Linux Assignment-2

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

In Linux FHS (Filesystem Hierarchy Standard), the / (root) is the top-level directory of the file system hierarchy. All other directories and files are contained within the root directory. It is represented as a forward slash / and it serves as the starting point for navigating the file system hierarchy on Linux systems. The root directory contains important system directories such as /bin, /usr, /var, and /etc. The root directory is the highest level of the file system hierarchy, and it is the parent directory of all other directories and files on the system.

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 binary executables (programs) that are required for the system to boot and function properly. These binaries are usually accessible to all users and are often used in system scripts. Examples of binaries stored in this directory include ls, cp, mv, mkdir, and bash.

/sbin: This directory contains binary executables that are used for system administration purposes. These executables are usually only accessible to the system administrator (root) and are often used in system scripts. Examples of binaries stored in this directory include shutdown, ifconfig, mount, and fdisk.

/usr/bin: This directory contains non-essential binary executables that are installed by the system or by individual users. These executables are usually accessible to all users and are often used in application scripts. Examples of binaries stored in this directory include curl, git, python3, and gcc.

/usr/sbin: This directory contains non-essential binary executables that are used for system administration purposes.

These executables are usually only accessible to the system administrator (root) and are often used in system scripts. Examples of binaries stored in this directory include sshd, dhcpd, httpd, and vsftpd.

/etc: This directory contains system-wide configuration files that are used by the system and by installed applications. Examples of files stored in this directory include passwd, group, hosts, resolv.conf, and fstab.

/home: This directory is the location of the home directories for all regular users on the system. Each user has their own subdirectory under /home that contains their personal files and settings.

/var: This directory contains variable data files that are created and modified by running processes. Examples of files stored in this directory include log files, databases, caches, and temporary files.

/tmp: This directory is used for temporary storage of files. It is often used by running processes for storing temporary data that is not needed after the process completes. The contents of this directory are typically deleted upon system reboot.

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

- /tmp is used to store temporary files.
- /tmp directory is used for writing temporary data and generally remove the data when it is no longer needed. Otherwise, the /tmp directory is cleared when the server restarts.
- The /tmp directory is often set up with more permissive permissions than other directories. This is to allow all users and processes to be able to create and access files in the directory.
- The /tmp directory is often configured to be automatically cleaned up at regular intervals to prevent the accumulation of old and unused files. This helps ensure that the directory remains available for use by running processes.

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

- Within the /proc/ directory, one can find a wealth of information detailing the system hardware and any processes currently running.

- Some of the files within the /proc/ directory tree can be manipulated by users and applications to communicate configuration changes to the kernel.
- This file shows the parameters passed to the kernel at the time it is started. The value of this information is in how the kernel was booted because any switches or special parameters will be listed.
- It provides detailed information about each running process on the system. This includes information such as process IDs, memory usage, CPU utilization and open files.
- It also provides information regarding system, device, network, and file system.

5. What makes /proc different from other filesystems?

- /proc is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo-file system.
- It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc). For this reason it can be regarded as a control and information centre for the kernel.
- In fact, quite a lot of system utilities are simply calls to files in this directory. For example, 'lsmod' is the same as 'cat /proc/modules' while 'lspci' is a synonym for 'cat /proc/pci'. By altering files located in this directory you can even read/change kernel parameters (sysctl) while the system is running.
- The most distinctive thing about files in this directory is the fact that all of them have a file size of 0, with the exception of kcore, mtrr and self.

6. True or False? only root can create files in /proc

False. In general, any user can create files in the /proc directory, but the files created by regular users will not have any system information or process information.

7. What can be found in /proc/cmdline?

The /proc/cmdline file contains the command-line arguments passed to the Linux kernel at boot time. These arguments specify various parameters such as the root file system, kernel parameters, and boot options.

The contents of /proc/cmdline will be a single line of text that includes various options and parameters separated by spaces. Some of the common parameters that may be found in /proc/cmdline include:

- root: This parameter specifies the root file system that the kernel should mount at boot time.
- quiet: This parameter tells the kernel to suppress most of its boot messages.
- loglevel: This parameter sets the level of kernel message logging, with higher values producing more detailed logs.
- init: This parameter specifies the path to the init process that should be executed after the kernel boots.
- ro: This parameter specifies that the root file system should be mounted read-only.
- rw: This parameter specifies that the root file system should be mounted read-write.

8. In which path can you find the system devices (e.g. block storage)?

The /dev directory is a special place in Linux. It is the location of files that can be used to represent the devices in your system.
Block storage devices such as hard drives, solid-state drives, and USB drives are typically represented as files with names that start with sd or hd. For example, /dev/sda might be the first hard drive on the system, while /dev/sdb might be the second.

Permissions

9. How to change the permissions of a file?

Permissions of a file can be changed using the chmod command. The chmod command allows to modify the read, write, and execute permissions for the owner, group, and others on a file.

chmod [permissions] [file]
where:
[permissions] - a three-digit number that represents the desired permissions.
[file] - filename.

Each digit is a sum of the permissions you want to grant, with read being 4, write being 2, and execute being you want to change.

For example, to give the owner read and write permissions, and the group and others read-only permissions on a file named example.txt, you could use the following command:
chmod 644 example.txt

10. What does the following permissions mean?

777 - owner, group, and others all have read, write, and execute permissions on the file

644 - owner has read and write permissions,
group and others have only read permissions.

750 - owner has read, write and execute permissions,
group has read and execute permissions,
others have no permissions.

11.What this command does? chmod +x some_file.

Sets the execute permission (+x) on the file named some_file.
The +x option tells the chmod command to add the execute permission to the file. Alternatively, -x to removes execute permission or = to set the exact permissions you want.

12. Explain what is setgid and setuid

setuid bit:
This bit is present for files which have executable permissions. The setuid bit simply indicates that when running the executable, it will set its permissions to that of the user who created it (owner), instead of setting it to the user who launched it.

- To set the setuid bit,
  chmod u+s
- To remove the setuid bit,
  chmod u-s

setgid bit:
The setgid affects both files as well as directories. When used on a file, it executes with the privileges of the group of the user who owns it instead of executing with those of the group of the user who executed it.

When the bit is set for a directory, the set of files in that directory will have the same group as the group of the parent directory, and not that of the user who created those files. This is used for file sharing since they can be now modified by all the users who are part of the group of the parent directory.

- To set the setgid bit,
  chmod g+s
- To remove the setgid bit,
  chmod g-s

13. What is the purpose of sticky bit?

The sticky bit was initially introduced to 'stick' an executable program's text segment in the swap space even after the program has completed execution, to speed up the subsequent runs of the same program. However, these days the sticky bit means something entirely different.
When a directory has the sticky bit set, its files can be deleted or renamed only by the file owner, directory owner and the root user.

- To set the sticky bit,
  chmod +t
- To remove the sticky bit,
  chmod -t

14. What the following commands do?

1.chmod

The chmod command is used to change the access mode of a file.
The name is an abbreviation of change mode.

Syntax :
chmod [reference][operator][mode] file...

The references are used to distinguish the users to whom the permissions apply i.e. they are list of letters that specifies whom to give permissions.
u->owner->file's owner.
g->group->users who are members of the file's group.
o->others->users who are neither the file's owner nor members of the file's group.
a->all->All three of the above, same as ugo.

The operator is used to specify how the modes of a file should be adjusted.
+ Adds the specified modes to the specified classes.

- Removes the specified modes from the specified classes.

= The modes specified are to be made the exact modes for the specified classes.

The modes indicate which permissions are to be granted or removed from the specified classes. There are three basic modes which correspond to the basic permissions:

r Permission to read the file.
w Permission to write (or delete) the file.
x Permission to execute the file, or, in the case of a directory, search it.

2.chown:

This command is used to change the owner of a file or directory. The owner is the user who has the most control over the file, including the ability to change permissions and modify the contents of the file.

Syntax :
chown [options] owner[:group] file,
where owner is the new owner and group is the new group ownership, if desired.
For example**, chown john:users some_file**
changes the owner to user "john" and the group ownership to "users".

3.chgrp:

This command is used to change the group ownership of a file or directory.

Syntax:
chgrp [options] group file
where group is the new group ownership and file is the file or directory to modify.
For example, **chgrp users some_file**
changes the group ownership to "users".

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

Allows a user to run commands with administrative or root privileges. It is used to perform tasks that require elevated permissions, such as installing software, modifying system files, or configuring the system.

Steps to setup sudo:

- Log in as the root user or a user with administrative privileges.
- Install sudo if it is not already installed on your system. You can use the package manager of your distribution to install it.
- Add the users who need to have sudo access to the sudoers file. This file is usually located at /etc/sudoers and can be edited using the visudo command, which opens the file in the default text editor.

- Save the changes to the sudoers file and exit the text editor. It's important to save the file using the visudo command, as it checks for syntax errors in the file before saving it.

Once you have set up sudo, authorized users can use it to execute commands with elevated privileges. To do so, they simply need to prefix the command with sudo.

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

True. In order to install packages on the system, one must have administrative privileges, which typically requires being the root user or using the sudo command. This is because installing packages can modify the system files and configurations, and can potentially cause security issues if done by an unauthorized user.

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

Access Control Lists (ACLs) are a type of file permission system in Linux and other Unix-based operating systems that allow more fine-grained control over file and directory permissions. ACLs extend the traditional Unix permissions system by allowing users and groups to be granted or denied specific permissions on a per-file or per-directory basis.

Use Cases:

- Shared directories: ACLs can be useful for shared directories where multiple users need access to specific files or directories. For example, if you have a directory where only certain users should be able to access specific files, you can use ACLs to grant or deny access to those files.

- Collaboration: In situations where multiple users are collaborating on a project, ACLs can be used to control access to specific files or directories. This can help ensure that only authorized users can modify or delete important files.

- Compliance: Some organizations require more granular control over access to sensitive files or directories to meet regulatory or compliance requirements. ACLs can help achieve this by allowing administrators to set specific permissions for individual 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?

- Permission denied: If the user does not have the necessary permissions to create a file in the specified directory, the creation will fail.
  This could happen if the user does not have write permissions for the directory, or if the directory is owned by another user.

- Disk full: If the disk where the file is being created is full or has insufficient free space, the file creation will fail. This could happen if the disk is running low on space or if the user is trying to create a file that is too large for the available space.

- Invalid filename: If the filename includes characters that are not allowed, such as special characters or spaces, the file creation will fail. The filename may also be too long or already exist in the specified directory.

- Other reasons why file creation could fail include file system errors, network connectivity issues, or resource exhaustion.

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

Here are the steps to fix this issue:

1.Use the cp command to make a copy of the chmod binary file:

sudo cp $(which chmod) /usr/local/bin/chmod_copy
This command will make a copy of the chmod binary file and save it as /usr/local/bin/chmod_copy.

2.Set the execute permission on the copied file:
sudo chmod +x /usr/local/bin/chmod_copy
This command will restore the execute permission on the copied chmod file.

3.Use the copied chmod command to restore the execute permission on the original chmod binary file:

sudo /usr/local/bin/chmod_copy +x $(which chmod)
This command will use the copied chmod command to restore the execute permission on the original chmod binary file.

Scenarios

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

Steps:
1.Open a terminal on the local computer and navigate to the directory where the file you want to copy is located.

2.Use the scp command to copy the file to the remote host.

Syntax for the scp command :
scp /path/to/local/file username@remote_host:/path/to/destination

For example:
scp example.txt ubuntu@192.168.1.100:~

3.Enter the password for the remote host when prompted.

4.Wait for the file to be copied to the remote host. Once the transfer is complete, you can close the terminal.

5.Make sure that the remote host has the necessary permissions to receive the file, and that any firewalls or network security settings allow the scp traffic.

21. How to generate a random string?

1.Using the openssl command:

openssl rand -base64 12
This command will generate a random string of 12 characters using base64 encoding. You can adjust the length of the string by changing the number after -base64.

2.Using the tr command:

tr -dc 'a-zA-Z0-9' < /dev/urandom | fold -w 12 | head -n 1
This command will generate a random string of 12 alphanumeric characters. You can adjust the length of the string by changing the number after fold -w.

3.Using the uuidgen command:

uuidgen | tr -d '\n'
This command will generate a random UUID string, which is a 36-character string consisting of alphanumeric characters and hyphens. The tr command is used to remove the newline character at the end of the string.

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

To generate a random string of 7 characters in Linux, you can use the openssl command as follows:

openssl rand -base64 5 | tr -dc 'a-zA-Z0-9' | cut -c1-7

This command will generate a random string of 5 bytes using base64 encoding, and then remove any characters that are not alphanumeric. Finally, it will use the cut command to extract the first 7 characters of the resulting string.

Alternatively, you can use the tr and head commands to generate a random string of 7 alphanumeric characters as follows:

tr -dc 'a-zA-Z0-9' < /dev/urandom | fold -w 7 | head -n 1
This command will generate a random string of 7 characters by selecting alphanumeric characters from /dev/urandom, wrapping the output to 7 characters using the fold command, and then selecting the first line of the output using the head command.

Systemd

23. What is systemd?

Systemd is a system and service manager for Linux operating systems. It is a replacement for the traditional SysVinit system initialization and service management scheme used in Unix-like systems.

Systemd provides a range of features, such as parallel startup of system services, on-demand service activation, and support for snapshotting and restoring the system state. It also provides a central management point for various system settings and resources, such as user sessions, network interfaces, mount points, and system timers.

Systemd has become the default system and service manager in many major Linux distributions, such as Red Hat Enterprise Linux, Fedora, Debian, and Ubuntu.

24. How to start or stop a service?

In a Linux system that uses systemd, you can use the systemctl command to start or stop a service.

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>
Again, replace <service_name> with the name of the service you want to stop.

You can also use the restart, reload, and status subcommands of systemctl to restart, reload, or check the status of a service, respectively:

sudo systemctl restart <service_name>
sudo systemctl reload <service_name>
sudo systemctl status <service_name>

25. How to check the status of a service?

In a Linux system that uses systemd, you can use the systemctl command to check the status of a service.

To check the status of a service, use the following command:

sudo systemctl status <service_name>
Replace <service_name> with the name of the service you want to check the status of.

This command will provide information about whether the service is running, its process ID (PID), the memory and CPU usage of the service, and any recent logs or error messages associated with the service.

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

On a system that uses systemd, you can use the journalctl command to display system logs. journalctl is a command-line utility that allows you to view logs from the systemd journal.

To display the most recent system logs, simply enter:
sudo journalctl
This will display the most recent system logs in reverse chronological order, with the most recent entries appearing first.

Various options to filter the logs by time, service, severity level, and more. For example, to display logs for a specific service, use the -u option followed by the service name:
sudo journalctl -u <service_name>
This will display the logs for the specified service.

Other options, such as -n to specify the number of log entries to display, -f to follow the logs in real-time, and -p to filter by severity level.

journalctl can be a powerful tool for troubleshooting and diagnosing problems on a Linux system that uses systemd.

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

To make a certain process/app a service in a Linux system that uses systemd, you will need to create a unit file for the service. A unit file is a configuration file that specifies how a service should be started, stopped, and managed by systemd.

Here are the basic steps to create a unit file and make a process/app a service:

1,Create a new file in the /etc/systemd/system/ directory with a .service extension. For example, if you want to make a service for the myapp process, you could create a file called /etc/systemd/system/myapp.service.

2. the file and specify the configuration options for the service. At minimum, you will need to specify the following:

- [Unit]: A section that defines the basic properties of the service, such as its description, dependencies, and required services.
- [Service]: A section that defines how the service should be started and managed by systemd, such as the command to run, the user and group to run as, and the working directory.
- [Install]: A section that specifies how the service should be installed and enabled at boot time.

Here is an example unit file for a service that runs the myapp process:

[Unit]
Description=My App
After=network.target

[Service]
User=myuser
Group=mygroup
WorkingDirectory=/path/to/myapp
ExecStart=/path/to/myapp/myapp

[Install]
WantedBy=multi-user.target

3.Save the file and reload the systemd daemon to read the new unit file:
sudo systemctl daemon-reload

4.Start the service:
sudo systemctl start myapp.service

You can also enable the service to start automatically at boot time:
sudo systemctl enable myapp.service

Once you have created the unit file and started the service, you can use the systemctl command to manage and monitor the service, just like any other systemd service. For example, you can use systemctl status myapp.service to check the status of the service, systemctl stop myapp.service to stop the service, and so on.

28. Troubleshooting and Debugging

Troubleshooting and debugging are important skills for any Linux system administrator or user. Here are some tips and techniques that can help you diagnose and resolve common problems:

1. Check the system logs: The system logs contain valuable information about what is happening on your system, including error messages, warnings, and other events. You can use tools like journalctl or tail to view the logs in real time, or examine log files in the /var/log/ directory.

2. Use the command line: The command line provides a powerful set of tools for troubleshooting and debugging. You can use commands like ps to view running processes, netstat to check network connections, and top to monitor system resources.

3. Check system resources: Problems with performance or stability can often be traced back to issues with system resources like CPU, memory, or disk space. You can use tools like df to check disk usage, free to check memory usage, and top or htop to monitor system resource usage in real time.

4. Use diagnostic tools: Linux provides a wide range of diagnostic tools that can help you diagnose and resolve specific problems. For example, you can use ping to test network connectivity, traceroute to trace the path of network traffic, and strace to trace system calls made by a process.

5. Try restarting the service: If you are experiencing problems with a particular service or application, try restarting it to see if that resolves the issue. You can use systemctl restart to restart a service, or simply stop and start it manually.

6. Check permissions and ownership: Many problems can be caused by incorrect file permissions or ownership. Use ls -l to check the permissions and ownership of a file or directory, and chmod and chown to modify them if necessary.

7. Consult documentation and online resources: If you are unable to resolve a problem on your own, consult the documentation or online resources for the software or system you are working with. There may be known issues or solutions that can help you resolve the problem more quickly.

29. Where system logs are located?

System logs in Linux are located in the /var/log/ directory. Each log file in this directory contains a different type of system or application log, such as syslog for system messages, auth.log for authentication messages, messages for kernel messages, and dmesg for boot messages. Other log files may be located in subdirectories of /var/log/ or in other system directories, depending on the specific application or system component. You can use the tail command to view the end of a log file in real time, or use tools like journalctl to view logs across multiple log files or sources.

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.
The syntax for the command is:

tail -f /path/to/file
This will display the last 10 lines of the file and continuously update the output as new lines are added to the file.

You can adjust the number of lines displayed with the -n option, like this:
tail -f -n 20 /path/to/file
This will display the last 20 lines of the file and continuously update the output as new lines are added. To exit the tail command and stop following the file, press Ctrl+C.

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

When troubleshooting and debugging network issues, typically use combination of the following tools:

1. ping: I use ping to test basic network connectivity between two devices, by sending ICMP packets and measuring the response time.

2. traceroute or mtr: These tools help identify the path that network traffic takes between two devices, by sending packets with increasing TTL values and noting the response from each hop along the way. mtr provides a more detailed view by continuously sending packets and updating statistics in real-time.

3. tcpdump or Wireshark: These packet sniffers allow me to capture and analyze network traffic in real-time, providing insight into the data being exchanged between devices.

4. netstat or ss: These tools provide information on active network connections, listening ports, and socket statistics, allowing me to identify which applications or services are generating network traffic.

5. ip or ifconfig: These tools provide information on network interfaces and IP addresses, allowing me to troubleshoot issues related to network configuration and routing.

6. dig or nslookup: These tools are used for DNS troubleshooting, allowing me to query DNS servers and inspect DNS resolution for specific domains or hostnames.

7. telnet or nc: These tools allow me to test network connectivity to specific ports on a remote device, and can be useful for identifying issues with firewall rules or network services.

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

When troubleshooting and debugging disk and file system issues, typically use this combination of the following tools:

1. dmesg: This tool displays kernel messages, including disk and file system related errors, that can help identify hardware or driver issues.

2. fdisk, parted, or lsblk: These tools display information about disks, partitions, and block devices, allowing me to inspect disk layout and partitioning.

3. mount or df: These tools display information about mounted file systems and disk usage, allowing me to verify that file systems are mounted correctly and have sufficient space.

4. smartctl: This tool provides information about disk health and status, including SMART data and self-tests.

5. fsck: This tool checks and repairs file system errors, including bad blocks and corrupted metadata.

6. lsof: This tool lists open files and the processes that have them open, allowing me to identify which processes are accessing specific files or directories.

7. strace or ltrace: These tools trace system calls and library calls respectively, allowing me to inspect the behavior of specific applications and identify issues related to file access or system calls.

8. debugfs: This tool allows me to manually inspect and modify file system metadata, which can be useful for recovering data from a corrupted file system or repairing issues with inode tables or directory structures.

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

When troubleshooting and debugging process issues, use this combination of the following tools:

1. ps or top: These tools display information about running processes, including the process ID (PID), CPU and memory usage, and status. They can help identify processes that are consuming excessive resources or have become unresponsive.

2. kill or pkill: These tools allow me to terminate a running process, either gracefully or with force. They can be useful for stopping a process that has become unresponsive or is consuming excessive resources.

3. strace or ltrace: These tools trace system calls and library calls, respectively, allowing me to inspect the behavior of specific applications and identify issues related to file access or system calls.

4. gdb: This tool is a debugger that allows me to attach to a running process or start a process in debug mode. It allows me to inspect and modify the state of a process at runtime, set breakpoints, and step through code.

5. systemctl: This tool allows me to manage systemd services, including starting, stopping, and restarting services. It can be useful for troubleshooting services that are not starting or stopping correctly.

6. journalctl: This tool displays the systemd journal, which contains information about system events and service status. It can be useful for identifying issues related to services or system events.

7. lsof: This tool lists open files and the processes that have them open, allowing me to identify which processes are accessing specific files or directories.

8. vmstat or sar: These tools display system performance statistics, including CPU usage, memory usage, and disk activity. They can help identify performance bottlenecks and resource contention issues that may be impacting process performance.

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

1. top or htop: These tools display system resource usage, including CPU usage, memory usage, and disk activity. They can help identify processes that are consuming excessive CPU resources.

2. ps or pidstat: These tools display information about running processes, including CPU usage, thread usage, and status. They can help identify processes that are consuming excessive CPU resources or have become unresponsive.

3. strace or ltrace: These tools trace system calls and library calls, respectively, allowing me to inspect the behavior of specific applications and identify issues related to CPU usage or system calls.

4. perf: This tool is a performance monitoring and profiling tool that can be used to analyze CPU usage and identify performance bottlenecks. It can be used to profile specific applications or system-wide CPU usage.

5. vmstat or sar: These tools display system performance statistics, including CPU usage, memory usage, and disk activity. They can help identify performance bottlenecks and resource contention issues that may be impacting CPU performance.

6. iotop: This tool displays I/O usage information, which can help identify processes that are consuming excessive disk I/O resources and impacting CPU performance.

7. mpstat: This tool displays CPU usage statistics for multiple processors, which can be useful for identifying issues related to CPU affinity or load balancing.

These tools can help identify and troubleshoot CPU-related issues on a Linux system.

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

When I receive a call from someone claiming that their system is slow, there are several steps I would take to diagnose and resolve the issue:

- Gather more information: I would ask the user to describe what they are experiencing in more detail. Is the entire system slow or just specific applications? When did the issue start? Have there been any recent changes to the system or any new software installations?

- Check system resources: I would use tools like top, htop, vmstat, or sar to check system resource usage, including CPU usage, memory usage, and disk activity. This can help identify if there are any resource contention issues that may be causing the system to slow down.

- Check logs: I would check system logs, including the syslog, kernel logs, and application logs, for any errors or warnings that may indicate issues that are impacting system performance.

- Check for malware: I would use a malware scanner to check the system for any malware infections that may be impacting system performance.

- Check for disk and file system issues: I would use tools like fsck, dmesg, and smartctl to check for any disk or file system issues that may be impacting system performance.

- Check for network issues: I would use tools like ping, traceroute, and tcpdump to check for any network issues that may be impacting system performance.

- Check for hardware issues: I would use hardware diagnostics tools to check for any hardware issues, such as failing memory or hard drives, that may be impacting system performance.

Once I have identified the root cause of the slow system, I would take steps to address the issue. This may involve installing software updates, removing malware infections, repairing disk or file system issues, or replacing faulty hardware components.

36. Explain iostat output

iostat is a system monitoring tool that reports input/output statistics for devices and partitions. The output of iostat includes a table with several columns that report various I/O statistics, including the device name, the number of reads and writes per second, the number of sectors read and written, the average request size in sectors, the average wait time, and the percentage of CPU time used.

Here is a breakdown of the columns in the iostat output:

- Device: The name of the device or partition being monitored.
- tps: Transfers per second, which is the number of read and write operations completed per second.
- kB_read/s: The amount of data read from the device or partition, in kilobytes per second.
- kB_wrtn/s: The amount of data written to the device or partition, in kilobytes per second.
- kB_read: The total amount of data read from the device or partition, in kilobytes.
- kB_wrtn: The total amount of data written to the device or partition, in kilobytes.
- r%util: The percentage of time that the device or partition was busy servicing requests.

37. How to debug binaries?

Debugging binaries typically involves using a debugger tool, such as gdb (GNU Debugger), to analyze and modify the behavior of the binary code during runtime.

Here are the basic steps to debug a binary using gdb:

- Compile the binary with debugging symbols: To enable debugging, you need to compile the binary with debugging symbols. You can do this by adding the -g flag to the compiler command.

- Start the binary with gdb: Run the binary with gdb using the following command: gdb <binary_file>.

- Set breakpoints: Set breakpoints at specific lines or functions of the binary code that you want to analyze using the break command.

- Start the program: Run the binary with the run command.

- Analyze and modify the program behavior: You can use various gdb commands to analyze and modify the binary code behavior during runtime.

- Some useful commands are: step to execute the next line of code, next to execute the next line of code and skip over function calls, print to display the value of a variable, and set to modify the value of a variable.

- Exit the debugger: When you are finished debugging, you can exit the debugger using the quit command.

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

- CPU load and utilization are two different concepts that refer to different aspects of CPU performance.

- CPU load is a measurement of the number of processes that are currently running on the CPU or waiting to run. It is typically expressed as a decimal number or percentage, with a value of 1.0 or 100% representing a fully utilized CPU.

CPU load takes into account both running processes and processes that are waiting to run. It is a measure of the workload on the CPU, and high CPU load can indicate that the system is under heavy demand.

- CPU utilization, on the other hand, is a measure of the amount of time the CPU spends executing non-idle processes. It is typically expressed as a percentage, with 100% indicating that the CPU is fully utilized. CPU utilization takes into account only running processes and not processes that are waiting to run. It is a measure of the efficiency of the CPU, and high CPU utilization can indicate that the CPU is performing well or that the system is underutilized.

In summary, CPU load measures the number of processes that are currently running or waiting to run, while CPU utilization measures the amount of time the CPU spends executing non-idle processes.

39. How you measure time execution of a program?

There are several ways to measure the execution time of a program: Using the time command: Simply prefix the command with the time command, and it will print the execution time of the program along with other resource utilization statistics like CPU usage and memory consumption.

Example: time ls -l

Using the date command: Run the command before and after the execution of the program, and calculate the difference between the two timestamps.

Example:

```
start=$(date +%s)
<command>
end=$(date +%s)
echo "Execution time: $((end - start)) seconds"
```

Using programming language-specific libraries: Many programming languages have built-in libraries or functions that can be used to measure the execution time of a program.

Example in Python:

```
import time
start = time.time()
<program>
end = time.time()
print(f"Execution time: {end - start} seconds")
```

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?

Fuser command is used to identify which process is currently accessing the file, and then the kill command is used to terminate it.

Command:

fuser -k /path/to/file

The -k option will send a SIGKILL signal to the process accessing the file. If you want to send a different signal, you can use the -SIGNAL option (e.g. -TERM for SIGTERM).

Note that killing a process without knowing exactly what it does can have unintended consequences, so use this approach with caution. It's always better to try and identify the process by name or PID if possible.

Kernel

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

In computing, the kernel is a central component of most operating systems. It is a program that controls and manages the system's hardware and resources, acting as a bridge between the hardware and software.

The kernel is responsible for providing low-level services to other parts of the operating system, such as memory management, process scheduling, device drivers, security, and more. It also provides an interface between the software applications and the hardware components, allowing the applications to interact with the hardware in a consistent and controlled manner.

In short, the kernel is the core of the operating system, providing a foundation for other programs to run on top of it, and managing the system's resources to ensure smooth and efficient operation.

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

To find out which kernel version your Linux system is using, use the uname command with the -r option, which displays the kernel release number:

uname -r

This command will output the version of the running kernel, such as 5.11.0-27-generic.

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. These modules can add new features to the kernel or extend its capabilities without the need to recompile the entire kernel.

To load a new module, you can use the modprobe command, which is used to add or remove modules from the Linux kernel. The syntax for loading a module with modprobe is:
sudo modprobe <module-name>

For example, to load the nvidia module, you can use:

sudo modprobe nvidia

This command will search for the nvidia module in the default module directory and load it into the kernel. If the module depends on other modules, modprobe will automatically load those modules as well.

If you have a module file in a non-default directory, you can specify the path to the module file with the insmod command, which directly inserts a module into the kernel:

sudo insmod /path/to/module.ko

This command will insert the specified module file into the kernel. Note that you need to have root privileges to load or unload kernel modules.

44. Explain user space vs. kernel space

User space	Kernel Space
Runs user-level applications and processes.	Runs the operating system's kernel and core services.
Accessible by user-level applications and processes.	Only accessible by the kernel and kernel modules.
Has limited access to hardware resources.	Has full access to hardware resources.
Provides an interface between the user-level applications and the kernel.	Handles system calls from user space, manages system resources, and provides core services like memory management and process scheduling.
Can be swapped out to disk by the kernel.	Cannot be swapped out to disk by the kernel.
Is not privileged and runs in a lower privilege mode.	Is privileged and runs in a higher privilege mode.

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

Linux kernel can change configuration in two phases of its lifecycle:

- Compilation phase: During this phase, you can configure the kernel by modifying its source code and recompiling it. This allows you to enable or disable various features, add, or remove device drivers, and so on.

- Runtime phase: Once the kernel has been compiled and loaded into memory, you can still modify its configuration during runtime by using various tools and interfaces provided by the kernel.

For example, you can modify kernel parameters using the sysctl interface or the /proc filesystem. You can also load and unload kernel modules dynamically to add or remove functionality from the kernel.

46. Where can you find kernel's configuration?

The kernel's configuration can be found in the /proc/config.gz file if the kernel configuration is compiled into the kernel. If the kernel configuration is compiled as a module, it can be found in the module directory /lib/modules/<kernel-version>/kernel/. Additionally, it can be viewed and modified during the kernel build process using the make menuconfig, make xconfig, or make gconfig commands.

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 depends on the specific boot loader being used.

In general, for systems that use the GRUB bootloader, the configuration file can be found at /boot/grub/grub.cfg or /boot/grub2/grub.cfg.

However, it is not recommended to edit this file directly as it is auto-generated by scripts. Instead, one should modify the appropriate configuration files located in /etc/default/grub or /etc/grub.d/.

Changes made to these files can be applied to the grub.cfg file by running the grub-mkconfig command.

48. How to list kernel's runtime parameters?

To list the kernel's runtime parameters, you can use the /proc/cmdline file. This file contains the command line options that were passed to the kernel at boot time.

To view the file, you can use the cat command:

cat /proc/cmdline

This will display a single line of text that shows the kernel's command line parameters. Each parameter is separated by a space.

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

Yes, running sysctl -a as a regular user vs. root will produce different results. The sysctl command is used to modify and view kernel parameters at runtime, which typically require root privileges to access.

If a regular user runs the sysctl -a command, they will only be able to see the subset of kernel parameters that are accessible to non-privileged users. On the other hand, if the command is run with root privileges, it will display all kernel parameters, including those that are restricted to root access only.

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

To enable IPv4 forwarding in the kernel, you can use the following command as root:

sysctl net.ipv4.ip_forward=1

This will enable IPv4 forwarding immediately in the running kernel. However, this change will not persist across reboots.

To make the change persistent, you can add the following line to the /etc/sysctl.conf file:
net.ipv4.ip_forward=1

After adding this line to the file, you can run the following command to reload the configuration:

sysctl -p

This will apply the changes to the running system.

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

- When we run the sysctl command, it reads and modifies the kernel's runtime parameters (also known as sysctl variables) stored in memory. The new values specified by the command are written to the corresponding variables in the kernel memory.
- This allows the kernel's behavior to be changed at runtime without requiring a system reboot. The changes are immediately applied to the system, affecting any processes that rely on the changed parameters.
- It's worth noting that changes made with the sysctl command are not permanent and will be lost on system reboot unless they are saved to a 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 be made persistent across reboots by modifying the system configuration files that are read during the boot process.

On Linux systems, these files are usually located in the /etc/sysctl.d/ directory and have a .conf file extension. You can create a new file in this directory or modify an existing one to add the desired kernel parameters and their values.

For example, to make the IPv4 forwarding setting persistent, you can create a new file /etc/sysctl.d/99-mysettings.conf with the following content:

net.ipv4.ip_forward=1

This will enable IPv4 forwarding every time the system boots up. To apply the new settings without rebooting, you can run the command sudo sysctl --system which will reload all configuration files in /etc/sysctl.d/.

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 inside a container only affect the kernel parameters of that container and do not affect the kernel parameters of the host system. Each container runs in its own isolated environment with its own set of kernel parameters.

SSH

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

SSH stands for Secure Shell, and it is a network protocol used to securely connect to a remote system. It provides an encrypted communication channel between a client and a server, allowing users to log in to a remote machine, execute commands, and transfer files securely over an unsecured network.

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

systemctl status sshd
This command checks the status of the SSH service, which is commonly named sshd in most Linux distributions. If the service is running, the command will output information about the service, including its PID (process ID), listening address and port, and more. If the service is not running, the command will indicate that the service is inactive or stopped.

55. Why SSH is considered better than telnet?

SSH (Secure Shell) is considered better than Telnet for several reasons:

- Encryption: SSH encrypts all traffic between the client and server, including passwords and commands, whereas Telnet sends all data in plaintext over the network, making it vulnerable to interception.

- Authentication: SSH provides strong authentication mechanisms, including public key authentication, which eliminates the need to transmit passwords over the network, reducing the risk of password theft.

- Security: SSH provides several security features that are not available in Telnet, such as port forwarding, X11 forwarding, and secure file transfer.

- Portability: SSH is available on a wide range of platforms, including Linux, macOS, Windows, and mobile devices, whereas Telnet is often limited to legacy systems.

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

The ~/.ssh/known_hosts file stores the public host keys of remote servers that the user has previously connected to using SSH. When the user connects to a remote server again, SSH will check the public key of the remote server against the one stored in the known_hosts file to ensure that the user is connecting to the same server and not to a potential attacker's server. If the public key of the remote server has changed since the last time the user connected, SSH will issue a warning to the user.

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

- The error message "Host key verification failed" indicates that the public key of the remote server that you are trying to connect to does not match the expected key stored in your local known_hosts file.
- This could be due to various reasons such as the remote server's key being regenerated or due to a man-in-the-middle (MITM) attack. In general, it means that the identity of the remote server cannot be verified and it is not safe to proceed with the connection.
- To resolve the issue, you can remove the entry for the remote server's hostname or IP address from your known_hosts file, and then try to reconnect to the server.

58. What is the difference between SSH and SSL?

SSH (Secure Shell)	SSL (Secure Sockets Layer)
Provides secure remote access to a machine via command line interface.	Provides secure communication between client and server applications over the internet.

Typically used for system administration, file transfers, and remote access to applications.	Typically used for secure web browsing, e-commerce transactions, and secure email communication.
Uses public-key cryptography for authentication and encryption.	Uses public-key and/or symmetric-key cryptography for authentication, encryption, and integrity.
Uses TCP port 22 by default.	Uses TCP port 443 by default.
Authentication is based on public and private key pairs, or username and password.	Authentication is based on digital certificates issued by trusted third-party Certificate Authorities.
Designed to provide secure remote access and data transfer within a network.	Designed to provide secure communication between applications across a network.
Can be used in combination with other protocols such as SCP, SFTP, and rsync for file transfers.	Can be used in combination with other protocols such as HTTP, FTPS, and SMTPS for secure communication.
Can be used to establish a secure tunnel for forwarding other protocols such as X11, VNC, and RDP.	Does not provide tunneling capabilities by itself.
Primarily used in Unix/Linux environments.	Used in a variety of operating systems and platforms including Windows, Linux, and macOS.

59. What ssh-keygen is used for?

- ssh-keygen is a command-line utility used for generating, managing, and converting authentication keys for SSH (Secure Shell) protocol.
- It is typically used for creating and managing public and private key pairs, which can be used for authentication between two machines or for accessing a remote server securely.
- The generated keys can also be used for other cryptographic purposes, such as encrypting data or verifying digital signatures.
- The utility supports various encryption algorithms and key types, and allows users to customize the keys by setting parameters such as key size and comment.

60. What is SSH port forwarding?

SSH port forwarding, also known as SSH tunneling, is a technique used to redirect network traffic from one machine to another through a secure SSH connection. It allows the user to access network services securely even if they are not directly reachable from the user's machine.

There are three types of SSH port forwarding:

1. Local port forwarding: This allows the user to forward traffic from a local port on their machine to a remote destination through an SSH server. It is useful when the user needs to access a service on a remote server that is not directly accessible from their machine. For example, if the user needs to access a database running on a remote server, they can forward traffic from a local port on their machine to the database port on the remote server through an SSH tunnel.

2. Remote port forwarding: This allows the user to forward traffic from a remote port on a remote server to a local destination through an SSH server. It is useful when the user needs to expose a service running on their machine to a remote server. For example, if the user is running a web server on their machine, they can forward traffic from a remote port on a remote server to the web server port on their machine through an SSH tunnel.

3. Dynamic port forwarding: This allows the user to create a SOCKS proxy on their machine through an SSH server. It is useful when the user needs to access multiple services on multiple remote servers through a single secure connection. For example, if the user needs to access a web server, a database server, and an email server on multiple remote servers, they can create a SOCKS proxy through an SSH tunnel and configure their web browser, database client, and email client to use the SOCKS proxy.