V I S H W A S C P

Assignment-2 Linux

Format:Lab Session
Time:90 mins

Instruction:
1.Complete the assignment and also upload the solutions in your Git repo
2.Take screenshot of the solution and paste it in word document
3.Convert the word doc into pdf before uploading.

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

   In the Linux FHS, the / represents the root directory of the file system hierarchy.

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 executable files that are required for the system to operate correctly, such as basic system utilities like ls, cp, and mv.
- /sbin: This directory contains system administration binaries that are generally used by the system administrator, like daemons, system initialization scripts, and low-level system utilities.

- /usr/bin: This directory contains non-essential user binaries, which are executable files that are generally installed by the system administrator or package manager, like programming tools, editors, and graphical applications.
- /usr/sbin: This directory contains non-essential system administration binaries, which are generally used by the system administrator.
- /etc: This directory contains system configuration files, which are used to configure various system and application settings.
- /home: This directory contains the home directories of regular users on the system.
- /var: This directory contains variable data files, which are files that are expected to grow over time, such as logs, spool files, and temporary files.
- /tmp: This directory contains temporary files that are created and used by applications and users. These files are generally expected to be deleted automatically, either when the application that created them exits or when the system is rebooted.

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

   The /tmp directory is special compared to other directories because it is intended to be used as a temporary storage location for files that are only needed for a short period. The contents of the /tmp directory are not preserved across system reboots and are typically deleted automatically.

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

   The /proc directory is a virtual file system that contains information about running processes and system resources. It provides a way for users and applications to access and manipulate kernel data structures and system statistics, such as CPU usage, memory usage, and network connections.

5. What makes /proc different from other filesystems?

   /proc is different from other file systems because it does not contain actual files and directories on disk. Instead, it provides a virtual view of the system's internal data structures and kernel interfaces, which are presented as files and directories in the /proc directory. This allows users and applications to access and manipulate kernel data and system statistics in a standardized and structured way.

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

   False. Any user can create files in the /proc directory, but only root or the owner of the file can modify or delete them. However, creating arbitrary files in /proc may not be a good idea and may cause unexpected behavior.

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

   The /proc/cmdline file contains the command-line arguments that were passed to the kernel when it was booted. This includes options such as the root file system, kernel parameters, and boot loader options.

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

   System devices such as block storage can be found in the /dev directory. The actual devices will be named differently depending on the type of device and the configuration of the system. For example, block storage devices may be named /dev/sda, /dev/sdb, etc.

Permissions

9. How to change the permissions of a file?

   To change the permissions of a file in Linux, you can use the chmod command followed by the desired permission mode and the file name chmod [permission mode] [file name]

   The permission mode consists of three digits that represent the permissions for the owner, group, and others, respectively. Each digit can be any combination of the following values:

- 4: read permission
- 2: write permission
- 1: execute permission

  To set the permissions, you can add up the values for each category. For example:

- 777: read, write, and execute permission for owner, group, and others
- 755: read, write, and execute permission for owner, and read and execute permission for group and others
- 644: read and write permission for owner, and read permission for group and others

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

- 777: This is a permission mode where the owner, group, and others have read, write, and execute permissions on the file or directory. This means that anyone can read, write, and execute the file.
- 644: This is a permission mode where the owner has read and write permissions, and the group and others have only read permission on the file. This means that only the owner can modify the file, while anyone can read it.
- 750: This is a permission mode where the owner has read, write, and execute permissions, the group has read and execute permissions, and

others have no permissions on the file or directory. This means that only the owner and members of the group can access the file and execute it, while others cannot do anything with the file.

11. What this command does? chmod +x some_file

The command **chmod +x some_file** adds execute permission to the file named **some_file**.
The **+x** option adds execute permission to the file, which means that the file can be executed as a program if it contains executable code. This command will give the owner, group, and others the permission to execute the file, if they have the appropriate privileges.

12. Explain what is setgid and setuid

- **setgid** (set group ID) is a permission that is set on an executable file. When a user executes a file with the **setgid** bit set, the process runs with the group ID of the file instead of the group ID of the user who launched it. This can be useful for allowing users to access files that belong to a specific group, even if they are not a member of that group.
- **setuid** (set user ID) is a permission that is also set on an executable file. When a user executes a file with the **setuid** bit set, the process runs with the user ID of the file owner instead of the user who launched it. This can be useful for allowing users to perform tasks that require elevated privileges, such as installing software or managing system settings.

13. What is the purpose of sticky bit?

The purpose of the sticky bit is to restrict the deletion of files within a directory. When the sticky bit is set on a directory, only the owner of a file, the owner of the directory, or the root user can delete the file. This can be useful in shared environments where multiple users have write access to the same directory, to prevent accidental deletion of files by other users.

The sticky bit is set using the **chmod** command with the **+t** option, like this: **chmod +t directory_name**.

14. What the following commands do?
chmod
chown
Chgrp

- **chmod**: The **chmod** command is used to change the permissions of a file or directory. It can be used to add or remove read, write, or execute permissions for the owner, group, and others. The **chmod** command can also be used to set the setuid, setgid, and sticky bit permissions.
- **chown**: The **chown** command is used to change the owner of a file or directory. It can be used to change the user or group ownership of a file or directory. The user who owns a file or directory has full control over it, including the ability to modify its permissions.
- **chgrp**: The **chgrp** command is used to change the group ownership of a file or directory. It can be used to change the group that has access to a file or directory. The group who owns a file or directory can be granted permissions to read, write or execute the file or directory as a group.

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

sudo (short for "superuser do") is a command that allows a user to run commands with the privileges of the superuser or another user. This can be useful for executing administrative tasks that require elevated privileges, such as installing packages or modifying system settings.
To set up **sudo**, you must first ensure that the user account is added to the **sudo** group. This can be done by adding the user to the **/etc/sudoers** file using the **visudo** command. The **visudo** command opens the **/etc/sudoers** file in a text editor, and ensures that the syntax of the file is correct before saving it.

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

True. In order to install packages on the system, one must either be logged in as the root user or use the **sudo** command to execute the installation command with superuser privileges.

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

ACLs (Access Control Lists) are a way of specifying more fine-grained permissions for files and directories than the standard Unix file permissions allow. ACLs allow you to grant specific permissions to individual users or groups, and to create more complex permission schemes that are tailored to the needs of your organization or application.

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

- The user does not have permission to create a file in the directory where they are trying to create it. This could happen if the directory has restrictive permissions that only allow certain users or groups to create files in it.
- The file already exists and the user does not have permission to overwrite it. This could happen if the file is owned by another user or group, or if the permissions on the file do not allow the user to modify it.
- The disk is full or has reached its quota limit, so there is no more space available to create the file.

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, making it impossible to change file permissions using **chmod** command in the future. To fix it, you will need to restore the execute permission on the **chmod** command. One way to do this is to use the **su** command to switch to the root user and then use the **chmod** command to add the execute permission back to **chmod**.

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. One common way is to use the **scp** (secure copy) command, which uses SSH to securely copy files between hosts. Here's an example command to copy a file called **file.txt** from the local host to a remote host at IP address **192.0.2.1**:

```
scp file.txt user@192.0.2.1:/path/to/destination
```

In this command, **user** is the username you use to log into the remote host, and **/path/to/destination** is the path on the remote host where you want to copy the file to. You will be prompted to enter the password for the remote user account to complete the copy operation.

21. How to generate a random string?

To generate a random string in Linux, you can use the **openssl** command with the **rand** option to generate a random sequence of bytes, and then use **base64** to encode the bytes as a printable string. Here's an example command:

```
openssl rand -base64 12
```

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

```
openssl rand -base64 5 | tr -dc 'a-zA-Z0-9' | head -c7
```

This command generates a random sequence of 5 bytes (which is equivalent to 7 base64-encoded characters), removes any non-alphanumeric characters using **tr**, and then uses **head** to output only the first 7 characters of the resulting string.

Systemd

23. What is systemd?

systemd is a system and service manager for Linux operating systems. It is responsible for controlling the startup and shutdown processes of the system, managing system services and daemons, and providing advanced logging and monitoring capabilities.

24. How to start or stop a service?

```
sudo systemctl start httpd
```

```
sudo systemctl stop httpd
```

25. How to check the status of a service?

```
sudo systemctl status httpd
```

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

On a system that uses **systemd**, you can view logs using the **journalctl** command.

```
sudo journalctl -b
```

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

To make a certain process or application a service, you need to create a **systemd** unit file. This file defines the service and its configuration, including the command to start the process, any dependencies, and other options. Once the unit file is created, you can use the **systemctl** command to manage the service.

28. Troubleshooting and Debugging

- Examining system logs using the **journalctl** command, which can help identify errors, warnings, and other issues
- Using the **systemctl** command to check the status of services and troubleshoot startup or shutdown issues
- Using the **strace** command to trace system calls and signals for a given process, which can help identify issues with system resources or permissions
- Examining the output of the **dmesg** command, which displays the kernel ring buffer and can help identify issues with hardware or system configuration
- Using a debugger such as **gdb** to analyze and debug the source code of a program.

29. Where system logs are located?

System logs in Linux are typically stored in the **/var/log** directory, which contains log files for various system components and services. The specific location and format of log files can vary depending on the distribution and configuration of the system.

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

To follow a file's content as it is being appended without opening the file every time, you can use the **tail** command with the **-f** (follow) option. For example, to follow the contents of a file called **logfile.txt**

```
tail -f logfile.txt
```

This command will display the last 10 lines of the file, and then continuously update the output as new lines are added to the file.

31. What are you using for troubleshooting and debugging network issues?
    - ping: To check the connectivity between two network devices
    - traceroute: To trace the path packets take through the network
    - netstat: To display active network connections and their status
    - tcpdump: To capture and analyze network traffic
    - wireshark: A GUI-based tool to capture and analyze network traffic
    - nmap: To scan for open ports and services on a remote system
    -

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

- df: To display disk space usage
- du: To estimate file space usage
- fsck: To check and repair file system errors
- badblocks: To check for bad blocks on a disk
- smartctl: To monitor and analyze disk health
- lsof: To list open files and processes accessing them

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

- ps: To display information about active processes

- top: To display real-time information about system processes and resource usage
- htop: A more advanced version of top
- strace: To trace system calls made by a process
- ltrace: To trace library calls made by a process
- gdb: A powerful debugger for analyzing and debugging program code

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

- top: To display CPU usage by processes
- htop: A more advanced version of top
- perf: A profiling tool to measure CPU performance
- mpstat: To display CPU statistics and usage
- sar: To collect and report system activity information

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

- Check CPU and memory usage using top or htop
- Check disk usage and I/O using df and iostat
- Check network connectivity using ping and netstat
- Check system logs in /var/log for any error messages or warnings
- Check running processes and their resource usage using ps

36. Explain iostat output

- Device: The name of the device being monitored
- tps: The number of transfers per second, which indicates disk I/O rate
- kB_read/s: The amount of data read from the device per second
- kB_wrtn/s: The amount of data written to the device per second
- kB_read: The total amount of data read from the device
- kB_wrtn: The total amount of data written to the device

37. How to debug binaries?

- gdb: A powerful debugger for analyzing and debugging program code
- strace: To trace system calls made by a process
- ltrace: To trace library calls made by a process
- objdump: To display information about object files and executable binaries
- readelf: To display information about ELF (Executable and Linkable Format) files

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

CPU load refers to the number of processes waiting to be executed by the CPU, while CPU utilization refers to the percentage of time the CPU is actively executing processes. A high CPU load indicates a system with many processes waiting to be executed, while a high CPU utilization indicates a system with many processes actively using the CPU.

39. How you measure time execution of a program?

To measure the time execution of a program, the time command can be used. Simply prefix the command with "time"

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?

1. Use the **fuser** command with the **-m** option to find the process ID (PID) of the process accessing the file. For example, if the file is located at **/path/to/file.txt**, run the following command:

```
fuser -m /path/to/file.txt
```

2. The **fuser** command will output the PID of the process that is accessing the file. Note down the PID.
3. Use the **kill** command to terminate the process. For example, if the PID of the process is 12345, run the following command:

   ```
   kill 12345
   ```

   This will send a signal to the process asking it to terminate gracefully. If the process does not respond to the signal, you can use the **kill** command with the **-9** option to forcefully terminate the process:

   ```
   kill -9 12345
   ```

Kernel

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

A kernel is the central component of an operating system that manages system resources and provides a platform for applications to run on. It is responsible for managing memory, processing tasks, managing input/output requests, and communicating with hardware devices.

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 run the command **uname -r** in a terminal window. This will display the version number of the currently running kernel.

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

A Linux kernel module is a piece of code that can be dynamically loaded and unloaded from the kernel at runtime. It allows users to add new functionality to the kernel without having to recompile the entire kernel. To load a new module, you can use the **insmod** command followed by the path to the module file. For example, **sudo insmod /path/to/module.ko**.

44. Explain user space vs. kernel space

User space refers to the part of the operating system where user-level applications and processes run. Kernel space, on the other hand, refers to the part of the operating system where the kernel and device drivers run. The main difference between the two is that user space processes cannot directly access hardware resources, while kernel space processes have direct access.

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

Configuration changes can be made to the kernel during three phases of its lifecycle: compile time, boot time, and runtime. Compile-time configuration is done by modifying the kernel source code before it is compiled. Boot-time configuration is done by passing parameters to the bootloader or kernel at boot time. Runtime configuration is done by modifying kernel parameters while the system is running.

46. Where can you find kernel's configuration?

The kernel configuration file is usually located in the **/boot** directory and is named **config-<kernel-version>**. For example, if your kernel version is 5.10.0-8-amd64, the configuration file would be located at **/boot/config-5.10.0-8-amd64**.

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 usually located in the **/boot** directory and is named **grub.cfg** or **menu.lst** depending on the boot loader being used.

48. How to list kernel's runtime parameters?

You can list the kernel's runtime parameters by running the command **sysctl -a** in a terminal window. This will display a list of all the current runtime parameters and their values.

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 of the kernel's runtime parameters can only be modified by the root user, so a regular user may not have access to all the parameters.

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

To enable IPv4 forwarding in the kernel, you can run the command **sysctl net.ipv4.ip_forward=1** in a terminal window. This will enable IPv4 forwarding immediately, but the change will not persist after a reboot unless it is added to the system's configuration files.

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

When you run the **sysctl** command to modify kernel runtime parameters, the changes are applied immediately to the running kernel. This is because **sysctl** communicates directly with the kernel to make the changes.

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

Changes to kernel runtime parameters can persist in a few ways. One way is to add the parameter and its value to the appropriate configuration file for your system, such as **/etc/sysctl.conf**. Another way is to use the **sysctl** command with the **-w** option followed by the parameter and its value, which will make the change persistent until the next reboot.

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 in a container only affect the kernel parameters within that container. They do not affect the kernel parameters of the host on which the container runs.

SSH

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

SSH (Secure Shell) is a protocol used to securely connect to a remote server over a network. To check if a Linux server is running SSH, you can use the **systemctl status sshd** command in a terminal window. If the SSH server is running, it will display information about its status.

55. Why SSH is considered better than telnet?

SSH is considered better than Telnet for several reasons. Telnet sends data in clear text, which means that any data sent over Telnet can be intercepted and read by anyone on the network. SSH, on the other hand, uses encryption to protect data sent over the network. Additionally, Telnet does not provide authentication, so it is easy for anyone to connect to a Telnet server and potentially gain access to sensitive information. SSH provides strong authentication mechanisms to ensure that only authorized users can access the server.

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

The **~/.ssh/known_hosts** file contains a list of public keys for remote servers that the user has connected to in the past. It is used by the SSH client to verify the identity of the server when connecting in the future.

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

"Host key verification failed" means that the SSH client was unable to verify the identity of the remote server. This can happen if the server's public key has changed since the user last connected to it, or if the user is connecting to a different server with the same hostname or IP address. To resolve this issue, the user should remove the server's entry from the **~/.ssh/known_hosts** file and try connecting again.

58. What is the difference between SSH and SSL?

SSH and SSL (Secure Sockets Layer) are both protocols used to provide secure communications over a network, but they are used for different purposes. SSH is primarily used for secure remote access to a server, while SSL is primarily used for secure communication between a client and a server, such as when accessing a website over HTTPS.

59. What ssh-keygen is used for?

ssh-keygen is a command-line utility used to generate SSH keys. SSH keys are used for authentication when connecting to a remote server over SSH. The **ssh-keygen** utility can generate both public and private keys, which are used to encrypt and decrypt data sent between the client and server.

60. What is SSH port forwarding?

SSH port forwarding, also known as SSH tunneling, is a feature of SSH that allows a user to forward network traffic from one port on a local machine to another port on a remote machine. This can be used to securely access services on a remote server that are not accessible directly from the user's local network. For example, a user could use SSH port forwarding to access a web server running on port 80 of a remote server, even if port 80 is blocked by a firewall.