and what does it do?

A kernel is a key component of an operating system (OS). It acts as a bridge between software applications and the computer hardware. Essentially, the kernel is the core of the operating system that manages system resources, including the CPU, memory, disk I/O, and network connectivity.

Here are some of the main functions that the kernel performs:

Process management: The kernel manages the creation, scheduling, and termination of processes (i.e., programs that are currently running on the system).

Memory management: The kernel is responsible for allocating and deallocating memory for processes and managing virtual memory, which allows processes to use more memory than is physically available in the system.

Device drivers: The kernel provides device drivers for various hardware components, such as disk drives, network cards, and graphics cards. These drivers allow applications to communicate with the hardware.

File systems: The kernel provides file system support, allowing applications to read and write files on disk.

Security: The kernel enforces security policies by controlling access to system resources and providing authentication and authorization services.
Overall, the kernel is responsible for managing the low-level details of the system, allowing higher-level software to interact with the hardware in a standardized way. Without a kernel, it would be difficult for software applications to interact with the underlying hardware and utilize system resources efficiently.

42.How do you find out which Kernel version your system is using?

To find out which kernel version your system is currently running on, you can use the "uname" command in Linux. Here's how:

Open a terminal window on your Linux system.

Type the following command and press Enter:

```
uname -r
```

This will display the kernel version that your system is currently running on.
Alternatively, you can use the "cat" command to view the contents of the "/proc/version" file, which contains information about the kernel version:

```
cat /proc/version
```

This will display information about the kernel version, including the version number, build date, and compiler version.

Both of these commands should give you the information you need to determine which kernel version your Linux system is using.

43.What is a Linux kernel module and how do you load a new module?

A Linux kernel module is a piece of code that can be dynamically loaded and unloaded into the Linux kernel at runtime, without the need to recompile or reboot the system. Kernel modules provide a way to extend the functionality of the kernel without modifying the kernel source code or the system itself.

To load a new kernel module, you can use the "insmod" or "modprobe" command in Linux. Here's how:

Open a terminal window on your Linux system.

Check if the kernel module you want to load is already installed by running the following command:

```
lsmod | grep <module-name>
```

Replace "<module-name>" with the name of the kernel module you want to load. If the module is already installed, this command will display information about the module.

If the kernel module is not already installed, you can load it using the "insmod" command. For example, to load a module named "my_module.ko", you would run the following command:

sudo insmod /path/to/my_module.ko

Replace "/path/to/my_module.ko" with the actual path to the kernel module file.

Note that you need to have root or sudo privileges to load a kernel module.

Alternatively, you can use the "modprobe" command to load a kernel module and its dependencies automatically. For example, to load a module named "my_module.ko", you would run the following command:

sudo modprobe my_module

This will load the "my_module" kernel module, as well as any other modules that it depends on.

Note that the "modprobe" command looks for the module file in the default kernel module directory, which is usually "/lib/modules/<kernel-version>/". If your module file is located elsewhere, you can specify the full path to the file:

sudo modprobe /path/to/my_module.ko

In addition, you can specify options for the module by adding them to the modprobe command:

sudo modprobe my_module option1=value1 option2=value2 ...

This will load the "my_module" kernel module with the specified options.

Once the kernel module is loaded, you can use the "lsmod" command to check if the module is loaded and to view information about the module, including its size, dependencies, and use count. To unload a kernel module, you can use the "rmmod" command:

sudo rmmod my_module

This will unload the "my_module" kernel module from the system.

44.Explain user space vs. kernel space

User space refers to the portion of memory that is used by user-level applications and programs. This includes things like web browsers, word processors, and media players, as well as other user-level tools and utilities. User space programs run in a restricted environment and do not have direct access to hardware resources or low-level system functions. Instead, they rely on system calls to interact with the kernel.

Kernel space, on the other hand, is the portion of memory that is reserved for the operating system kernel. This space contains the most fundamental parts of the operating system, including the device drivers, hardware abstraction layer, and other critical system components. Kernel space programs have direct access to hardware resources and can execute low-level system functions.

The main difference between user space and kernel space is the level of privilege and access to system resources. User space programs are limited in their ability to interact with hardware and system functions, and rely on the kernel to perform these operations on their behalf. This is done for security reasons, as it prevents user-level programs from interfering with other system processes or causing damage to the system.

In contrast, kernel space programs have full access to hardware resources and can execute privileged operations directly. However, this also makes them potentially dangerous, as a poorly written kernel module or driver can crash the system or cause security vulnerabilities.

Overall, user space and kernel space are two distinct environments that serve different purposes in a modern operating system. While user space provides a safe and restricted environment for user-level programs to operate in, kernel space is where the critical system components reside and where low-level operations are performed.

45.In what phases of kernel lifecycle, can you change its configuration?

There are several phases in the lifecycle of the Linux kernel where you can change its configuration:

Compile time: During the initial compilation of the kernel, you can customize its configuration by selecting the appropriate options in the kernel configuration menu. This is done before the kernel is built and installed on the system.

Boot time: The kernel can be configured at boot time by passing command line options to the kernel. These options can be used to specify kernel parameters, such as the root file system, the amount of memory to use, or other system settings.

Runtime: Some kernel parameters can be modified at runtime using system tools such as sysctl or procfs. This allows for fine-tuning of system settings and can be useful in situations where a system is experiencing performance issues or other problems.

It's important to note that not all kernel parameters can be changed at runtime, and some may require a system reboot in order to take effect. Additionally, changing certain kernel parameters can have unintended consequences or may negatively impact system stability, so it's important to understand the implications of any changes before modifying the kernel configuration.

46.Where can you find kernel's configuration?

The kernel configuration file for a Linux system is typically located in the /boot directory and is named config-<version>. For example, on a Ubuntu system with kernel version 5.10.0, the configuration file would be located at /boot/config-5.10.0.

You can view the kernel configuration file using a text editor or command line utility such as cat or less. For example, to view the kernel configuration file for the currently running kernel on a Ubuntu system, you can use the following command:

```
less /boot/config-$(uname -r)
```

This will display the contents of the kernel configuration file in your terminal window, allowing you to view and edit the kernel configuration settings.

47.Where can you find the file that contains the command passed to the boot loader to run the kernel?

The file that contains the command passed to the boot loader to run the kernel is typically located in the /boot directory of the Linux file system and is named "grub.cfg" or "menu.lst" depending on the boot loader used.

In some Linux distributions, such as Ubuntu, the boot loader configuration file may be located in a subdirectory within /boot. For example, on Ubuntu systems using the GRUB2 bootloader, the configuration file is located at /boot/grub/grub.cfg.

You can view the contents of the boot loader configuration file using a text editor or command line utility such as cat or less. However, it's important to exercise caution when editing the boot loader configuration file, as making incorrect changes can prevent the system from booting properly. It's generally recommended to make a backup copy of the configuration file before making any modifications.

48.How to list kernel's runtime parameters?

ist the kernel's runtime parameters in Linux using the sysctl command.
To view all of the kernel's runtime parameters, run the following command:

```
sysctl -a
```

This will display a list of all of the kernel parameters, along with their current values.
You can also view the value of a specific kernel parameter by running the following command, replacing parameter_name with the name of the parameter you want to view:

```
sysctl parameter_name
```

For example, to view the value of the vm.swappiness parameter, which controls the tendency of the kernel to swap out inactive memory pages to disk, you can run the following command:

```
sysctl vm.swappiness
```

This will display the current value of the vm.swappiness parameter in your terminal window.

49.Will running sysctl -a as a regular user vs. root, produce different result?

Yes, running sysctl -a as a regular user will produce different results than running it as root. This is because sysctl command requires root privileges to view and modify certain kernel parameters that are protected by the kernel.
When run as a regular user, the sysctl -a command will only display the values of kernel parameters that are visible to unprivileged users. These parameters are typically related to the user's environment and do not require root privileges to access.
On the other hand, when run as root, the sysctl -a command will display the values of all kernel parameters, including those that are protected by the kernel and require root privileges to access. This allows the root user to view and modify a wider range of system settings, including network configuration, virtual memory settings, and more.

50.You would like to enable IPv4 forwarding in the kernel, how would you do it?

To enable IPv4 forwarding in the Linux kernel, you can use the sysctl command to modify the corresponding kernel parameter.

Open a terminal window and switch to the root user using the su command or sudo command.

Check the current value of the net.ipv4.ip_forward parameter by running the following command:

```
sysctl net.ipv4.ip_forward
```

By default, this parameter is usually set to 0, which means IPv4 forwarding is disabled.

To enable IPv4 forwarding, run the following command:

```
sysctl -w net.ipv4.ip_forward=1
```

This will set the value of the net.ipv4.ip_forward parameter to 1, which enables IPv4 forwarding.

Note that this change is temporary and will only last until the next reboot.

To make this change permanent, you can add the following line to the /etc/sysctl.conf file:

```
net.ipv4.ip_forward = 1
```

This will ensure that IPv4 forwarding is enabled every time the system boots up.

51.How sysctl applies the changes to kernel's runtime parameters the moment you run sysctl command?

The sysctl command applies changes to kernel's runtime parameters immediately by modifying the corresponding values in the kernel's virtual file system.

When you run a sysctl command to modify a kernel parameter, the command updates the corresponding file in the /proc/sys directory with the new value. This file is read by the kernel during its normal operation, so the new value is immediately applied to the kernel.

For example, if you run the following command to modify the net.ipv4.ip_forward parameter:

```
sysctl -w net.ipv4.ip_forward=1
```

The sysctl command will update the value of the net.ipv4.ip_forward file in the /proc/sys/net/ipv4 directory with the value 1. The kernel reads this file when it needs to access the value of the net.ipv4.ip_forward parameter, so the new value will be applied immediately.

This allows changes to be made to the kernel's runtime parameters without needing to reboot the system or restart any services. However, it's important to note that changes made using sysctl are not persistent across reboots unless they are saved to the /etc/sysctl.conf file or another configuration file.

52.How changes to kernel runtime parameters persist? (applied even after reboot to the system for example)

Changes to kernel runtime parameters can persist across reboots by modifying configuration files such as /etc/sysctl.conf or files in the /etc/sysctl.d/ directory.

The sysctl.conf file is a configuration file that specifies kernel parameters to be set at boot time. It is read by the sysctl command when the system boots, and the values specified in this file are applied to the kernel's runtime parameters.

To make changes to kernel runtime parameters persist across reboots, you can add lines to the /etc/sysctl.conf file in the following format:

Php

```
<parameter_name>=<value>
```

For example, to enable IPv4 forwarding, you can add the following line to /etc/sysctl.conf:

```
net.ipv4.ip_forward=1
```

After modifying /etc/sysctl.conf, you can apply the changes by running the sysctl command with the -p option:

Css

```
sysctl -p
```

This will load the values specified in /etc/sysctl.conf into the kernel's runtime parameters.

Alternatively, you can create a file with a .conf extension in the /etc/sysctl.d/ directory to store your changes. This is useful if you want to organize your changes into separate files. The files in /etc/sysctl.d/ are read in alphabetical order, so you can use a naming convention to ensure that the files are read in the correct order.

For example, you can create a file called /etc/sysctl.d/99-mysettings.conf with the following contents:

```
net.ipv4.ip_forward=1
```

This will enable IPv4 forwarding and ensure that the changes are applied at boot time.

53.Are the changes you make to kernel parameters in a container, affects also the kernel parameters of the host on which the container runs?

No, changes made to kernel parameters within a container do not affect the kernel parameters of the host on which the container runs. This is because a container is a self-contained environment that runs as a separate process on the host, with its own set of resources and configuration. The container has its own view of the system, including its own set of kernel parameters, which are isolated from the host system. However, it is still possible to configure the host system to affect the behavior of containers, for example by changing the default values of kernel parameters that affect container behavior.

SSH

54.What is SSH? How to check if a Linux server is running SSH?

SSH stands for Secure Shell and is a network protocol that allows secure remote access to a server or other computer over an unsecured network. SSH provides a secure, encrypted connection between the client and the server, allowing users to log in and run commands on the remote system as if they were physically present at the machine.

To check if a Linux server is running SSH, you can use the following command:

```
systemctl status sshd
```

This command will show the status of the SSH daemon (sshd) service on the server. If the service is running, you should see output similar to the following:

```
 sshd.service - OpenSSH server daemon
         Loaded: loaded (/usr/lib/systemd/system/sshd.service; enabled;
         vendor preset: enabled)
         Active: active (running) since Mon 2021-09-27 13:20:05 EDT; 5
         days ago
```

```
      Docs: man:sshd(8)
            man:sshd_config(5)
         Process: 1234 ExecStartPre=/usr/sbin/sshd -t (code=exited,
status=0/SUCCESS)
   Main PID: 1235 (sshd)
      Tasks: 1 (limit: 4915)
     Memory: 2.2M
     CGroup: /system.slice/sshd.service
             └─1235 /usr/sbin/sshd -D
```

If the service is not running, you will see a message indicating that the service is inactive or failed. In that case, you can start the service using the following command:

```
systemctl start sshd
```

Or enable it to start automatically at boot time:

```
systemctl enable sshd
```

55.Why SSH is considered better than telnet?

SSH is considered better than Telnet for several reasons:

Security: Telnet sends all data, including passwords, in clear text, which makes it vulnerable to network sniffing attacks. SSH, on the other hand, encrypts all data sent between the client and the server, which makes it much more secure.

Authentication: SSH supports several authentication methods, including public key authentication, which is more secure than Telnet's simple username and password authentication.

Portability: SSH is available on most operating systems, including Linux, Windows, and macOS, making it more versatile than Telnet.

Session management: SSH provides more robust session management features than Telnet, including the ability to resume interrupted sessions and to transfer files securely.

Overall, SSH provides a more secure and flexible way to access remote servers than Telnet, which is why it has become the de facto standard for remote access on Unix-like systems.

56.What is stored in ~/.ssh/known_hosts?

The ~/.ssh/known_hosts file contains a list of public keys of all the remote hosts that the user has previously connected to using SSH. When the user connects to a remote host using SSH, the host's public key is stored in this file, along with the host name and IP address.

The purpose of this file is to prevent man-in-the-middle attacks. When the user tries to connect to a remote host using SSH in the future, SSH compares the host's public key stored in known_hosts with the public key presented by the host during the connection attempt. If the keys match, then the connection is established. If the keys do not match, SSH will warn the user that the host's key has changed, which could indicate a security breach.

By default, the known_hosts file is located in the .ssh directory in the user's home directory.

57.You try to ssh to a server and you get "Host key verification failed". What does it mean?

"Host key verification failed" error message occurs when the SSH client fails to verify the authenticity of the remote server's host key.

SSH clients maintain a list of known host keys in the ~/.ssh/known_hosts file. When the SSH client connects to a server for the first time, it receives the host key from the server and prompts the user to confirm that they trust the key by adding it to the known_hosts file. Once the key is added, the SSH client can verify the server's identity on subsequent connections.

However, if the host key of the server changes unexpectedly (for example, due to a security breach or a server rebuild), the SSH client will not be able to verify the authenticity of the new key and will display the "Host key verification failed" error.

To resolve this issue, you can remove the old host key from the known_hosts file and add the new host key manually. Alternatively, you can contact the system administrator of the remote server to verify that the new host key is legitimate.

58.What is the difference between SSH and SSL?

SSH (Secure Shell) and SSL (Secure Sockets Layer) are both protocols used for secure communication over networks, but they serve different purposes and operate at different layers of the network stack.

SSH is a protocol that provides a secure, encrypted channel for remote access to a system's command line interface. It is primarily used for securely accessing and managing remote servers and network devices.

SSH provides a secure way to authenticate users and encrypts all data transmitted between the client and the server.

SSL, on the other hand, is a protocol that provides secure communication between applications over a network, typically over the internet. It is primarily used for securing web traffic and is used by HTTPS, the secure version of HTTP. SSL provides a secure way to encrypt data transmitted between a web server and a web client (such as a web browser).

In summary, SSH is primarily used for secure remote access and management of systems, while SSL is primarily used for securing web traffic.

59.What ssh-keygen is used for?

ssh-keygen is a command-line tool used for generating, managing, and converting authentication keys for SSH (Secure Shell) connections. Specifically, it is used to generate public and private key pairs for user authentication purposes. The private key is kept secret and stored on the user's computer, while the public key is placed on the remote server that the user wants to access. When the user attempts to connect to the server, the server checks the user's public key against the list of authorized keys on the server. If the key matches, the server grants the user access.

ssh-keygen can also be used to convert between different key formats, as well as to create digital signatures for user authentication purposes. It supports various encryption algorithms and key lengths.

60.What is SSH port forwarding?

SSH port forwarding, also known as SSH tunneling, is a technique used to forward network traffic securely from one network node to another over an encrypted SSH connection. It is a useful feature that allows users to establish secure connections between two machines without having to expose unencrypted traffic to the internet.

There are three types of SSH port forwarding:

Local Port Forwarding: It allows traffic coming to a specified port on the local machine to be forwarded to a specified host and port on the remote side.

Remote Port Forwarding: It allows traffic coming to a specified port on the remote machine to be forwarded to a specified host and port on the local side.

Dynamic Port Forwarding: It creates a dynamic SOCKS proxy on the local machine that forwards traffic through the SSH connection to any remote host.

SSH port forwarding can be used for a variety of purposes, including accessing remote services securely, bypassing firewalls and network restrictions, and encrypting traffic on unsecured networks.