# Assignment-2
# for Linux

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

   / -> root directory.
   It is the top-level directory of the file system.

2. **What is stored in each of the following paths? /bin, /sbin, /usr/bin and /usr/sbin /etc /home /var /tmp**

   **/bin:** This directory contains executable files and essential command-line utilities that are required by the system administrator and users to perform basic tasks.
   Ex: ls, cp, rm, mv, and more.

   **/sbin:** This directory contains system binaries (i.e., programs) that are essential for the system administrator to manage the system, such as mount, umount, reboot, and others.

   **/usr/bin:** This directory contains non-essential command-line utilities that are used by regular users, such as text editors, web browsers, and other applications that are installed by the user or system administrator.

   **/usr/sbin:** This directory contains non-essential system binaries that are used by the system administrator to perform specialized system management tasks, such as network daemons, system backup utilities, and others.

   **/etc:** This directory contains system configuration files that are used by the operating system and various applications running on the system.

   **/home:** This directory contains the home directories of regular users, where they can store their personal files.

   **/var:** This directory contains variable data files, such as logs, spool files, and other files that may change frequently during system operation.

**/tmp:** This directory contains temporary files that are created by the system and various applications running on the system. These files are typically deleted automatically when the system is rebooted or when they are no longer needed.

3. **What is special about the /tmp directory when compared to other directories?**
   /tmp is meant as fast (possibly small) storage with a short lifetime. Many systems clean /tmp very fast - on some systems it is even mounted as RAM-disk. /var/tmp is normally located on a physical disk, is larger and can hold temporary files for a longer time.

4. **What kind of information one can find in /proc?**
   **/proc** is very special in that it is also a virtual filesystem. It's sometimes referred to as a process information pseudo-file system. It doesn't contain 'real' files but runtime system information (e.g. system memory, devices mounted, hardware configuration, etc).

5. **What makes /proc different from other filesystems?**
   The /proc filesystem does not store files on disk, but instead provides a dynamic interface for accessing information about the system and its processes in real-time. When a process or system component needs information about the system, it can access it directly from the /proc filesystem.

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

7. **What can be found in /proc/cmdline?**
   This file shows the parameters passed to the kernel at the time it is started.

8. **In which path can wefind the system devices (e.g. block storage)?**
   The /dev directory.

## Permissions

9. **How to change the permissions of a file?**
   Using chmod command:

Syntax of chmod:        chmod permissions filename

Where,

**permissions** can be read, write, execute or a combination of them.

**filename** is the name of the file for which the permissions need to change.

This parameter can also be a list if files to change permissions in bulk.

**Change permissions for files:**

Read (r), Write (w) and Execute (x) modes.

**Change Permissions using Symbolic Mode**

**u ->** user/owner

**g ->** group

**o ->** other

**+ ->** Adds a permission to a file or directory

**-   ->** Removes the permission

**= ->** Sets the permission if not present before. Also overrides the permissions if set earlier.

**10. What does the following permissions mean?**

**777, 644, 750**

**777:** This file has full read, write, and execute permissions for the owner, group, and other users.

**644:** This file has read and write permissions for the owner, and read-only permissions for the group and other users.

**750:** This file has read, write, and execute permissions for the owner, and read and execute permissions for the group.

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

Linux provides the **chmod** command which is used to change file and folder permission.

The **+x** parameter is used to add the x permission which is the symbol for the **execute** permission for the owner, group or others.

**12. Explain what is setgid and setuid**

**Setgid:** a bit that makes an executable run with the privileges of the group of the file.

**Setuid:** a bit that makes an executable run with the privileges of the owner of the file.

**13. What is the purpose of sticky bit?**

**Sticky bit:** a bit set on directories that allows only the owner or root can delete files and subdirectories.

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

**chmod**: Change file access permissions.

**Description:** This command is used to change the file permissions. These permissions read, write and execute permission for the owner, group, and others.

**Syntax (symbolic mode)**: chmod [ugoa][[+-=][mode]] file

The first optional parameter indicates who – this can be (u)ser, (g)roup, (o)there, or (a)ll.

The second optional parameter indicates opcode – this can be for adding (+), removing (-), or assigning (=) permission.

The third optional parameter indicates the mode – this can be (r)ead, (w)rite, or e(x)ecute.

**chown**: Change ownership of the file.

**Description**: Only the owner of the file has the right to change the file ownership.

**Syntax**: chown [owner] [file]

**chgrp**: Change the group ownership of the file

**Description**: Only the owner of the file has the right to change the file ownership

**Syntax**: chgrp [group] [file]

**15. What is sudo? How do weset it up?**

Sudo stands for either "substitute user do" or "super user do" and it allows weto temporarily elevate your current user account to have root privileges.

**Steps to Add Sudo User on Ubuntu**

Step 1: Create New User. Log into the system with a root user or an account with sudo privileges.

Step 2: Add User to Sudo Group. Most Linux systems, including Ubuntu, have a user group for sudo users.

Step 3: Verify User Belongs to Sudo Group.

Step 4: Verify Sudo Access.

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

True.

**17. Explain what are ACLs. For what use cases would werecommend to use them?**

An access control list (**ACL**) is a list of rules that specifies which users or systems are granted or denied access to a particular object or system resource.

Collaborative work, Compliance, File servers etc.

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

Insufficient permissions

Disk space limitations

Filesystem errors

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

To fix this issue, follow these steps:

1. Reboot your system into single-user mode.
2. Choose the option "root Drop to root shell prompt" to obtain a root shell.
3. Remount the root filesystem in read/write mode by executing the following command:
   ```
   mount -o remount,rw /
   ```
4. Give back the executable permission to the ‘chmod’ command by executing the following command:
   ```
   chmod +x $(which chmod)
   ```

5. Reboot your system by executing the command:

   reboot

After the system reboots, weshould be able to use the ‘chmod’ command again with the correct permissions.

## 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 from an Ubuntu machine, we can use the scp command, which stands for secure copy. The scp command uses SSH to securely copy files between hosts.

Ex: scp example.txt remote_username@remote_host_ip:/path/to/destination/

Replace example.txt with the name of the file we want to copy, remote_username with the username we use to log in to the remote host, remote_host_ip with the IP address of the remote host, and /path/to/destination/ with the path to the destination directory on the remote host where we want to copy the file.

We will be prompted to enter the password for the remote host account, after which the file will be copied over to the remote host.

**21. How to generate a random string?**

To generate a random string in Ubuntu, you can use the ‘openssl’ command. The ‘openssl’ command can generate random data, which can be used to create a random string.

Ex: openssl rand -base64 15

This command generates 15 random bytes and encodes them in base64, resulting in a 20-character string. You can adjust the number of bytes to generate by changing the number after ‘-base64’.

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

```
cat /dev/urandom | tr -dc 'a-zA-Z0-9' | fold -w 7 | head -n 1
```

This command uses the urandom device file, which provides an interface to the kernel's random number generator, to generate a stream of random bytes. The tr command is used to filter out all characters except letters (both upper and lowercase) and digits. The fold command is then used to split the stream into lines of 7 characters, and the head command selects the first line, which represents the random string of 7 characters.

## Systemd

**23. What is systemd?**

Systemd is a system and service manager for Linux, compatible with SysV and LSB init scripts. Systemd provides: Aggressive parallelization capabilities.

**24. How to start or stop a service?**

To start a service:

```
sudo systemctl stop ufw
```

To stop a service:

```
sudo systemctl start ufw
```

**25. How to check the status of a service?**

To check a service's status, use the systemctl status service-name command.

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

journalctl is a command for viewing logs collected by systemd. The systemd-journald service is responsible for systemd's log collection, and it retrieves messages from the kernel, systemd services, and other sources.

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

1. cd /etc/systemd/system.
2. Create a file named your-service.service.
3. Reload the service files to include the new service.
   sudo systemctl daemon-reload
4. Start your service.
   sudo systemctl start your-service.service
5. To check the status of your service.
   sudo systemctl status example.service

6. To enable your service on every reboot.
   sudo systemctl enable example.service
7. To disable your service on every reboot.
   sudo systemctl disable example.service

**28. Troubleshooting and Debugging**

1. Check the systemd journal: can view the journal using the journalctl command, which allows you to search, filter, and analyze log data.

2. Enable verbose logging: If you are trying to diagnose a specific problem, you can enable verbose logging for a specific unit using the systemctl set-property command.
   Ex: sudo systemctl set-property sshd.service LogLevel=debug

3. Check unit status and logs: You can check the status of a specific unit using the systemctl status command.
   Ex: systemctl status nginx.service

4. Debugging failed units: If a unit fails to start, you can use the systemctl start command with the --debug option to run the unit in debug mode.
   Ex: sudo systemctl start --debug apache2.service

5. Analyzing dependencies: You can use the systemd-analyze command to analyze the startup time and dependencies of units.

6. Use the systemd-cgtop command: The systemd-cgtop command provides a real-time view of system resource usage by cgroups.

**29. Where system logs are located?**

In systemd, system logs are stored in the journal, which is a centralized logging system that provides a unified and structured view of system events and messages. The journal is stored in binary format in the directory /var/log/journal, and can be accessed and managed using the **journalctl** command.

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

    In systemd, you can use the journalctl command with the --follow or -f option to follow the contents of a file as it is being appended. This is useful for monitoring log files and other continuously updated files without having to open and close the file every time.

    **journalctl -f -o cat -n 0 -f /var/log/myapp.log**

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

    **systemctl command**: You can use systemctl to check the status of network services and restart them if necessary.

    Ex: sudo systemctl restart networking

    **journalctl command**: You can use journalctl to view logs related to network services and identify any errors or warnings.

    Ex: sudo journalctl -u networking

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

    **systemctl command**: You can use systemctl to check the status of file system-related services, such as systemd-fsck, systemd-journald, and systemd-udevd.

    Ex: sudo systemctl status systemd-fsck

    **journalctl command**: You can use journalctl to view logs related to disk and file system events, such as disk errors, file system mounts and unmounts, and file system errors.

    Ex: sudo journalctl -u systemd-fsck -u systemd-remount-fs

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

    **systemctl command**: You can use systemctl to check the status of services and processes, and to start or stop them as needed.

    Ex: sudo systemctl status apache2

    **journalctl command**: You can use journalctl to view logs related to process events, such as process start and stop events, process crashes, and process errors.

    Ex: sudo journalctl -u apache2

**kill command**: You can use kill to send signals to processes, such as the SIGTERM signal to gracefully terminate a process, or the SIGKILL signal to forcefully terminate a process.
Ex: kill -TERM 1234

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

**top command**: You can use top to display real-time information about CPU usage and other system resources, such as memory usage and process information.
Ex: top

**htop command:** You can use htop as an alternative to top with more advanced features and a more user-friendly interface.
Ex: htop

**systemd-cgtop command**: You can use systemd-cgtop to display information about the system control groups (cgroups) and their resource usage, such as CPU usage and memory usage.
Ex: sudo systemd-cgtop

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

1. **Check the system load:** The system load is a measure of how many processes are running or waiting to run. You can check the system load using the uptime command, which will display the system load average for the last 1, 5, and 15 minutes. For example, you can run:
   Ex: uptime

2. **Check the system resources**: Check the system resources such as CPU, memory, and disk usage using tools like top, htop, free, or df. You can identify any resource bottlenecks which might be causing the slow performance.

3. **Check for runaway processes:** Runaway processes can consume excessive resources and slow down the system. You can use tools like ps, top, or htop to identify any processes that are consuming an unusually high amount of resources.

You can then investigate further and determine if the process should be terminated or if there is an underlying issue that needs to be addressed.

4. **Check system logs:** Use the journalctl command to check system logs for any errors or warnings that might indicate a problem. You can use filters to search for specific log messages related to system performance. For example, to search for log messages related to CPU usage, you can run:
   Ex: sudo journalctl _SYSTEMD_UNIT=systemd-cpu-load.service

5. **Check network performance**: Slow performance could also be related to network issues. You can use tools like ping or traceroute to check network connectivity and identify any network latency issues.

**36. Explain iostat output**

iostat is a tool that is used to monitor system input/output (I/O) statistics for devices and partitions. It is useful for identifying I/O performance issues and bottlenecks. Here is an example of iostat output:

```
Device    tps   kB_read/s   kB_wrtn/s   kB_read   kB_wrtn
sda       0.00   0.00        0.00        0         0
sdb       0.00   0.00        0.00        0         0
sdc       0.01   0.14        0.00        566       0
sdd       0.00   0.00        0.00        0         0
```

**Device**: the name of the device or partition being monitored.
**tps**: the number of transfers per second between the device and the system.
**kB_read/s**: the number of kilobytes read per second from the device.
**kB_wrtn/s**: the number of kilobytes written per second to the device.
**kB_read:** the total number of kilobytes read from the device since the system was started.
**kB_wrtn**: the total number of kilobytes written to the device since the system was started.

**37. How to debug binaries?**

**systemd-cgls**: This tool is used to display the control group (cgroup) hierarchy of the systemd system.

**systemd-analyze**: This tool is used to analyze and troubleshoot system boot performance.

**systemd-run:** This tool is used to execute a command in a transient unit. It can be used to test how a binary runs in a specific environment or with specific parameters.

**systemd-nspawn:** This tool is used to spawn a new namespace container for debugging purposes.

**systemd-cat**: This tool is used to read and write messages to the journal. It can be used to debug binaries that write messages to the journal.

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

| **CPU load** | **CPU utilization** |
|---|---|
| CPU load is a measure of the number of processes that are waiting to be executed by the CPU at a given time. It is typically measured as an average over a specific time period, such as 1, 5, or 15 minutes. | CPU utilization, on the other hand, is a measure of the percentage of time that the CPU is actively executing instructions. It is typically measured as an average over a specific time period, such as 1, 5, or 15 minutes. |
| The CPU load is calculated based on the number of processes waiting in the system's run queue. A high CPU load indicates that the CPU is busy and may be struggling to keep up with the demand from running processes. | CPU utilization takes into account the total amount of time that the CPU spends executing instructions, regardless of whether there are processes waiting in the run queue or not. A high CPU utilization indicates that the CPU is working hard and may be approaching its maximum capacity. |
| CPU load measures the number of processes waiting to be executed by the CPU. | CPU utilization measures the percentage of time that the CPU is actively executing instructions. |

39. **How you measure time execution of a program?**

    **time:** This is a built-in command in most Linux distributions, including systemd-based systems.

    **systemd-run:** This command can be used to run a program in a transient systemd unit, which allows you to measure its execution time using systemd's built-in logging and monitoring tools.

    **perf**: This is a powerful performance monitoring tool that is built into most Linux kernels, including those used in systemd-based systems. It can be used to measure the execution time of a program, as well as other performance metrics such as CPU utilization, cache misses, and more.

## 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?**

    **The file is actively being written to**: If the file is actively being written to, you can use the lsof command to list all processes that have the file open. For example, to find all processes that have the file /path/to/file open, you can run lsof /path/to/file.

    **The file is not actively being written to:** If the file is not actively being written to, you can use the fuser command to find the PID of the process that has the file open. For example, to find the PID of the process that has the file /path/to/file open, you can run fuser /path/to/file.

    **Multiple processes are writing to the file:** If multiple processes are writing to the file, you can use the lsof command to identify all processes that have the file open, as described in scenario 1. Once you have identified all processes that are writing to the file, you can use the kill command to terminate them one by one, or use a script to automate the process.

## Kernel

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

The kernel is a core component of an operating system and serves as the main interface between the computer’s physical hardware and the processes running on it. The kernel enables multiple applications to share hardware resources by providing access to CPU, memory, disk I/O, and networking.

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

Type the following cat command to print out information about the running Linux kernel:

```
$ cat /proc/version
```

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

The Linux kernel is modular, which means it can extend its capabilities through the use of dynamically-loaded kernel modules. A kernel module can provide:

1. a device driver which adds support for new hardware; or,
2. support for a file system such as btrfs or NFS.

To load a kernel module, run modprobe module_name as root. For example, to load the wacom module, run:

```
~]# modprobe Wacom
```

**44. Explain user space vs. kernel space**

| **Kernel Space** | **User Space** |
|---|---|
| Kernels and OS core execute here. | Normal program and applications softwares run here. |
| Its the core space of OS. | It's a form of sand-boxing that restricts user processes to access OS kernel. |
| It has full access to all memory and machine hardware. | It has limited access to memory and access kernel through system calls only. |
| It contains the page table for process, kernel data structure, threads, and kernel code etc. | It contains the program code, data, stacks, and heap of the process. |

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?**

We can list the kernel's runtime parameters on a Linux system by viewing the contents of the /proc/cmdline file. This file contains the command line arguments that were passed to the kernel at boot time.

To view the contents of the /proc/cmdline file, open a terminal window and type the following command:

```
cat /proc/cmdline
```

This will display a list of kernel parameters, separated by spaces. Each parameter is typically a key-value pair, separated by an equals sign (=).

**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?**

Use command sysctl -a|grep net.ipv4.ip_forward to check the IP forwarding status.
If net.ipv4.ip_forward=1, the ip forwarding is enabled.
If net.ipv4.ip_forward=0, follow the steps below to enable it.

1. Open /usr/lib/sysctl.d/50-default.conf with a supported editor.
2. Check if there is a line:net.ipv4.ip_forward = 0

   If there is such a line, change the line to:

   **net.ipv4.ip_forward = 1**

   If there is no such a line, add a line as below:

   **net.ipv4.ip_forward = 1**
3. Update the IP forwarding setting with the following command: **/sbin/sysctl --system**

**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 a kernel runtime parameter, the changes take effect immediately. This is because the sysctl command interacts directly with the kernel's sysctl interface, which allows you to read and modify kernel parameters at runtime.

The kernel's sysctl interface then applies the new values to the corresponding kernel parameters, and any associated behavior or functionality will be affected immediately. For example, if you use sysctl to increase the size of the kernel's file buffer cache, the kernel will start caching more files in memory immediately.

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

    To make a kernel parameter's value persist across reboots, you can add a line to the /etc/sysctl.conf file with the parameter and its desired value. For example, to set the maximum number of open files (file descriptors) allowed per process to 10000, you can add the following line to /etc/sysctl.conf:

    ```
    fs.file-max = 10000
    ```

    After adding the line to the file, save and close it. The next time the system is booted, the kernel will read the /etc/sysctl.conf file and apply the configured values to the corresponding kernel parameters.

    You can also use the sysctl command to load a configuration file containing kernel parameter settings, by running the command with the -p option followed by the path to the configuration file.

    ```
    sudo sysctl -p /etc/sysctl.conf
    ```

    This will apply the settings from the specified file to the kernel parameters, and the changes will persist across reboots.

    Note that not all kernel parameters can be set in the /etc/sysctl.conf file, and some parameters may require additional configuration steps to persist across reboots.

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?**

    Changes made to kernel parameters inside a container do not affect the kernel parameters of the host, and vice versa.

    This is because a container runs within its own namespace, which provides a layer of isolation between the container and the host system. Each namespace includes its own set of kernel parameters, so changes made within a container are only visible within that container.

## SSH

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

SSH or Secure Shell is a network communication protocol that enables two computers to communicate (c.f http or hypertext transfer protocol, which is the protocol used to transfer hypertext such as web pages) and share data.

To check if SSH is enabled on your system, open a command prompt and end the command ssh. If it provides you with help for using SSH, it is already enabled! You should be able to follow the Linux instructions using the ssh-keygen command from the command prompt.

**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.

Overall, SSH is a more secure and reliable protocol than telnet, and is widely used for remote system administration and file transfer.

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

The ~/.ssh/known_hosts file is used by the SSH (Secure Shell) client to store information about remote hosts that the client has previously connected to. Specifically, the file contains a list of public keys for remote hosts, along with information about their corresponding hostnames or IP addresses.

The ~/.ssh/known_hosts file is a security feature of SSH, as it helps to prevent man-in-the-middle attacks by ensuring that the SSH client is connecting to a known, trusted remote host. However, it's important to periodically review the contents of the file and remove any entries that are no longer needed or are no longer valid.

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

This error message indicates that the SSH client has detected a change in the remote host's public key since the last time you connected to it. This could happen if the remote host's SSH configuration has changed, or if the remote host has been compromised by an attacker who is attempting a man-in-the-middle attack.

To resolve this issue, we can try one of the following options:

1. Verify the public key manually: If we trust the remote host and believe that its public key has been legitimately changed, we can manually verify the public key and add it to our ~/.ssh/known_hosts file. To do this, we can use the ssh-keygen command to extract the remote host's public key fingerprint, then compare it with the fingerprint provided by the remote host administrator. If the fingerprints match, we can add the public key to our ~/.ssh/known_hosts file using the ssh-keyscan command.

2. Remove the old public key: If we no longer trust the remote host or believe that its public key has been compromised, we can remove the old public key from our ~/.ssh/known_hosts file. The next time we connect to the remote host, the SSH client will prompt us to verify the new public key and add it to our ~/.ssh/known_hosts file if we choose to do so.

**58. What is the difference between SSH and SSL?**

| **SSH** | **SSL** |
|---|---|
| Used for securely and remotely connecting to another machine to issue commands. | Used for securely transmitting data between two parties – normally a visitor to your website and your website’s server |

| | |
|---|---|
| Based on network tunnels | Based on digital certificates (i.e. SSL certificate) |
| Runs on port 22 | Runs on port 443 |
| Requires the client to authenticate with a username/password or cryptographic key | Only requires authentication on the server side (the client isn't required to authenticate) |
| Is a cryptographic network protocol | Is a security protocol |

**59. What ssh-keygen is used for?**

ssh-keygen is a command-line utility that is used to generate, manage, and convert SSH keys. SSH keys are a form of public key cryptography that is used to securely authenticate a user or device when connecting to a remote server over an SSH connection.

**Generating new SSH keys:** The ssh-keygen command can be used to generate a new SSH key pair, which consists of a public key and a private key. The private key is kept on the user's local machine, while the public key is uploaded to the remote server to authenticate the user or device.

**Managing existing SSH keys:** ssh-keygen can be used to manage existing SSH keys by changing their passphrase, converting them to different formats, or printing their public key fingerprint.

**Converting between SSH key formats:** ssh-keygen can be used to convert between different SSH key formats, such as OpenSSH and PuTTY, to ensure compatibility with different SSH clients or servers.

**Creating SSH key-based authentication:** ssh-keygen is used to create and configure SSH key-based authentication, which allows users or devices to authenticate with remote servers without having to enter a password.

**60. What is SSH port forwarding?**

SSH port forwarding, also known as SSH tunneling, is a technique for forwarding network traffic from one computer to another over an encrypted SSH connection.
With SSH port forwarding, you can securely access resources on a remote network that would otherwise be inaccessible due to firewall or other network restrictions.

There are two types of SSH port forwarding:

1. **Local port forwarding**: This is used to forward traffic from a local port on your local machine to a remote host and port. For example, you could forward traffic from your local machine's port 8080 to a remote server's port 80.

2. **Remote port forwarding:** This is used to forward traffic from a remote port on a remote host to a local host and port. For example, you could forward traffic from a remote server's port 5432 to your local machine's port 5432.

SSH port forwarding can be set up using the ssh command-line tool. By default, SSH port forwarding is encrypted, so it provides a secure way to access remote resources over an insecure network.