LINUX ASSIGNMENT 2

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

In the Linux Filesystem Hierarchy Standard (FHS), the root directory is denoted by the forward slash symbol (/). It is the highest-level directory in the filesystem hierarchy and serves as the starting point for all other directories and files in the system. All other directories are either directly or indirectly located under the root directory.

The root directory contains important system directories such as /bin, /etc, /home, /dev, /lib, /usr, /var, and others. It also contains files that are critical to the system's operation, such as the kernel, boot loader, and configuration files.

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 system binaries (executable programs) that are required for booting and repairing the system, such as the ls, cp, and mv commands.
- **/sbin:** This directory contains essential system binaries that are primarily used by the system administrator, such as system daemons, network utilities, and disk management tools. Examples include the init, fdisk, and ifconfig commands.
- **/usr/bin:** This directory contains non-essential system binaries that are used by general users, such as command-line tools, programming languages, and editors.
- **/usr/sbin:** This directory contains non-essential system binaries that are primarily used by the system administrator, such as network servers, system daemons, and other administrative tools.
- **/etc:** This directory contains system-wide configuration files, such as those for the network, users, and system services.
- **/home:** This directory contains the home directories for regular users. Each user has a subdirectory here with their username, where they can store personal files and settings.
- **/var:** This directory contains variable data files, such as log files, spool files, and caches, that are expected to grow in size as the system runs.
- **/tmp:** This directory is used for temporary files that are created and destroyed during system operation, such as files created by applications or the system itself. It is often cleared at system boot or periodically to free up disk space.

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

The /tmp directory is a special directory in the Linux filesystem hierarchy for temporary files. It differs from other directories in a few ways:

1. **Purpose:** The /tmp directory is specifically intended for the storage of temporary files that are created and deleted during system operation. It is not meant to be used for long-term storage, as the contents of the directory may be cleared periodically.

2. **Permissions:** The /tmp directory is typically world-writable, meaning that any user on the system can create or delete files within it. This is because any user or process may need to create temporary files in this directory, and restrictive permissions could impede the normal functioning of the system.

3. **Clean-up:** The contents of the /tmp directory are often automatically cleared on system boot or periodically by system processes. This helps prevent the directory from becoming cluttered with unused temporary files that could take up valuable disk space.

4. **Security:** Due to the world-writable nature of the /tmp directory, it can be a security risk if it is not properly managed. Malicious users or processes could potentially create files or execute code in the /tmp directory, which could compromise the security of the system. To mitigate this risk, some system administrators may choose to mount the /tmp directory as a separate filesystem or use additional security measures such as access controls or auditing.

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

The /proc directory in Linux is a special directory that contains a virtual filesystem that provides information about the current state of the system. The files in the /proc directory are not real files on the filesystem, but rather virtual files that are generated on the fly by the kernel to provide information about various system resources and processes. Here are some examples of the kind of information that can be found in the /proc directory:

1. **System information:** The /proc/cpuinfo file contains information about the CPU(s) in the system, including their clock speed, cache size, and architecture. The /proc/meminfo file contains information about the system's memory usage and configuration.

2. **Process information:** The /proc/PID directory (where PID is the process ID) contains information about a particular process, such as its memory usage, file descriptors, and status.

3. **Network information:** The /proc/net directory contains information about the state of network interfaces, active connections, and other network-related information.

4. **Hardware information:** The /proc/interrupts file contains information about the interrupt requests being handled by the system, while the /proc/ioports file contains information about the input/output ports being used by the system.

5. **Kernel information:** The /proc/sys directory contains a variety of system configuration parameters that can be modified to change the behavior of the kernel.

Overall, the /proc directory is a powerful tool for system administrators and developers to gather information about the current state of the system and diagnose problems. However, it should be used with caution, as modifying or deleting files in the /proc directory can have serious consequences for the stability and security of the system.

5. What makes /proc different from other filesystems?

The /proc filesystem is different from other filesystems in several ways:

1. **It is a virtual filesystem:** Unlike most other filesystems that store data on disk, the /proc filesystem is a virtual filesystem that exists only in memory. The files and directories in the /proc filesystem are not actual files on a disk, but rather a way for the kernel to expose information about the system and its processes.

2. **It is dynamic:** The contents of the /proc filesystem are generated on the fly by the kernel, and they can change at any time. This means that the information in the /proc filesystem is always up to date, and it can be accessed and modified by any process with the appropriate permissions.

3. **It exposes system information:** The /proc filesystem is primarily used to provide information about the system and its processes. It contains files and directories that expose information about the system's hardware, kernel configuration, processes, network connections, and more.

4. **It is read-only:** While some files in the /proc filesystem can be modified, such as those that control kernel parameters, most of the files are read-only. This is because they are generated by the kernel and any changes made to them could have unexpected consequences.

5. **It is a hierarchy:** Like other filesystems, the /proc filesystem is organized as a hierarchy of directories and files. However, the structure of the hierarchy is not based on physical storage locations or devices, but rather on the relationships between different system resources.

Overall, the /proc filesystem is a powerful tool for system administrators and developers to access and modify information about the system and its processes. However, because it is a dynamic and virtual filesystem, it should be used with caution, and changes should only be made after careful consideration of the potential consequences.

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

False.

Files can be created in the /proc directory by any process that has the appropriate permissions. However, most of the files in the /proc directory are created and managed by the kernel and modifying them without the proper knowledge and care can have serious consequences for the stability and security of the system. Therefore, it is generally not recommended to create or modify files in the /proc directory unless you have a specific reason to do so and understand the implications of your actions.

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

The /proc/cmdline file contains the command-line arguments passed to the kernel at boot time. This includes any options set in the bootloader configuration, such as the kernel image, the root device, and any kernel parameters.

Some of the information that can be found in the /proc/cmdline file includes:

1. **Bootloader information:** The bootloader used to start the system, such as GRUB or LILO.

2. **Kernel image:** The location of the kernel image on the filesystem.

3. **Root device:** The device that contains the root filesystem.

4. **Kernel parameters:** Any additional parameters passed to the kernel at boot time, such as options to control hardware or system behavior.

Accessing the /proc/cmdline file can be useful for system administrators and developers who need to troubleshoot boot-time issues or verify the kernel configuration on a running system. However, it should be used with caution, as modifying the kernel parameters in the /proc/cmdline file can have serious consequences for the stability and security of the system.

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

In Linux, the system devices are typically represented as files in the /dev directory. Block storage devices, such as hard drives and solid-state drives, are represented as device files with names like /dev/sda, /dev/sdb, etc. Other devices, such as serial ports, network interfaces, and sound cards, are also represented as files in the /dev directory.

The /dev directory is a special directory that provides access to the devices in the system as if they were regular files. This allows programs and system components to interact with hardware devices using standard file input/output operations, such as reading and writing.

Note that not all devices are represented as files in the /dev directory. Some devices, such as USB devices, are not created until they are plugged in, and may not be visible in the /dev directory until they are detected by the system. Additionally, some devices may be represented in different ways on different systems or with different Linux distributions.

PERMISSIONS

9. How to change the permissions of a file?

To change the permissions of a file, you can use the 'chmod' command in a terminal or command prompt.
The 'chmod' command stands for "change mode" and allows you to modify the permissions of a file or directory.
Here's the basic syntax of the 'chmod' command:

```
chmod [permissions] [filename]
```

The '[permissions]' parameter specifies the new permissions that you want to set for the file. There are three types of permissions:

- 'r'(read): Allows the file to be read
- 'w'(write): Allows the file to be modified
- 'x'(execute): Allows the file to be executed as a program

Each permission can be assigned to three categories of users:

- 'u'(user): The owner of the file
- 'g'(group): Users in the same group as the file
- 'o'(other): All other users

You can specify the permissions using a numeric code or a symbolic code. Here's an example of using the symbolic code:

```
chmod u+rwx,g+rx,o+r myfile.txt
```

If you want to remove a permission, you can use a hyphen - in front of the permission. For example, to remove write permission for the owner of the file, you can use the following command:

```
chmod u-w myfile.txt
```

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

- **777:** This permission code gives read (r), write (w), and execute (x) permissions to the owner of the file or directory, as well as to users in the same group and to all other users. This means that anyone can read, write, and execute the file or directory.

- **644:** This permission code gives read (r) permission to the owner of the file or directory and read permission to users in the same group and to all other users. It also gives write permission (w) to the owner of the file or directory. This means that only the owner of the file or directory can modify it, but anyone can read it.

- **750:** This permission code gives read (r), write (w), and execute (x) permissions to the owner of the file or directory, and gives read and execute permissions to users in the same group. It does not give any permissions to other users. This means that only the owner and users in the same group can modify or execute the file or directory, but other users cannot access it.

11.What this command does? chmod +x some_file

The command 'chmod +x some_file' gives the execute ('x') permission to the file 'some_file'. The '+x' parameter adds the execute permission to the file. This means that the file can be executed as a program or script.

The 'chmod' command is used to change the permissions of files and directories in Unix-based operating systems. The '+x' parameter specifies to add the execute permission, and the 'some_file' parameter specifies the file whose permissions are being changed.

Note that in order to execute a file, it must also have the read permission (r). If the file does not have the read permission, you may need to add it using the 'chmod +r some_file' command before adding the execute permission.

In summary, the 'chmod +x some_file' command makes the file some_file executable.

12. Explain what is setgid and setuid.

'setgid' and 'setuid' are special permissions that can be set on files and directories in Unix-based operating systems. These permissions allow a process to be executed with the privileges of the file's owner or group, rather than the privileges of the user who executed the process.

Here's a brief explanation of each permission:

- **'setgid':** When the 'setgid' permission is set on a directory, any files or subdirectories created in that directory inherit the group ownership of the parent directory. This means that all users who are members of the parent directory's group have the same permissions on the files in the child directory, even if they did not create the files. When the 'setgid' permission is set on a file, any process that runs the file is executed with the group ownership of the file, rather than the group ownership of the user who executed the process.

- **'setuid':** When the 'setuid' permission is set on a file, any process that runs the file is executed with the user ID of the file owner, rather than the user ID of the user who executed the process. This means that the process is executed with the same privileges as the file owner, which can be useful for running privileged commands or applications that require elevated privileges. However, setuid can also pose security risks if used improperly, as it can allow unauthorized access to privileged resources.

To set the 'setgid' or 'setuid' permissions on a file or directory, you can use the 'chmod' command with the 'g+s' or 'u+s' parameters, respectively.

For example, to set the 'setgid' permission on a directory called 'mydir', you can use the following command:

```
chmod g+s mydir
```

Similarly, to set the 'setuid' permission on a file called 'myapp', you can use the following command:

```
chmod u+s myapp
```

13. What is the purpose of sticky bit?

The sticky bit is a special permission that can be set on directories in Unix-based operating systems. When the sticky bit is set on a directory, it ensures that only the owner of a file or directory, or the root user, can delete or rename the file or directory. This can be useful in situations where multiple users have write access to a shared directory, but you want to ensure that users can only delete or modify their own files.

When the sticky bit is set on a directory, it is represented by the letter "t" at the end of the directory's permission string. For example, if the directory '/mydir' has the sticky bit set, its permission string would look like this:

```
drwxrwxrwt
```

The sticky bit can be set using the 'chmod' command, with the '+t' parameter. For example, to set the sticky bit on the directory '/mydir', you can use the following command:

```
chmod +t /mydir
```

In summary, the sticky bit is a permission that ensures that only the owner of a file or directory, or the root user, can delete or rename the file or directory. This can help prevent accidental deletion or modification of files in shared directories.

14. What the following commands do? chmod ,chown ,Chgrp

- **'chmod':** This command is used to change the permissions of a file or directory in Linux/Unix. It stands for "change mode". The 'chmod' command modifies the read, write, and execute permissions for the owner, group, and other users. The permissions are specified using a three-digit code, where each digit represents the permissions for the owner, group, and others. For example, 'chmod 755 file.txt' would give the owner read, write, and execute permissions, while group and others would have only read and execute permissions.

- **'chown':** This command is used to change the ownership of a file or directory in Linux/Unix. It stands for "change owner". The 'chown' command allows you to change the owner and group of a file or directory. For example, 'chown user:group file.txt' would change the owner to "user" and the group to "group".

- **'chgrp':** This command is used to change the group ownership of a file or directory in Linux/Unix. It stands for "change group". The 'chgrp' command allows you to change the group of a file or directory. For example, 'chgrp group file.txt' would change the group ownership of the file to "group".

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

'sudo' is a command in Unix/Linux-based operating systems that allows a user to run commands with elevated privileges or as another user, typically the root user. This is useful for performing administrative tasks that require elevated privileges, such as installing software or modifying system files.

To set up 'sudo':

1. Log in as the root user or a user with administrative privileges.
2. Install the 'sudo' package if it is not already installed on your system.

3. Open the '/etc/sudoers' file using a text editor, preferably 'visudo' which checks syntax errors before saving the file.
4. Add a line to the file specifying which user(s) or group(s) should have access to 'sudo'. For example, to give the user "johndoe" access to 'sudo', add the following line to the file:

    johndoe ALL=(ALL) ALL

This allows the user "johndoe" to run any command with elevated privileges using 'sudo'.
5.Save and close the file.
Now, the user "johndoe" can use 'sudo' to run commands with elevated privileges by prefixing the command with 'sudo', and entering their password if prompted.

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 a system, one must have administrative privileges. This can be achieved by either logging in as the root user or by using the 'sudo' command to run commands with elevated privileges.

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

ACLs (Access Control Lists) are a type of security mechanism used in computer systems to control access to resources such as files, directories, or network shares.

Traditionally, access control in computer systems is based on a simple permissions model that allows or denies access to resources based on the user's identity or group membership. However, this approach can become too rigid and difficult to manage when dealing with complex scenarios that require finer-grained control over access.

ACLs provide a more flexible and granular approach to access control by allowing the specification of access permissions for individual users or groups on a per-resource basis. This means that different users or groups can be granted different levels of access to the same resource.

Some use cases where ACLs can be particularly useful include:

1. **File and folder permissions:** ACLs can be used to grant or restrict access to specific files or folders for individual users or groups.

2. **Network sharing:** ACLs can be used to control access to network shares, allowing or denying access to specific users or groups.

3. **Database access:** ACLs can be used to restrict access to specific tables or records within a database to ensure data security and privacy.

Overall, ACLs are a powerful tool for managing access to resources in a more flexible and granular way, and can be particularly useful in scenarios where traditional permissions models are too rigid or difficult to manage.

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

There can be several reasons why creating a file may fail. Here are three different reasons:

1. **Insufficient permissions:** If the user does not have the necessary permissions to create a file in the directory where they are trying to create the file, the creation process will fail. In such cases, the user may need to check the permissions of the directory or contact the system administrator for assistance.

2. **File already exists:** If a file with the same name already exists in the directory, the creation process will fail unless the user specifies a different name for the new file. In such cases, the user may need to check if the file already exists in the directory or choose a different name for the new file.

3. **Disk full or quota exceeded:** If the disk where the file is being created is full or if the user's quota has been exceeded, the creation process will fail. In such cases, the user may need to free up space on the disk or contact the system administrator to increase their quota limit.

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

The command 'chmod -x $(which chmod)' removes the execute permission from the 'chmod' command, which makes it impossible to modify the file permissions of any other file using the 'chmod' command.

To fix this, one approach is to use the full path of the 'chmod' command to restore its execute permission. Here are the steps to do so:

1. Open a terminal or command prompt window.
2. Log in as the root user or switch to the root user using the 'sudo su' command.
3. Run the following command to restore the execute permission of the 'chmod' command:

```
chmod +x /bin/chmod
```

This will restore the execute permission of the 'chmod' command and allow the user to modify the file permissions of other files using the 'chmod' command again.

SCENARIOS

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

There are several ways to copy a file to a remote Linux host, but one of the most common ways is to use the Secure Copy (SCP) command, which uses the SSH protocol to securely transfer files between hosts. Here's how you can use SCP to copy a file to a remote Linux host:

1. Open a terminal on your local machine.
2. Type the following command:

```
scp /path/to/local/file username@remote_host:/path/to/remote/directory
```

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

3. Press Enter and you will be prompted for the password for the remote host.
4. Once you enter the password, the file will be copied to the remote host.

21. How to generate a random string?

To generate a random string in a programming language, the approach can differ slightly depending on the language used. However, here is an example using Python's built-in 'secrets' module:

```
import string
import secrets
alphabet = string.ascii_letters + string.digits
random_string = ''.join(secrets.choice(alphabet) for i in range(10))
print(random_string)
```

This will print a random string of 10 characters, consisting of uppercase and lowercase letters, as well as digits.

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

To generate a random string of 7 characters in Python, you can use the 'string' and 'random' modules. Here is an example code:,

```
import string
import random
alphabet = string.ascii_letters + string.digits
random_string = ''.join(random.choice(alphabet) for i in range(7))
print(random_string)
```

This will print a random string of 7 characters, consisting of uppercase and lowercase letters, as well as digits.

SYSTEMD

23. What is systemd?

systemd is a system and service manager for Linux operating systems. It is designed to provide a more centralized and unified way of managing system services, including the initialization of the system, the management of user sessions, and the management of system daemons and services.

systemd replaces traditional init systems and provides a range of new features such as on-demand starting of daemons, parallel startup of services, resource management, logging, and error reporting. It is used in many popular Linux distributions, including Debian, Ubuntu, Fedora, and Red Hat Enterprise Linux.

Some of the benefits of systemd include improved boot speed and reliability, easier management of system services and resources, and better integration with modern Linux features like cgroups and namespaces. However, systemd has also been controversial in the Linux community due to its perceived complexity and the amount of control it gives to the system manager.

24. How to start or stop a service?

The method to start or stop a service can vary depending on the operating system and service management system being used. Here are some common methods:

1. **Using systemctl:** If you're using a Linux distribution that uses systemd as the service management system, you can use the 'systemctl' command to start or stop services. For example, to start the Apache web server, you can use the following command:

```
sudo systemctl start apache2
```

To stop the Apache server, you can use:

```
sudo systemctl stop apache2
```

2. **Using service:** The 'service' command is another common way to start or stop services on Linux. This works with different service management systems like systemd, SysVinit, and Upstart. For example, to start the Apache server using the 'service' command, you can use:

```
sudo service apache2 start
```

To stop the Apache server, you can use:

```
sudo service apache2 stop
```

3. **Using init.d scripts:** Some services may have their own init scripts located in the '/etc/init.d' directory. You can use these scripts to start or stop the service. For example, to start the SSH service, you can use the following command:

```
sudo /etc/init.d/ssh start
```

To stop the SSH service, you can use:

```
sudo /etc/init.d/ssh stop
```

25. How to check the status of a service?

To check the status of a service on Linux, you can use one of the following methods depending on the service management system being used:

1. **Using systemctl:** If you're using a Linux distribution that uses systemd as the service management system, you can use the 'systemctl' command to check the status of a service. For example, to check the status of the Apache web server, you can use the following command:

```
systemctl status apache2
```

This will display the current status of the Apache service, including whether it is running or stopped, and any errors or warnings.

2. **Using service:** If you're using a Linux distribution with SysVinit or Upstart as the service management system, you can use the 'service' command to check the status of a service. For example, to check the status of the SSH service, you can use the following command:

```
service ssh status
```

This will display the current status of the SSH service, including whether it is running or stopped, and any errors or warnings.

3. **Using init.d scripts:** If a service has its own init script located in the '/etc/init.d' directory, you can use the script to check the status of the service. For example, to check the status of the MySQL service, you can use the following command:

```
/etc/init.d/mysql status
```

This will display the current status of the MySQL service, including whether it is running or stopped, and any errors or warnings.

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

On a system which uses systemd, you can display the logs of a service or of the system itself using the 'journalctl' command. Here are some examples:

1. **Display all system logs:**

```
sudo journalctl
```

This will display all the system logs, starting from the oldest entry to the newest.

2. **Display logs for a specific service:**

```
sudo journalctl -u servicename.service
```

Replace 'servicename' with the name of the service you want to view the logs for. This will display all the logs related to that service.

3. **Display logs within a specific time range:**
    sudo journalctl --since "2022-01-01 00:00:00" --until "2022-02-01 00:00:00"
Replace the date and time values with the desired start and end times for the log range.

4. **Display logs with a specific priority level:**
    sudo journalctl -p err
Replace 'err' with the desired priority level, such as 'emerg', 'alert', 'crit', 'err', 'warning', 'notice', 'info', or 'debug'.

5. **Display logs in real-time:**
    sudo journalctl -f
This will display the logs in real-time, as new log entries are added.

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

To make a certain process/app a service on Linux, you need to create a service file in the appropriate directory, depending on the service management system being used. Here's a general overview of the steps involved:

1. **Determine the service management system:** Check which service management system your Linux distribution is using, as the process for creating a service may differ depending on the system. For example, systemd, SysVinit, or Upstart.

2. **Create the service file:** Create a service file with the appropriate configuration parameters for your process or application. The location and format of the service file depends on the service management system being used.

3. **Copy the service file to the appropriate directory:** Copy the service file to the appropriate directory, depending on the service management system being used. For example, '/etc/systemd/system' for systemd, '/etc/init.d/' for SysVinit, or '/etc/init/' for Upstart.

4. **Enable and start the service:** Enable the service to start automatically at boot time and start the service. This step also varies depending on the service management system being used.

Here's an example of creating a systemd service file for a Python script:

1. Create a new file named 'myapp.service' in the '/etc/systemd/system' directory with the following content:
    [Unit]
    Description=My App
    After=network.target
    [Service]
    User=myuser
    WorkingDirectory=/path/to/myapp
    ExecStart=/usr/bin/python3 /path/to/myapp/myapp.py
    Restart=always
    [Install]
    WantedBy=multi-user.target
2. Save the file and exit.
3. Reload the systemd daemon to make the system aware of the new service file:
    sudo systemctl daemon-reload
4. Enable the service to start automatically at boot time:
    sudo systemctl enable myapp.service
5. Start the service:
    sudo systemctl start myapp.service

28. Troubleshooting and Debugging

Troubleshooting and debugging are important skills for anyone working with technology, whether it's software, hardware, or networks. Troubleshooting involves identifying and diagnosing problems, while debugging involves finding and fixing errors in code.

Here are some tips for effective troubleshooting and debugging:

1**. Understand the problem:** Before you can begin to fix a problem, you need to understand what's causing it. Take the time to gather information about the problem, including any error messages, recent changes, and relevant system information.

2. **Reproduce the problem:** If you can reproduce the problem consistently, you'll have an easier time diagnosing and fixing it. Try to identify the steps that lead up to the problem and document them.

3. **Check for common causes:** Some problems have common causes that are easy to fix. Check for common issues like network connectivity problems, incorrect configuration settings, and software conflicts.

4. **Use diagnostic tools:** There are many diagnostic tools available for troubleshooting and debugging. Use tools like traceroute, ping, and network analyzers to help identify the source of a problem.

5. **Narrow down the problem:** Once you've identified the general area where the problem is occurring, try to narrow it down further. This can help you pinpoint the specific cause and develop a more effective solution.

6. **Test solutions:** Once you have a solution in mind, test it to see if it works. Test it in a controlled environment before implementing it in a live system.

7. Document the process: Document the troubleshooting and debugging process, including any solutions that worked and those that didn't. This can be helpful for future reference and for training others.

29. Where system logs are located?

The location of system logs varies depending on the operating system and configuration, but here are some common locations:

1. **Linux/Unix:** In Linux and Unix systems, logs are typically stored in the /var/log directory. Each application or system component usually has its own log file within this directory. For example, system messages are logged in the /var/log/messages file, while Apache logs are stored in /var/log/apache2 or /var/log/httpd.

2. **Windows:** In Windows, logs are stored in the Event Viewer. To access the Event Viewer, open the Control Panel and navigate to Administrative Tools > Event Viewer. System, application, and security logs are stored here.

3. **macOS:** In macOS, logs are stored in the /var/log directory, similar to Linux and Unix systems. The Console app can be used to view and filter system logs.

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

You can use the 'tail' command to follow a file's content as it's being appended without having to open the file each time. Here's how to do it:

1. Open a terminal window.
2. Type 'tail -f filename' and press Enter. Replace "filename" with the name of the file you want to follow.
3. The 'tail' command will start outputting the last 10 lines of the file, and will continue to output any new lines as they're appended to the file.
4. To stop following the file, press Ctrl + C in the terminal window.

The '-f' option tells 'tail' to "follow" the file, meaning that it will continue to output any new lines that are appended to the file. You can use this command to monitor log files or any other type of file that's being written to in real-time.

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

There are many tools that network administrators and engineers use for this purpose. Here are some common tools:

1. **Ping:** A command-line tool that tests network connectivity by sending an ICMP echo request to a target host and measuring the response time.

2. **Traceroute:** A command-line tool that shows the path packets take from your computer to a target host, including the number of hops and the time it takes for each hop.

3. **Wireshark:** A free and open-source packet analyzer that can be used to capture and analyze network traffic in real-time.

4. **Netstat:** A command-line tool that shows network connections, listening ports, and network statistics.

5. **Nmap:** A network exploration and security auditing tool that can be used to scan networks and identify hosts and services.

6. **tcpdump:** A command-line tool that captures and analyzes network traffic in real-time.

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

There are many tools that system administrators and engineers use for this purpose. Here are some common tools:

1**. fsck:** A command-line tool that checks and repairs file system errors on Unix and Linux systems.

2. **chkdsk:** A command-line tool that checks and repairs file system errors on Windows systems.

3. **Disk Utility:** A graphical tool that can be used to verify and repair disk and file system issues on macOS.

4. **SMART monitoring tools:** Many disk drives have built-in Self-Monitoring, Analysis, and Reporting Technology (SMART) that can be monitored and analyzed for issues. There are various tools available for monitoring and interpreting SMART data, such as smartmontools on Unix and Linux systems.

5. **TestDisk:** A free and open-source data recovery tool that can be used to recover lost partitions and files on various file systems.

6. **NTFS-3G:** A third-party driver for reading and writing to NTFS file systems on Unix and Linux systems.

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

There are many tools that system administrators and engineers use for this purpose. Here are some common tools:

1. **ps:** A command-line tool that shows information about running processes, such as their process ID (PID), CPU usage, and memory usage.

2. **top:** A command-line tool that provides real-time information about system processes and resource usage.

3. **lsof:** A command-line tool that lists open files and the processes that have them open.

4. **strace:** A command-line tool that traces system calls and signals made by a process, which can help identify issues with system calls or other low-level operations.

5. **gdb:** A debugger that can be used to analyze and debug running processes, including the ability to inspect memory and execute instructions in the process.

6. pstree: A command-line tool that displays the process hierarchy in a tree-like format, which can help identify parent-child relationships between processes.

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

There are many tools that system administrators and engineers use for this purpose. Here are some common tools:

1. **top:** A command-line tool that provides real-time information about system processes and their CPU usage.

2. **htop:** A more advanced version of top that provides additional information about system processes and their resource usage.

3. **mpstat:** A command-line tool that shows CPU usage statistics for each processor on the system.

4. **perf:** A profiling tool that can be used to analyze CPU and system performance in real-time.

5. **strace:** A command-line tool that traces system calls and signals made by a process, which can help identify issues with system calls or other low-level operations that are affecting CPU usage.

6. **ltrace:** A command-line tool that traces library calls made by a process, which can help identify issues with shared libraries that may be affecting CPU usage.

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

If someone calls me claiming that their system is slow, I would take the following steps to diagnose and resolve the issue:

1. **Gather more information:** I would ask the person questions to understand what they mean by "slow" and to gather more information about the system, such as its hardware specifications, operating system, and any recent changes or updates that may have been made.

2. **Check system resources:** I would check the system's CPU, memory, and disk usage to see if any of these resources are being heavily utilized, which could be causing the system to slow down.

3. **Check running processes:** I would use tools like ps, top, or htop to check for any processes that are using a lot of system resources or causing high CPU or memory usage.

4. **Check system logs:** I would check system logs for any errors or warnings that could indicate a problem with the system, such as hardware failures or software errors.

5. **Check network connectivity:** If the system is accessing remote resources or services, I would check the network connectivity to ensure that it is not a network issue causing the system to slow down.

6. **Perform hardware diagnostics:** If necessary, I would perform hardware diagnostics to check for any hardware issues that could be causing the system to slow down, such as failing hard drives or overheating processors.

7. **Optimize system settings:** If no hardware issues are found, I would review and optimize system settings, such as adjusting power management settings, disabling unnecessary startup programs, or adjusting system performance settings.

8. **Update or reinstall software:** If the system is running outdated or problematic software, I would recommend updating or reinstalling the software to resolve any issues.

36. Explain iostat output

iostat is a command-line tool that is used to monitor system input/output (I/O) performance. The iostat output provides detailed statistics on the utilization of system disks, CPU, and other I/O devices. Here is an example iostat output:

```
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
4.54    0.00    0.56    0.72    0.00   94.19

Device:         rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await w_await  svctm  %util
sda               0.10     2.16    1.10    0.80    26.36    28.71    73.53     0.03   12.33    7.60   17.85   3.68   0.52
sdb               0.13     2.16    1.10    0.80    26.57    28.71    73.53     0.03   12.33    7.60   17.85   3.68   0.52
```

The output is divided into two sections: "CPU" and "Device".

1. The CPU section provides information on the utilization of the system's CPU. The values are expressed as percentages of total CPU time and are broken down into user, system, and idle time, as well as any time spent waiting for I/O operations to complete.

2. The Device section provides information on the utilization of the system's devices, including disks. The values are broken down by device and include the following statistics:

- rrqm/s: the number of read requests merged per second
- wrqm/s: the number of write requests merged per second
- r/s: the number of read requests issued per second
- w/s: the number of write requests issued per second
- rkB/s: the number of kilobytes read per second
- wkB/s: the number of kilobytes written per second

- avgrq-sz: the average size (in sectors) of the requests that were issued to the device
- avgqu-sz: the average number of requests waiting for service
- await: the average time (in milliseconds) for I/O requests to be served
- r_await: the average time (in milliseconds) for read requests to be served
- w_await: the average time (in milliseconds) for write requests to be served
- svctm: the average service time (in milliseconds) for I/O requests
- %util: the percentage of time that the device was busy servicing I/O requests

These statistics can be used to diagnose performance issues related to disk and other I/O devices. For example, high values for avgqu-sz or await may indicate that the device is experiencing a high level of I/O queueing, while a high %util value may indicate that the device is being heavily utilized and may need to be upgraded or replaced.

37. How to debug binaries?

Debugging binaries can be done using a debugger tool, such as GDB (GNU Debugger) or LLDB. Here are the general steps to debug a binary using GDB:

1. Compile the binary with debug symbols. This can be done by adding the -g flag to the compiler command. Debug symbols provide additional information about the binary that can be used during the debugging process.

2. Start GDB by running the command gdb <binary>, where <binary> is the name of the binary that you want to debug.

3. Set breakpoints in the code using the break command. A breakpoint is a point in the code where execution will pause and allow you to examine the state of the program.

4. Run the binary using the run command. The binary will start executing, and GDB will pause execution when it reaches a breakpoint.

5. Examine the state of the program using various GDB commands. For example, the print command can be used to print the value of a variable, and the backtrace command can be used to display a stack trace.

6. Step through the code using the step and next commands. The step command will step into a function call, while the next command will step over a function call.

7. Continue execution using the continue command. Execution will continue until the next breakpoint or until the program exits.

8. When you have finished debugging, exit GDB using the quit command.

Debugging binaries can be a complex process, and it may take some practice to become comfortable with the various GDB commands and techniques. It is also important to keep in mind that debugging can introduce new issues or alter the behavior of the program, so it is important to make backups of important files and test thoroughly after making changes.

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

CPU load and CPU utilization are both metrics used to measure the performance of a CPU, but they represent slightly different things.

CPU load refers to the amount of work being performed by the CPU at a given moment, measured as the number of processes waiting in the system's run queue. CPU load is typically measured as a percentage of the total number of CPUs in the system, with a value of 100% indicating that all CPUs are fully loaded.

On the other hand, CPU utilization refers to the percentage of time that the CPU is actively performing work. This includes time spent executing user processes, as well as time spent performing system tasks such as handling interrupts and scheduling processes. A CPU utilization of 100% indicates that the CPU is fully utilized and is not available to perform additional work.

In summary, CPU load measures the number of processes waiting for CPU time, while CPU utilization measures the percentage of time that the CPU is actively performing work. Both metrics can be useful in understanding the performance of a system and diagnosing performance issues.

39. How you measure time execution of a program?

There are various ways to measure the time execution of a program, depending on the programming language and platform being used. Here are some general approaches:

1. **Using the time command:** In Unix/Linux systems, the time command can be used to measure the execution time of a program. Simply prefix the command with time to measure the real, user, and system time.

2. **Using a profiler:** A profiler is a tool that can be used to analyze the performance of a program and identify performance bottlenecks. Many programming languages come with built-in profilers, such as cProfile for Python, or you can use third-party profilers such as Valgrind.

3. **Using performance counters:** Modern CPUs often have performance counters that can be used to measure the performance of a program. These counters can be accessed using specialized libraries or tools.

4. **Using timing functions:** Many programming languages provide built-in functions or libraries for measuring time, such as the time() function in C or the timeit module in Python. These functions can be used to measure the time taken by specific portions of a program.

It's important to note that measuring the time execution of a program can be affected by various factors such as the system load, I/O operations, and the size of the input data. Therefore, it's recommended to run the program multiple times and take an average of the results to get a more accurate measurement.

SCENARIOS

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

To find the process writing to a file, you can use the 'lsof' command in Unix/Linux systems.

The 'lsof' command lists open files and the processes that opened them. You can use it to find the process ID (PID) of the process writing to the file by specifying the file path as an argument. Here's an example command:

```
lsof /path/to/file
```

This will list all the processes that have the file open, along with their PIDs and other information.

Once you have the PID of the process, you can use the 'kill' command to terminate it. Here's an example command:

```
kill PID
```

Replace 'PID' with the actual process ID you obtained from the 'lsof' command. This will send a SIGTERM signal to the process, which will give it a chance to clean up and exit gracefully. If the process doesn't exit within a reasonable amount of time, you can use the 'kill -9' command to send a SIGKILL signal, which will

forcefully terminate the process. However, it's recommended to use SIGTERM first and only resort to SIGKILL if necessary.

KERNEL

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

In computing, a kernel is a fundamental component of an operating system (OS) that serves as the bridge between the hardware and software components of a computer system. The kernel is responsible for managing system resources such as memory, input/output (I/O) devices, and processor time, and it provides an interface for applications to access these resources.

The kernel is the core of the operating system and is loaded into memory when the computer boots up. It remains in memory throughout the operation of the computer, providing services to other parts of the operating system and applications.

The kernel performs many tasks, including:

1. **Memory management:** Allocating and managing memory for applications and the operating system.

2. **Process management:** Scheduling and managing the execution of multiple processes or programs.

3. **Device management:** Controlling and managing input/output devices such as keyboards, printers, and hard drives.

4. **Security management:** Enforcing security policies and providing protection against unauthorized access.

5**. File system management:** Managing the storage and retrieval of data on storage devices.

Overall, the kernel is a crucial component of an operating system, providing the foundation for all other software to run on top of it.

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

To find out which kernel version your system is using, you can use the command-line interface on your operating system.
Here are some common commands to check the kernel version:

On Linux systems:

1. Open a terminal and type the following command:
   uname -r
This will display the kernel version currently running on your system.

2. Alternatively, you can also use the following command to get more detailed information about the kernel:
   uname -a
This will display the kernel version along with other system information such as the operating system name, machine architecture, and hostname.

On Windows systems:

1. Press the Windows key + R to open the Run dialog box.
2. Type in "cmd" and press Enter to open the command prompt.

3. Type the following command:

```
systeminfo
```

This will display detailed system information, including the kernel version under the "System type" section.

Alternatively, you can also use the following command to display the kernel version only:

```
ver
```

This will display the version of the Windows kernel currently running on your system.

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

In Linux, a kernel module is a piece of code that can be dynamically loaded into or removed from the kernel at runtime. These modules provide additional functionality to the kernel without having to rebuild the entire kernel or reboot the system.

A kernel module can be used to add support for a new device, add a new filesystem, or extend the functionality of an existing driver or subsystem. Kernel modules are typically distributed as object files with a ".ko" file extension.

To load a new module in Linux, follow these steps:

1. Open a terminal and become the root user by typing the command:

```
sudo su
```

2. Use the 'modprobe' command to load the module. For example, to load the "pcspkr" module that provides support for the PC speaker, type:

```
modprobe pcspkr
```

3. To verify that the module was loaded successfully, use the 'lsmod' command to list all loaded modules:

```
lsmod
```

This will display a list of all currently loaded kernel modules.

4. To remove a module, use the 'rmmod' command followed by the name of the module. For example, to remove the "pcspkr" module, type:

```
rmmod pcspkr
```

44. Explain user space vs. kernel space

In computing, user space and kernel space refer to different areas of memory and functionality in an operating system.

User space is the portion of memory where user applications and programs run. It contains user-level processes, such as text editors, web browsers, and other software applications that users interact with directly. User space is a protected area of memory and applications in this space are isolated from other user applications and from the kernel space.

Kernel space, on the other hand, is the protected area of memory where the kernel and its drivers run. The kernel is the core component of an operating system that manages system resources such as memory, processor time, and input/output devices. Drivers are specialized programs that communicate directly with hardware devices, such as printers, network cards, and storage devices.

Kernel space has unrestricted access to all of the hardware resources and memory of the system. Because of this, programs running in kernel space have complete control over the system and can execute privileged operations that are not available to user-level programs in user space.

The separation between user space and kernel space is necessary for security and stability reasons. User space applications are limited in their access to system resources, preventing them from interfering with other programs or accidentally damaging the system. On the other hand, the kernel and its drivers have

full control over the system, allowing them to perform critical system-level functions without interference from user space programs.

In summary, user space and kernel space are two distinct areas of memory and functionality in an operating system. User space contains user-level processes and applications, while kernel space contains the kernel and its drivers, which have unrestricted access to system resources.

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

In the Linux kernel lifecycle, there are several phases in which you can change its configuration.

1. **During compilation:** The kernel configuration options can be set during the compilation phase, which generates the final kernel image. This process involves selecting which features and drivers to include in the kernel based on the specific requirements of the system. The configuration options can be accessed through the make menuconfig or make xconfig commands.

2. **During boot:** The kernel configuration can also be changed during the boot process by passing kernel boot parameters. These parameters can be used to enable or disable specific features, such as debugging or hardware support. Boot parameters can be set by modifying the bootloader configuration or by passing them directly to the kernel during boot.

3. **At runtime:** The Linux kernel allows for runtime configuration changes through the use of system control (sysctl) settings. Sysctl is a mechanism for changing kernel parameters at runtime, such as setting the maximum number of open files or adjusting network settings. The sysctl command can be used to view and change these parameters.

4. **With kernel modules:** Some kernel configuration options can be changed dynamically by loading or unloading kernel modules. For example, a module may add support for a new hardware device or enable a new feature. Module configuration options can be set using the modprobe command or by editing the module configuration file in the /etc/modprobe.d/ directory.

In summary, the Linux kernel configuration can be modified during compilation, during boot, at runtime through sysctl settings, and with kernel modules. It is important to note that some configuration options require a kernel rebuild, while others can be changed dynamically at runtime.

46. Where can you find kernel's configuration?

The kernel configuration is stored in the '.config' file in the root of the kernel source tree. This file contains all of the configuration options selected during the kernel compilation process.

To view the kernel configuration, you can navigate to the root of the kernel source tree and use a text editor or the 'cat' command to open the '.config file'. For example, you can run the following command in the terminal to view the kernel configuration:

cat /usr/src/linux/.config

Alternatively, you can use the make 'menuconfig' or 'make xconfig' commands to view and modify the kernel configuration options. These commands provide a graphical user interface that allows you to select which features and drivers to include in the kernel.

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

The file that contains the command passed to the boot loader to run the kernel is the bootloader configuration file. The exact location of this file depends on the bootloader used by the system.

For example, if you are using the GRUB bootloader, the configuration file is typically located at /boot/grub/grub.cfg or /boot/grub2/grub.cfg on most Linux distributions. However, it is not recommended to

edit this file directly as it is automatically generated by the system and any changes made to it may be overwritten.

Instead, the recommended way to modify the bootloader configuration is to create a new configuration file in the /etc/grub.d/ directory and add the necessary configuration options. You can then run the update-grub command to regenerate the bootloader configuration file based on the new configuration.

Other bootloaders, such as LILO, SYSLINUX, or UEFI-based bootloaders, have their own configuration files with different names and locations. It is recommended to consult the documentation for the specific bootloader you are using to determine the location of the configuration file.

48. How to list kernel's runtime parameters?

You can list the Linux kernel's runtime parameters using the 'sysctl' command. The 'sysctl' command is used to view and modify kernel parameters at runtime.

To list all available kernel parameters, you can run the following command in the terminal:

```
sysctl -a
```

This will display a list of all kernel parameters with their current values. You can also use the 'sysctl' command to view a specific kernel parameter by specifying its name. For example, to view the maximum number of open files allowed by the system, you can run the following command:

```
sysctl fs.file-max
```

This will display the current value of the 'fs.file-max' parameter.

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

Yes, running 'sysctl' -a as a regular user versus as root will produce different results.

Many of the kernel parameters that can be viewed and modified using the sysctl command require elevated privileges to access. By default, regular users do not have the necessary permissions to view or modify these parameters.

When running sysctl -a as a regular user, you will only see the kernel parameters that are accessible to that user. These parameters typically include a subset of the overall parameters available on the system, such as networking or virtual memory parameters.

On the other hand, running sysctl -a as root will display all available kernel parameters, including those that are restricted to elevated privileges. This allows root users to view and modify all kernel parameters on the system.

It is important to note that modifying kernel parameters can have significant effects on system performance and stability, and should only be done with caution and with a thorough understanding of the implications of any changes made.

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

To enable IPv4 forwarding in the Linux kernel, you can modify the value of the 'net.ipv4.ip_forward' parameter using the 'sysctl' command.

By default, IPv4 forwarding is usually disabled in the kernel. To enable it, you can run the following command as root:

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

This command sets the value of the 'net.ipv4.ip_forward' parameter to '1', which enables IPv4 forwarding.

To make this change persistent across reboots, you can add the following line to the '/etc/sysctl.conf' file:

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

This will cause the 'sysctl' command to automatically set the value of the 'net.ipv4.ip_forward' parameter to '1' on system startup.

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

When you run the sysctl command to change a kernel runtime parameter, sysctl communicates with the kernel via the sysctl() system call to update the corresponding kernel parameter value in the /proc/sys virtual file system.

When a program or user space process reads a value from a file in the /proc/sys directory, the kernel returns the current value of the corresponding kernel parameter.

Therefore, any changes made to the kernel parameter through the sysctl command are immediately reflected in the /proc/sys virtual file system, and any program or user space process that reads the value of the corresponding kernel parameter will receive the updated value.

This allows system administrators to make runtime changes to kernel parameters without having to reboot the system or rebuild the kernel.

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

Changes to kernel runtime parameters made via the 'sysctl' command are not persistent by default, meaning they will be lost after a system reboot.

To make changes to kernel runtime parameters persist across reboots, you need to save them to a configuration file. On most Linux systems, this file is usually located in the '/etc/sysctl.conf' file.

To make a change persistent, you can add a line to the /etc/sysctl.conf file in the following format:

<parameter name>=<value>

For example, to set the maximum number of open files allowed for all users to 100000, you can add the following line to the '/etc/sysctl.conf' file:

fs.file-max=100000

Once you have made changes to the '/etc/sysctl.conf' file, you can either reboot the system or run the following command to apply the changes immediately without rebooting:

sudo sysctl -p

This command reloads the kernel parameters from the /etc/sysctl.conf file and applies them to the system's current kernel.

After the changes have been made and applied, they will persist across reboots and be automatically loaded by the kernel at system startup.

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?

By default, changes made to kernel parameters within a container will not affect the kernel parameters of the host system on which the container runs. This is because containers use namespaces to provide a separate and isolated environment for running applications and services.

When a container is started, it creates a new namespace for the process running inside the container. This namespace includes a separate view of the kernel and its runtime parameters. Changes made to kernel parameters inside the container only affect the container's own view of the kernel and do not affect the host system or other containers running on the same host.

However, some kernel parameters can be shared between the host and the container by using the --privileged flag when starting the container. When a container is started with this flag, it has access to all the kernel parameters of the host system, and any changes made inside the container will affect the host's kernel parameters as well.

It is generally not recommended to use the --privileged flag as it can introduce security risks and potentially cause issues with other containers running on the same host. Instead, it is recommended to manage kernel parameters separately for each container and to avoid relying on privileged access to the host's kernel parameters.

SSH

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

SSH (Secure Shell) is a cryptographic network protocol used to establish a secure connection between two computers over an unsecured network. It allows users to securely access and control a remote system, as well as transfer files securely.

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

```
systemctl status sshd
```

This will display the status of the SSH service on the system. If the service is running, you will see a message like this:

```
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: active (running) since Thu 2022-03-10 14:53:24 PST; 3 weeks 0 days ago
```

If the service is not running, you will see a message like this:

```
● sshd.service - OpenSSH server daemon
  Loaded: loaded (/lib/systemd/system/sshd.service; enabled; vendor preset: enabled)
  Active: inactive (dead)
```

In this case, you can start the SSH service using the following command:

```
sudo systemctl start sshd
```

Or, if the SSH service is not installed, you can install it using the following command:

```
sudo apt-get install openssh-server
```

This command will install the OpenSSH server package on your system.

55. Why SSH is considered better than telnet?

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

1. **Security:** SSH encrypts all data transmitted between the client and the server, providing a secure connection. Telnet, on the other hand, transmits data in plain text, making it vulnerable to interception and snooping.

2. **Authentication:** SSH provides strong user authentication using a variety of methods, such as passwords, public-key authentication, and multi-factor authentication. Telnet, on the other hand, does not have any built-in authentication mechanisms and relies on the security of the underlying network.

3. **Portability:** SSH is available on a wide range of operating systems and platforms, including Windows, Linux, macOS, and mobile devices. Telnet, on the other hand, is not as widely supported and is often not installed by default on modern systems.

4. **Functionality:** SSH provides many advanced features, such as file transfer, X11 forwarding, and remote command execution. Telnet, on the other hand, only provides basic terminal access.

5. **Better performance:** SSH has better performance compared to Telnet as it has better compression capabilities.

Overall, SSH is considered a more secure and versatile alternative to Telnet and is the recommended choice for remote shell access and file transfer over unsecured networks.

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

The ~/.ssh/known_hosts file is a plain text file that contains a list of known hosts that the user has connected to using SSH (Secure Shell). When a user connects to a remote server for the first time using SSH, the server's host key fingerprint is added to this file.

The host key fingerprint is a cryptographic hash of the server's public key, and it is used to verify the authenticity of the remote server during subsequent connections. When the user connects to the same remote server again, SSH compares the server's host key fingerprint to the one stored in the known_hosts file. If they match, the user can be sure that they are connecting to the same server they connected to before.

If the host key fingerprint of the remote server has changed since the last time the user connected, SSH will issue a warning and ask the user to confirm whether they want to connect to the server. This is a security feature designed to prevent man-in-the-middle attacks, where an attacker intercepts the user's SSH connection and pretends to be the remote server.

The known_hosts file can be edited manually to add or remove host key fingerprints, but it is generally not recommended to do so unless you know what you are doing. Incorrectly modifying this file can cause SSH connections to fail or leave the user vulnerable to man-in-the-middle attacks.

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

"Host key verification failed" is an error message that occurs when the host key fingerprint of a remote server does not match the one stored in the ~/.ssh/known_hosts file on the local machine. This error usually occurs when the user tries to connect to a remote server for the first time, or when the server's host key has changed since the last time the user connected.

When a user connects to a remote server using SSH, the server's host key fingerprint is added to the known_hosts file on the local machine. This is done to verify the authenticity of the server during subsequent connections. If the host key fingerprint of the remote server has changed since the last time the user connected, SSH will issue a warning and refuse to connect to the server, as it may indicate a potential man-in-the-middle attack.

To resolve the "Host key verification failed" error, the user should first verify that they are connecting to the correct remote server. If they are certain that they are connecting to the correct server, they can remove the old host key fingerprint from the known_hosts file using a text editor or the ssh-keygen command. Alternatively, they can use the -R option with the ssh command to remove the offending key automatically.

For example, to remove a host key for a server with IP address 192.0.2.1, you can use the following command:

```
ssh-keygen -R 192.0.2.1
```

After removing the old host key fingerprint, the user can try to connect to the remote server again. SSH will prompt the user to add the new host key fingerprint to the known_hosts file, and they can confirm the addition if they trust the remote server.

58. What is the difference between SSH and SSL?

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

SSH is primarily used for secure remote access to servers and devices, enabling users to securely log in and execute commands remotely. SSH provides strong authentication and encryption, making it resistant to eavesdropping, interception, and tampering. SSH operates at the application layer of the network stack, using the TCP (Transmission Control Protocol) as the underlying transport protocol.

SSL, on the other hand, is primarily used for secure communication between clients and servers over the Internet, such as web browsing, email, and file transfers. SSL provides encryption and authentication of

data transmitted between the client and server, making it resistant to eavesdropping and tampering. SSL operates at the transport layer of the network stack, using TCP or UDP (User Datagram Protocol) as the underlying transport protocol.

Both SSH and SSL use public-key cryptography to establish secure connections between parties. However, SSH uses its own set of keys and algorithms, while SSL uses X.509 digital certificates issued by trusted certificate authorities.

Overall, SSH and SSL are both important security protocols used to protect network communication, but they are designed for different use cases and operate at different layers of the network stack.

59. What ssh-keygen is used for?

'ssh-keygen' is a command-line utility used to generate, manage, and convert cryptographic keys for use with the SSH (Secure Shell) protocol. The ssh-keygen tool is included with most SSH installations and is typically located in the /usr/bin directory.

The main purpose of ssh-keygen is to generate a pair of cryptographic keys, which consists of a private key and a public key. The private key is kept secret and is used to authenticate the user to a remote server or device, while the public key is shared with the remote server or device and is used to verify the user's identity.

The ssh-keygen tool can also be used to convert keys between different formats, change the passphrase on a private key, and revoke or regenerate keys if they have been compromised. Additionally, ssh-keygen can be used to create a key fingerprint, which is a unique identifier of a key that can be used to verify its authenticity.

Overall, ssh-keygen is a powerful tool for managing SSH keys, and it is an essential component of any secure SSH setup. It is recommended that users take care when using ssh-keygen and follow best practices for key management, such as using strong passphrases and storing private keys securely.

60. What is SSH port forwarding?

SSH port forwarding (also known as SSH tunneling) is a technique used to securely transfer data between two computers over an untrusted network, such as the Internet. The technique involves creating a secure SSH connection between the two computers and forwarding traffic from one computer's ports to the other.

There are three main types of SSH port forwarding:

1. **Local port forwarding:** This is used to forward traffic from a local port on the client computer to a remote port on the server computer. The server then sends the traffic to its intended destination.

2. **Remote port forwarding:** This is used to forward traffic from a remote port on the server computer to a local port on the client computer. The client then sends the traffic to its intended destination.

3. **Dynamic port forwarding:** This is used to create a dynamic SOCKS proxy on the client computer that can be used to route traffic to any destination on the server network.

SSH port forwarding is commonly used to bypass firewalls or access resources on a remote network that would otherwise be unavailable. For example, a user may use SSH port forwarding to securely access a database server on a remote network, or to access a web server that is only available on a local network.

SSH port forwarding is a powerful tool for secure network communication, but it requires careful configuration and management to ensure that it is used securely and efficiently. It is recommended that users follow best practices for SSH port forwarding, such as restricting access to authorized users, using strong authentication and encryption, and monitoring network traffic for suspicious activity.