Linux Assignment – 2

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

   In the Linux FHS (Filesystem Hierarchy Standard), the / (root) directory is the highest-level directory in the filesystem hierarchy. It is the starting point for all file paths in Linux.

2. What is stored in each of the following paths?

/bin, /sbin, /usr/bin and /usr/sbin
/etc
/home
/var
/tmp

/bin : The '/bin' directory contains user binaries, executable files, Linux commands that are used in single user mode, and common commands that are used by all the users, like cat, cp, cd, ls, etc....

/sbin: The /sbin directory in Linux is a directory containing some executable files.

/usr/bin: It contains most user-level binary programs, which are installed by the system administrator or by the users themselves.

/usr/sbin: It contains system-level binary programs that are used for system administration tasks, just like /sbin.

/etc: It contains system-wide configuration files that control the behavior of various programs and services on the system.

/home: This directory contains the home directories for regular users on the system. Each user has their own subdirectory within "/home"

/var: This directory contains variable data files, such as logs, spool files, and temporary files that are generated during normal system operation.

/tmp: This directory contains temporary files that are used by various programs and services on the system.

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

The /tmp directory is typically used for **storing temporary files**.

This directory is usually accessible to all users on the system, which allows programs to easily share temporary files among users and it can be deleted any point of time without causing any problem.

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

In the /proc/ directory, one can find 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.

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 (ex: system memory etc..)

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

False

(/proc can only be modified by the root user or users with appropriate privileges)

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

/proc/cmdline file contains the command-line arguments that were passed to the kernel when the system was booted.

The /proc/cmdline file includes:

The kernel version

The initial RAM disk location and size

The root file system location and type.

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.

Permission:

9. How to change the permissions of a file?

We need to use the chmod command in a terminal or command prompt.

The chmod command allows you to change the read, write, and execute permissions of a file for the owner, group, and others.

Syntax: chmod [permissions] [filename]

Each permission is represented by a number or letter:

0 or --- indicates no permission

1 or --x indicates execute permission

2 or -w- indicates write permission

3 or -wx indicates write and execute permission

4 or r-- indicates read permission

5 or r-x indicates read and execute permission

6 or rw- indicates read and write permission

7 or rwx indicates read, write, and execute permission

10. What does the following permissions mean?:

777
644
750

777: This file mode means that the file owner, the group owner, and other users all have read, write, and execute permissions on the file.

644: This file mode means that the file owner has read and write permissions, and the group owner and other users have only read permission.

750: This file mode means that the file owner has read, write, and execute permissions, the group owner has read and execute permissions, and other users have no permissions.

11.What this command does? chmod +x some_file

The command chmod +x some_file makes the file some_file executable by adding the execute permission for the owner, group, and others.

The **+x** option adds the execute permission to the existing permissions of the file.

The some_file in the command is the name of the file that you want to make executable.

12. Explain what is setgid and setuid

Setgid is a permission bit that can be set on a directory.

When a new file is created in a directory with the setgid bit set, the group ownership of the file is set to the group that owns the directory, rather than the group of the user who created the file.

Setuid is a permission bit that can be set on an executable file.

When a user executes an executable file with the setuid bit set, the process runs with the privileges of the file owner, rather than the privileges of the user who executed the file.

13. What is the purpose of sticky bit?

The sticky bit is a special permission bit that can be set on directories in Unix-based systems.

When the sticky bit is set on a directory, it restricts the ability of users to delete or rename files within that directory, even if they have write permissions to the directory.

The purpose of the sticky bit is to allow users to share files in a directory while preventing accidental deletion or modification of other users' files.

14. What the following commands do?
chmod
chown
chgrp

chmod is a command in Unix-based systems that is used to change the permissions of a file or directory. It can be used to add or remove read, write, and execute permissions for the owner, group, and others.

chown is a command in Unix-based systems that is used to change the owner of a file or directory. It can be used to transfer ownership of a file or directory from one user to another.

chgrp is a command in Unix-based systems that is used to change the group ownership of a file or directory. It can be used to transfer group ownership of a file or directory from one group to another.

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

sudo (short for "superuser do") is a command in Unix-based systems that allows authorized users to execute commands as another user, typically the root user, who has elevated privileges.

To set up sudo, follow these steps:

. 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
(Only the root user or a user with sudo privileges can install packages on the system.)

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

ACL stands for Access Control List. It is a security mechanism that provides fine-grained access control to files and directories in Unix-based systems.

ACLs allow you to set permissions for individual users or groups beyond the traditional owner, group, and others permissions.

There are several use cases where ACLs can be useful:

Shared directories: When you have a directory that is shared among several users or groups, you may want to restrict access to specific files or subdirectories within that directory

Highly restricted directories: In some cases, you may have directories that need to be highly restricted to only specific users or groups..

Multi-user systems: In multi-user systems, it's common to have multiple users accessing the same files or directories. ACLs can be used to provide more fine-grained control over who can access, modify, or delete specific files or directories.

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

There could be several reasons why creating a file fails. Here are few possible reasons:

Permission issues: If the user does not have the appropriate permissions to create files in the directory, creating a file will fail.

Disk space issues: If there is not enough disk space available on the file system or quota for the user, creating a file will fail. This can happen if the file system is full or if the user has reached their disk space quota limit.

File name issues: If the file name is invalid or contains characters that are not allowed in file names, creating a file will fail.

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

If a user accidentally executed the command "chmod -x $(which chmod)", it would have removed the execute permission from the "chmod" command, making it unusable.
To fix this, you can use the absolute path of the "chmod" command to restore the execute permission.

Here are the steps to fix the issue:

. Open a terminal and log in as the root user or a user with sudo privileges.

. Run the following command to restore the execute permission on the "chmod" command: **chmod +x /bin/chmod**

. This will restore the execute permission on the "chmod" command, allowing it to be used again.

Scenarios :

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

To copy a file to a remote Linux host, you can use the scp (secure copy) command. The scp command allows you to securely transfer files between hosts using the SSH protocol.

The basic syntax for using scp to copy a file from a local host to a remote host is:

Code
scp /path/to/local/file username@remote:/path/to/remote/directory

Where /path/to/local/file is the path to the file you want to copy, username is the username of the remote host, remote is the hostname or IP address of the remote host, and /path/to/remote/directory is the directory on the remote host where you want to copy the file.

For example, to copy a file named example.txt from your local machine to a remote host with the IP address 192.168.1.100, you could use the following command:
Code
scp /path/to/local/example.txt user@192.168.1.100:/path/to/remote/directory
You will be prompted to enter the password for the remote user before the file transfer starts.

21. How to generate a random string?

In Linux, you can generate a random string using the openssl command or the uuidgen command.

To generate a random string of a specified length using the openssl command, you can use the following ,

syntax: **openssl rand -hex <length>**

Where <length> is the number of characters you want in the random string. For example, to generate a random string of 10 characters, you can run the following command:
Code : **openssl rand -hex 10**

To generate a random UUID (universally unique identifier) string using the uuidgen command, you can simply run the following command:
Code :**uuidgen**

This will generate a random UUID string that is 36 characters long, consisting of numbers and letters. If you want to generate a shorter or longer random string, you can use the -r option to specify the number of random bytes to use, and then convert the output to a string using a tool like base64 or hexdump.

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

To generate a random string of 7 characters using the openssl command, you can use the following command:

Code
openssl rand -base64 9 | cut -c1-7

This command generates a random string of 9 characters using the openssl command and the base64 encoding format. The cut command is then used to extract the first 7 characters of the output, giving you a random string of 7 characters.

Alternatively, you can use the tr command to remove any non-alphanumeric characters from the output, like this:
Code : **openssl rand -hex 4 | tr -dc 'a-zA-Z0-9' | cut -c1-7**

This command generates a random string of 4 bytes using the openssl command and the hex encoding format.

The tr command is then used to remove any non-alphanumeric characters from the output, and the cut command is used to extract the first 7 characters of the output.

Systemd

23. What is systemd?

Systemd is a system and service manager for Linux operating systems that replaces the traditional init system.

It is responsible for managing the boot process, system services, and daemons on a Linux system.

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?

The method to start or stop a service in Linux depends on the init system being used by the distribution. If the system is using the Systemd init system, the following commands can be used:
To start a service:

Code : **sudo systemctl start <service-name>**

To stop a service:

Code : **sudo systemctl stop <service-name>**

To restart a service:

Code : **sudo systemctl restart <service-name>**

To check the status of a service:

Code :**sudo systemctl status <service-name>**

Replace <service-name> with the actual name of the service that you want to start or stop.

If the system is using another init system like SysVinit, the commands may differ slightly. You can start or stop a service using the service command, like this:

To start a service:
Code : **sudo service <service-name> start**

To stop a service:
Code :**sudo service <service-name> stop**

To restart a service:
Code :**sudo service <service-name> restart**

To check the status of a service:
Code :**sudo service <service-name> status**

Again, replace <service-name> with the actual name of the service that you want to start or 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 if your system is using the Systemd init system. Simply run the following command:
Code :**sudo systemctl status <service-name>**

Replace <service-name> with the actual name of the service that you want to check the status of.

If your system is using another init system like SysVinit, you can use the service command to check the status of a service. Simply run the following command:

Code :**sudo service <service-name> status**

Again, replace <service-name> with the actual name of the service that you want to check the status of.

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

On a system which uses Systemd, you can display logs using the journalctl command. This command provides access to the logs collected by the systemd journal. Here are some useful examples:

To display all logs:

Code :**sudo journalctl**

To display logs for a specific unit/service:

Code : **sudo journalctl -u <service-name>**

Replace <service-name> with the actual name of the unit or service that you want to display logs for.

To display logs since the last system boot:

Code :**sudo journalctl -b**

To display logs for a specific time range:
Code :**sudo journalctl --since "2022-01-01 00:00:00" --until "2022-01-02 00:00:00"**
Replace the date and time values with the appropriate range.

To display logs with additional information like the system hostname and process IDs:

Code : **sudo journalctl -a**

These are just a few examples of how to use journalctl command. The command provides many more options and filtering capabilities to help you quickly find the information you need.

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

To make a certain process or application a service, you can create a Systemd unit file. Here are the general steps:

Create a new file in the /etc/systemd/system directory with a .service extension. For example:
bash
Code :**sudo nano /etc/systemd/system/myapp.service**

In this file, specify the necessary information about the service. Here's an example file content:
makefile

Code
[Unit]
Description=My Application
After=network.target

[Service]
ExecStart=/path/to/myapp
Restart=always
User=myuser
[Install]
WantedBy=multi-user.target

This file defines the service name, description, dependencies, and other important settings like the executable path, restart policy, and user account.

Save and close the file.
Reload the Systemd daemon to read the new unit file:
Code :**sudo systemctl daemon-reload**

Enable the service to start automatically on boot:
Code : **sudo systemctl enable myapp.service**

Start the service:
Code :**sudo systemctl start myapp.service**

Now the process or application specified in the unit file will run as a Systemd service, with all the benefits of being managed by the init system, such as automatic restarts, logging, and dependency management. You can check the status of the service using the systemctl status myapp.service command.

28. Troubleshooting and Debugging

Troubleshooting and debugging are essential skills for any Linux system administrator. Here are some general tips for troubleshooting and debugging Linux issues:

Check the logs: Most issues can be diagnosed by looking at the system logs. The logs can be found in /var/log/. Check the logs that correspond to the service or process that is having issues.

Use the command line: Many issues can be diagnosed and fixed using the command line. Use tools like ps, top, netstat, and lsof to diagnose the issue.

Check the network: If the issue involves networking, check the network configuration and connectivity. Use tools like ping, traceroute, and ifconfig to diagnose the issue.

Check the permissions: If the issue involves file or directory permissions, check the permissions using the ls -l command. Use the chmod and chown commands to modify permissions.

Check the configuration: If the issue involves configuration files, check the configuration files for syntax errors and typos. Use the diff command to compare the configuration files to a known good configuration.

Check the hardware: If the issue involves hardware, check the hardware for errors using tools like dmesg and smartctl. Check the system logs for any hardware error messages.

Check the service status: If the issue involves a service, check the service status using the systemctl status <service> command. Restart the service using the systemctl restart <service> command.

Google it: Don't be afraid to Google the error message or issue. Chances are, someone else has had the same issue and posted a solution on a forum or blog.

29. Where system logs are located?

System logs are located in the /var/log directory in Linux systems. Each service or application may have its own log file located in this directory.

Some of the commonly used log files include syslog, messages, auth.log, kernel.log, and dmesg.

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 the content of a file as it is being appended without opening the file every time. For example, to follow the content of a file called example.log, you can use the following command:
Code :**tail -f example.log**

This will display the contents of example.log and keep updating the display as new content is added to the file. To stop following the file, you can press Ctrl + C.

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

There are several tools that can be used for troubleshooting and debugging network issues in Linux, including:

ping: To check network connectivity and latency to a specific host.
traceroute: To trace the route taken by packets from the source to the destination.

netstat: To display network connections, routing tables, and other network statistics.

tcpdump: To capture and analyze network traffic in real-time.

ip: To display and manipulate routing, network devices, and network interfaces.

nslookup or dig: To perform DNS queries and resolve hostnames to IP addresses.

telnet or nc: To test network connectivity and test specific ports on a remote host.

These tools can provide valuable information for diagnosing and troubleshooting network issues in Linux.

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

There are several tools that can be used for troubleshooting and debugging disk and file system issues in Linux, including:

fsck: To check and repair the file system.

dmesg: To display kernel messages related to disk and file system errors.

lsof: To list open files and processes accessing them.

fdisk or parted: To view and manage disk partitions.

mount: To display mounted file systems and their options.

df: To display disk usage statistics for file systems.

du: To display disk usage statistics for directories and files.

These tools can help identify and fix issues with file systems and disks in Linux systems.

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

There are several tools that can be used for troubleshooting and debugging process issues in Linux, including:

ps: To display information about running processes.

top: To display real-time information about system processes, resource usage, and process statistics.

kill: To terminate a process.

pkill: To send a signal to a process based on its name.

strace: To trace system calls and signals made by a process.

lsof: To list open files and processes accessing them.

htop: An interactive version of top with more features and capabilities.

These tools can help identify and diagnose issues related to processes, such as high CPU

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

For debugging CPU-related issues, there are several tools available on Linux, including:

top: To display real-time information about system processes and CPU usage.

htop: An interactive version of top with more features and capabilities.

ps: To display information about running processes, including CPU usage.

perf: A powerful performance analysis tool that can be used to profile CPU usage and identify bottlenecks.

strace: To trace system calls and signals made by a process, which can help identify any CPU-intensive operations being performed by the process.

lsof: To list open files and processes accessing them, which can help identify any processes that are using excessive CPU resources.

Using these tools, you can monitor CPU usage and identify any processes or operations that are consuming a large amount of CPU resources.

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

If I receive a call from someone claiming that their system is slow, I would follow the following steps:

Ask for more information: I would ask the user for more information about the problem they are experiencing, such as when they noticed the system

slowing down, what actions they were performing at the time, and whether they have noticed any error messages or unusual behavior.

Check system resources: I would check the system's CPU, memory, and disk usage to see if there are any obvious resource constraints that could be causing the slowdown.

Check system logs: I would review the system logs to see if there are any errors or warnings that could be related to the slowdown.

Check for malware or viruses: I would run a virus scan or malware check to ensure that the system is not infected.

Check for running processes: I would check for any running processes that are consuming excessive CPU or memory resources and terminate them if necessary.

Check for disk and file system issues: I would check the disk and file system for any errors or issues that could be causing the slowdown.

Check for network issues: I would check the network connection and traffic to see if there are any issues that could be causing the slowdown.

By following these steps, I can help diagnose and resolve the issue and restore the system's performance.

36. Explain iostat output

iostat is a command-line tool that is used to monitor system input/output (I/O) device loading. The iostat output consists of a table that contains various statistics related to disk utilization and I/O activities. The table consists of the following columns:

Device: The name of the device (disk) being monitored.

tps: The number of transfers per second that were issued to the device.

kB_read/s: The amount of data in kilobytes read from the device per second.

kB_wrtn/s: The amount of data in kilobytes written to the device per second.

kB_read: The total amount of data in kilobytes read from the device.

kB_wrtn: The total amount of data in kilobytes written to the device.

%util: The percentage of time the device was busy handling I/O 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:

Code

```
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:

Code
```
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?

One way to achieve this is to use the lsof command to list all processes that have the file open, and then use the kill command to terminate the process. Here are the steps:

Use the lsof command to list all processes that have the file open. For example, if the file path is /path/to/file, you can run:

Code
```
lsof /path/to/file
```
Look for the process ID (PID) of the process that is writing to the file. It should be listed in the output of the lsof command.
Use the kill command to terminate the process. For example, if the PID of the process is 12345, you can run:

Code
```
kill 12345
```
Note that killing a process may have unintended consequences, so it should be done with caution.

Kernel

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

In computing, the kernel is a central component of an operating system (OS). It acts as a bridge between the software and the hardware, providing low-level services for memory management, process management, device management, and networking. The kernel controls all the system resources, such as the CPU, memory, and peripheral devices, and ensures that each process or task gets the necessary resources to operate correctly.

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

To find out the kernel version your system is using, you can use the uname command with the -r option. The command uname -r will display the kernel release version, which usually consists of three numbers separated by dots. For example:

Code

$ uname -r
5.11.0-41-generic
This output indicates that the system is using kernel version 5.11.0-41.

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 running Linux kernel. It can add new features or extend the functionality of the kernel without the need to recompile the kernel or reboot the system.

To load a new module, you can use the insmod command followed by the name of the module file, which has the extension .ko. For example:

Code
$ sudo insmod my_module.ko

You can check if the module was loaded successfully by running the lsmod command. If you want to unload a module, you can use the rmmod command followed by the name of the module, like this:

Code
$ sudo rmmod my_module

Keep in mind that some kernel modules are essential for the system to function properly, so unloading them can cause problems or even crashes.

44. Explain user space vs. kernel space

In operating systems, the user space and kernel space are two separate virtual address spaces. The kernel space is the part of the memory where the kernel and its drivers reside. It is protected and has complete control over the system's hardware resources. On the other hand, the user space is where the user's applications run. It is a restricted area that has no direct access to the hardware and relies on the kernel's system calls to interact with the hardware.

When a user process wants to execute a system call, it makes a request to the kernel by entering into kernel space. Once the system call is completed, the kernel returns to user space, and the user process resumes its execution.

This separation between user space and kernel space provides security and stability to the operating system. Since the kernel controls access to hardware resources, user processes cannot directly manipulate them, preventing possible system crashes or security breaches.

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

In general, you can change the kernel configuration during the build time and runtime.

During the build time, you can modify the configuration by using the make menuconfig, make xconfig, or makeconfig command. These commands allow you to select and configure various kernel options before building the kernel.

During runtime, you can modify the kernel configuration by editing the /proc/sys virtual file system, which contains runtime system parameters. These parameters can be changed using the sysctl command, which allows you to read, modify and set kernel parameters in real-time.

46. Where can you find kernel's configuration?

The kernel configuration can typically be found in the /boot/config-* file or in the /proc/config.gz file if the kernel is compiled with CONFIG_IKCONFIG_PROC.

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.

For GRUB (Grand Unified Bootloader), the configuration file is typically located at /boot/grub/grub.cfg or /boot/grub2/grub.cfg. However, it is not recommended to modify this file directly.

Instead, changes to the boot command line can be made in the file /etc/default/grub (or /etc/default/grub2, depending on the version of GRUB being used), followed by running the command sudo update-grub to regenerate the grub.cfg file with the updated settings.

48. How to list kernel's runtime parameters?

You can list the kernel's runtime parameters using the /proc/cmdline file. This file contains the kernel command-line parameters passed by the boot loader during system startup. You can display the contents of the /proc/cmdline file using the cat command, like this:

Code
cat /proc/cmdline

This will display a single line of text that lists all the kernel command-line parameters that were passed at boot time. Each parameter is separated by a space character.

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. Many kernel parameters can only be read or modified by the root user, and attempting to access or modify these parameters as a regular user will result in a permission denied error. Running the command with root privileges will provide access to all kernel parameters.

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

To enable IPv4 forwarding in the kernel, you need to set the value of the /proc/sys/net/ipv4/ip_forward file to 1.

You can do this using the following command as the root user or with sudo privileges:

Code
echo 1 > /proc/sys/net/ipv4/ip_forward
Alternatively, you can use the sysctl command to modify the kernel parameter:

Code
sysctl -w net.ipv4.ip_forward=1
To make the change persistent across reboots, you can modify the /etc/sysctl.conf file and add the following line:

Code
net.ipv4.ip_forward = 1
This will enable IPv4 forwarding in the kernel on the next system boot.

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

When you run the sysctl command to modify kernel parameters, it updates the values in the kernel's virtual file system (/proc/sys/) rather than writing them directly to the running kernel. The kernel then detects the change in this virtual file system and applies the new values to the running kernel immediately. This allows changes to be made to the kernel's runtime parameters without having to reboot the system.

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 by modifying the system configuration files where these parameters are defined. The location and syntax of these files may vary depending on the specific Linux distribution and version being used, but some common examples are:

/etc/sysctl.conf: This file contains a list of kernel parameters in the format parameter=value. Any changes made to this file will be applied at system startup.

/etc/default/grub: This file contains the command line options passed to the kernel when booting the system. Any changes made to this file will require running update-grub command to update the bootloader configuration.

/etc/modprobe.d/*.conf: This directory contains configuration files for kernel modules. Each file contains a list of parameters in the format options module_name parameter=value. Any changes made to these files will be applied when the corresponding module is loaded.

After modifying these files, the changes can be applied immediately by running the sysctl -p command or rebooting the system.

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, the changes made to the kernel parameters in a container do not affect the kernel parameters of the host on which the container runs. This is because containers use the same kernel as the host but have their own isolated user space, which means that changes made within the container only affect the container itself and not the host.

SSH

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

SSH (Secure Shell) is a network protocol used for secure remote access to a computer or server. It is typically used to log into a remote machine and execute commands, but it also supports tunneling, forwarding TCP ports, and X11 connections.

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

Code
systemctl status sshd
This command will show the status of the SSH service and whether it is running or not. If the service is running, it means the server is running SSH.

55. Why SSH is considered better than telnet?

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

Encryption: SSH encrypts all data transmitted between the client and the server, providing secure remote access to servers and preventing eavesdropping by attackers.

Authentication: SSH provides stronger authentication mechanisms than telnet, which relies on clear text passwords that can be easily intercepted by attackers.

Data integrity: SSH provides data integrity checks to ensure that data is not altered during transmission.

Port forwarding: SSH allows for secure port forwarding, enabling users to securely access remote services without exposing them to the public internet.

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

The ~/.ssh/known_hosts file stores a list of known remote hosts for the SSH client. Each line in the file represents a host, including the hostname and the public key of the host. When the SSH client connects to a remote host, it checks the host key presented by the remote host against the keys in known_hosts file. If the key matches, the client knows that it is safe to connect to the remote host. If the key does not match, the client will display a warning that the remote host's identity is unknown or has changed.

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 SSH client cannot verify the authenticity of the server. This can happen if the server's public key has changed, or if the client's known_hosts file has been modified.

The SSH client stores the public key of the server in the known_hosts file after the first connection to the server. On subsequent connections, the client checks the server's public key against the one stored in known_hosts. If they don't match, the client will issue the "Host key verification failed" error message to prevent a potential man-in-the-middle attack.

To resolve this issue, you can either remove the entry for the server in the known_hosts file (if you're sure that the change is legitimate), or manually verify the server's public key fingerprint and update the known_hosts file accordingly.

58. What is the difference between SSH and SSL?

SSH and SSL are both security protocols that provide encryption and authentication, but they serve different purposes.

SSH (Secure Shell) is a protocol used for secure remote access to a computer or server over an unsecured network. It provides secure authentication, data encryption, and confidentiality between two systems.
SSL (Secure Sockets Layer) is a protocol used to secure communication over the internet, typically between a web server and a web browser. SSL provides encryption and authentication for data transmitted over the network and is commonly used for secure transactions, such as online shopping or banking.

In summary, SSH is used for secure remote access to a computer or server, while SSL is used to secure communication over the internet.

59. What ssh-keygen is used for?

The ssh-keygen command is used to generate, manage and convert authentication keys for SSH. It is typically used to generate pairs of public and private keys, which can be used for passwordless authentication to a remote server or for other purposes such as encryption or digital signatures. The ssh-keygen command can also be used to generate host keys for SSH servers. These keys are used to verify the authenticity of the server to clients when they connect.

60. What is SSH port forwarding?

SSH port forwarding, also known as SSH tunneling, is a method of forwarding network traffic from one network port to another, over a secure SSH connection. This enables the user to access network services, such as databases, web servers, or other applications that are normally only available on a remote machine.

There are three types of SSH port forwarding:

Local port forwarding: it forwards traffic from a local port to a remote host and port, allowing the user to access a service on the remote machine as if it were local.

Remote port forwarding: it forwards traffic from a remote port to a local host and port, allowing a service running on a remote machine to be accessed from a local network.

Dynamic port forwarding: it creates a SOCKS proxy on a local port that can be used to forward traffic through an SSH connection to any destination host and port.
SSH port forwarding is a powerful and flexible feature that can be used for a variety of purposes, such as securely accessing a remote server behind a firewall, forwarding a remote desktop session over an encrypted connection, or accessing a database that is not directly accessible from the internet