Linux

Commands

By:

Tanu Singh

Linux :- 1). Open-Source
2). Unix-like OS.
3). Family of O.S

Linux distribution :-

Ubuntu, one of the most popular desktop Linux distribution

Linux : REPL

Read Evaluate print loop

Note :- For any Linux command you can check the usage and different i/p flags it expects by running the command

followed by (--help)

Linux Commands :-

→ (Present Working directory)

1. Pwd :- This represents what is the current directory we are currently at.

2. ls :- you can print the content directory we are in. All the files and sub-directories will be printed.

3. cd :- Change directory

→ This can help you to move into folder and move out of a folder.

4. Cd.. :- If you want to jump one folder back from the current directly.

(cd_..)
(Space)

5. Cd ../.. :- If you want to make two jumps back from the current directory.

6. Cd ~ :- from any directory this will help you to come back to the home directory

7. Cd directory1/directory2/ :-

we can move into internal Sub directories directly by Separating them with a forward Slash /.

Note :-

1). ~ : This tilda refers to the home directory.

Ex: \users\tanuj

2). Relative path :- It describes location of a file/folder w.r.t current folder.

Eg: cd.. → change directory w.r.t current folder/file.

3). Absolute path :- We mention the location from home directory or root directory.

Eg: cd

4). / :- this slash leads you to root directory.

5). When you give absolute path of a file or folder that means you will give the whole path of that file or folder, whereas in relative path you do jumps w.r.t to current folder.

6). Clear :- Clear the working space by actually moving you to the top of the current shell.

For Window people :-

(ls - la) (If you want to display the folder/file name on the screen)

↑ list ↑ listall

1. ls -la
2. ls

---

7). ls -l :- It will give you complete info of that particular path.

8). ls -a : It will give you the all file/folder present in the current directory.

9). ls --help :-

10). ls -l : list down more details about file such as owner, permission, date etc.

ls -lj : works like ls -l but also give size of the files.

ls -a : also lists files & folders starting with.

For Creating new Folder :-

1). @mkdir - make directory.

This helps us to create a new folder.

(b) cd folderName

(c) touch <filename>: we can create a brand new blank folder out of the blue.

(d) ls

2). Cat folderName

code.

Cat <filename> : prints the whole content of a file

3). rm filename : deleting file

rmdir foldername :- It will delete the folder.

→ (rm -rf)

4) mkdir Subfolder

touch Subfolder/new-file.txt

ls

cd..

Creating file

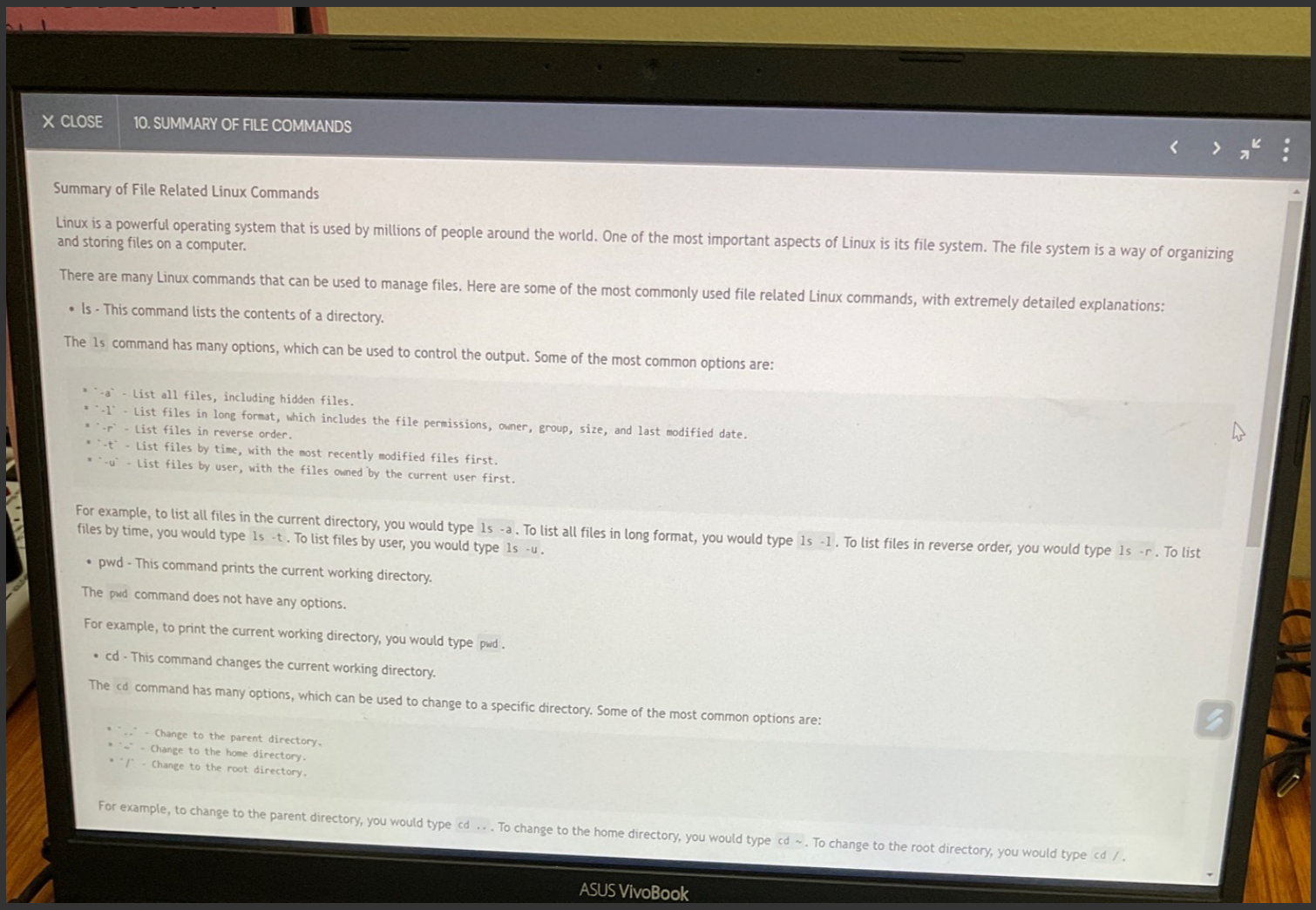
5). rm <filename> : This command deletes a file.

6). rmdir <foldername> : This command deletes an empty folder.

7). rm -r <foldename> : the -r flag enables rm to recursively delete all the content of the folder and delete the folder

8). rm -rf Subfolder
(recursive & forcefully)

→ Summary of file Commands :-



Summary of file related Linux Commands.

* `/` - Change to the root directory.

For example, to change to the parent directory, you would type `cd ..`. To change to the home directory, you would type `cd ~`. To change to the root directory, you would type `cd /`.

- mkdir - This command creates a new directory.

The `mkdir` command has one required argument, which is the name of the new directory.

For example, to create a new directory called `my_directory`, you would type `mkdir my_directory`.

- rmdir - This command removes an empty directory.

The `rmdir` command has one required argument, which is the name of the directory to be removed.

For example, to remove the empty directory `my_directory`, you would type `rmdir my_directory`.

- touch - This command creates an empty file.

The `touch` command has one required argument, which is the name of the new file.

For example, to create an empty file called `my_file`, you would type `touch my_file`.

- cp - This command copies a file or directory.

The `cp` command has many options, which can be used to control the copy operation. Some of the most common options are:

* `-a` - Copy all files, including hidden files and directories.
* `-r` - Copy directories recursively, including all subdirectories and files.
* `-v` - Verbose mode, which prints out a message for each file that is copied.

For example, to copy the file `my_file` to the directory `/tmp`, you would type `cp my_file /tmp`. To copy the directory `my_directory` recursively to the directory `/backup`, you would type `cp -r my_directory /backup`.

- mv - This command moves or renames a file or directory.

The `mv` command has many options, which can be used to control the move or rename operation. Some of the most common options are:

* `-f` - Force the move or rename operation, even if the destination file or directory already exists.

X CLOSE 10. SUMMARY OF FILE COMMANDS

The `mkdir` command has one required argument, which is the name of the new directory.

For example, to create a new directory called `my_directory`, you would type `mkdir my_directory`.

- rmdir - This command removes an empty directory.

The `rmdir` command has one required argument, which is the name of the directory to be removed.

For example, to remove the empty directory `my_directory`, you would type `rmdir my_directory`.

- touch - This command creates an empty file.

The `touch` command has one required argument, which is the name of the new file.

For example, to create an empty file called `my_file`, you would type `touch my_file`.

- cp - This command copies a file or directory.

The `cp` command has many options, which can be used to control the copy operation. Some of the most common options are:

* `-a` - Copy all files, including hidden files and directories.
* `-r` - Copy directories recursively, including all subdirectories and files.
* `-v` - Verbose mode, which prints out a message for each file that is copied.

For example, to copy the file `my_file` to the directory `/tmp`, you would type `cp my_file /tmp`. To copy the directory `my_directory` recursively to the directory `/backup`, you would type `cp -r my_directory /backup`.

- mv - This command moves or renames a file or directory.

The `mv` command has many options, which can be used to control the move or rename operation. Some of the most common options are:

* `-f` - Force the move or rename operation, even if the destination file or directory already exists.
* `-v` - Verbose mode, which prints out a message for each file that is moved or renamed.

For example, to move the file `my_file` to the directory `/tmp`, you would type `mv my_file /tmp`. To rename the file `

Vim Introduction :- (Vim is CLI text editor)

(CLI - Command line Interface)

1) Vim <filename> : this will create a file (If it doesn't exist) & then open it in the normal mode. In normal mode we can do changes to the file but we can read it & navigate it. You can also use vi <filename> to do the same thing.

2) Now after opening if you want to start making changes you need to first of all make it change the mode from normal to insert mode. To go in the insert mode we can press i. If you want to come back to the normal mode then press esc.

3) Esc + :q → If you want to exit a file we can do this.

4) Esc + :q! → If file has some changes and we want to exist without saving changes.

5). Esc + :wq → If file has changes and we want to save it and then exit

6). l : In normal mode you can move the cursor right.

7). h : In normal mode you can move the cursor left.

8). j : In normal mode you can move the cursor down.

9). k: In normal mode you can move the cursor up.

→ You can use normal right left up down arrow keys as well to navigate.

← Play VIM Adventures →

learning VIM while playing a game.

10). dd: In normal mode, it will delete the line. the cursor is currently at

11). gg: In normal mode, it will make the cursor go on the first line.

G: In normal mode, it will make the cursor go on the last line.

12). W: In normal mode it can make you jump one word.

13). 2w: In normal mode, it will make you jump 2 words.

14). d2w: then this will delete 2 words.

15). :%s/foo/bar/: to replace all occurrences of foo with bar.

16). yw: in normal mode, it copies one word.

17). yy: in normal mode, it copies a whole line.

18). p: for pasting in normal mode.

19). tail:- last few lines

20). head:- starting few lines.

21). echo "Tanu Singh"

O/p: Tanu Singh

Linux Commands :-

1). ls | grep python: this will actually pass the O/p of ls as an I/p to grep, grep does a substring search of python on the O/p of ls.

2). ls > new.txt :- Whatever is the result of ls will be dumped into next.txt, nothing will be printed on the console. If the new.txt has some content already then what will be replaced.

3). ls >> new.txt: Whatever is the result of ls will be dumped into new.txt, nothing will be printed on the console. If the new.txt has some content already then the new content will be appended.

4). <command1> && <command2> :-
This executes command1 followed by command2 considering command1 has no errors.

5). cp file1 file2: copies the content of file1 to file2.

6). mv file1 file2: moves (cut paste) the file1 to a new position as file2.
This can also help us to rename a file.

7). tar -cf archive.zip 1.txt 2.txt : this command will add all the files mentioned after archive.zip into the zipped archives as mentioned.

8) tar -zcf archive1.zip 1.txt 2.txt :-

This command will not only add the files to the zip but also compress them.

9). tar xf archive1.zip -C extract :-

all the content of the archive will be extracted out into the extract folder.

Comprehensive Guide

Understanding Linux File Permissions and Binary Calculations: A Comprehensive Guide

Linux, renowned for its robust security features, relies on a sophisticated system of file permissions to control access to files and directories. In this comprehensive guide, we will explore Linux file permissions, their structure, and how to manipulate them using binary calculations with the `chmod` command. Understanding file permissions is essential for securing your system and data.

The Basics of File Permissions

In Linux, each file and directory has three distinct sets of permissions:

1. **Owner Permissions:** These permissions apply to the user who owns the file or directory.
2. **Group Permissions:** These permissions apply to the group associated with the file or directory.
3. **Other (or World) Permissions:** These permissions apply to all users who are neither the owner nor part of the group.

Each permission set comprises three components:

Each permission set comprises three components:

- **Read (r):** This permission grants the ability to view the content of a file or the names of files within a directory.
- **Write (w):** This permission grants the ability to modify, edit, or delete a file. For directories, it allows creating, deleting, or renaming files within.
- **Execute (x):** This permission grants the ability to run or execute a file. For directories, it permits access and traversal within.

Viewing Permissions

To view file permissions in Linux, use the `ls -l` command in the terminal. The output will appear like this:

```
-rw-r--r-- 1 user1 users 1024 Sep 3 10:00 myfile.txt
```

Here's a breakdown of the displayed information:

- The first character represents the file type. A hyphen `-` signifies a regular file, while `d` indicates a directory.
- The next nine characters represent the permissions for the owner, group, and others, respectively. In the example above, the owner (user1) has read and

ASUS VivoBook

```
-rw-r--r-- 1 user1 users 1024 Sep 3 10:00 myfile.txt
```

Here's a breakdown of the displayed information:

- The first character represents the file type. A hyphen `-` signifies a regular file, while `d` indicates a directory.
- The next nine characters represent the permissions for the owner, group, and others, respectively. In the example above, the owner (`user1`) has read and write permissions, while the group (`users`) and others have only read permission.
- The number `1` represents the number of hard links to the file.
- The owner (`user1`) and the group (`users`) are listed, followed by the file size, modification date, and the file's name.

Binary Calculation of Permissions

X CLOSE 21. LINUX FILE PERMISSIONS

You can calculate permissions using binary values. In this system, each permission type is assigned a numeric value:

- Read (r) = 4
- Write (w) = 2
- Execute (x) = 1

To set permissions using binary calculations, follow these steps:

1. **Assign a binary value to each permission type:** Assign values to read (r), write (w), and execute (x). For example, read = 4, write = 2, and execute = 1.
2. **Determine the desired permission set:** For instance, to set read and write permissions, you would calculate `4 (read) + 2 (write) = 6` .
3. **Apply the calculated value:** Use the `chmod` command with the calculated value. For example, to set read and write permissions for the owner, execute:

```
chmod 600 file
```

The `6` corresponds to read (4) + write (2), and `file` is the target file.

The `6` corresponds to read (4) + write (2), and `file` is the target file.

Changing Permissions Using Binary Calculation

To modify file permissions using binary calculation, use the `chmod` command. The syntax is as follows:

```
chmod [options] permissions file
```

- `[options]` can include flags like `-R` for recursive changes.
- `permissions` is the numeric value calculated using binary representation.
- `file` is the name of the file or directory whose permissions you want to change.

Example: Binary Calculation with `chmod`

Let's assume you want to set read and write permissions for the owner and read-only permissions for the group and others. You can calculate the binary value as follows:

- Owner: Read (4) + Write (2) = 6
- Group: Read (4)
- Others: Read (4)

Now, use the `chmod` command to apply these permissions:

```
chmod 644 myfile.txt
```


- Others: Read (4)

Now, use the `chmod` command to apply these permissions:

```
chmod 644 myfile.txt
```

This command sets read and write permissions for the owner (6), and read-only permissions for the group and others (4).

Best Practices for File Permissions

1. **Principle of Least Privilege:** Grant only the necessary permissions to users and groups. Avoid giving global read/write/execute permissions when they are not required.

2. **Regularly Audit Permissions:** Periodically review and audit file and directory permissions to ensure they align with your security policies.

3. **Use Groups:** Create groups and assign users to them to simplify permission management. Group permissions can help avoid setting permissions individually for each user.

4. **Secure Sensitive Files:** Critical system files and sensitive data should have restricted access to prevent unauthorized access or modification.

5. **Backup and Restore:** Before making extensive changes to file permissions, create backups or snapshots of critical data to avoid accidental data loss.

Conclusion

Linux file permissions are a foundational element of system security, offering granular control over who can access and manipulate files and directories. By understanding the binary calculations behind file permissions and how to manipulate them using the `chmod` command, you can bolster the security and integrity of your Linux system while safeguarding sensitive data. Following best practices, such as the principle of least privilege and regular permission audits, is essential for maintaining a secure Linux environment. Mastering file permissions is a vital skill for Linux administrators and users, contributing to the overall safety and stability of the system.

4. **Secure Sensitive Files:** Critical system files and sensitive data should have restricted access to prevent unauthorized access or modification.

5. **Backup and Restore:** Before making extensive changes to file permissions, create backups or snapshots of critical data to avoid accidental data loss.

Conclusion

Linux file permissions are a foundational element of system security, offering granular control over who can access and manipulate files and directories. By understanding the binary calculations behind file permissions and how to manipulate them using the `chmod` command, you can bolster the security and integrity of your Linux system while safeguarding sensitive data. Following best practices, such as the principle of least privilege and regular permission audits, is essential for maintaining a secure Linux environment. Mastering file permissions is a vital skill for Linux administrators and users, contributing to the overall safety and stability of the system.

Assignment

X CLOSE | 22. ASSIGNMENT 2

Assignment 1: File and Directory Manipulation

Create a directory named "LinuxBasics" in your home directory.

Inside the "LinuxBasics" directory, create a text file named "sample.txt" and add some sample text to it.

List the contents of the "LinuxBasics" directory, including hidden files and directories.

Rename the "sample.txt" file to "renamed.txt" within the "LinuxBasics" directory.

Delete the "LinuxBasics" directory and its contents.

Assignment 2: File Permissions and Ownership

Create a new text file named "secret.txt" in your home directory.

Change the permissions of "secret.txt" so that only you (the owner) can read and write to it, while others have no access.

Verify the permissions of "secret.txt" using the "ls" command.

Change the ownership of "secret.txt" to a different user on your system.

Check the ownership of "secret.txt" to ensure it has changed.

Assignment 3: Working with Processes

List all the running processes on your system using the "ps" command.

Find the process ID (PID) of the "bash" process running in your terminal.

Use the "kill" command to terminate the "bash" process by its PID.

Start a new background process (e.g., "sleep 60 &") and find its PID.

Use the "kill" command to stop the background process you started in the previous step.

Assignment 4: Text Manipulation

Create a new text file named "data.txt" with some sample text in it.

Use the "cat" command to display the contents of "data.txt."

ASUS VivoBook

Verify the permissions of "secret.txt" using the "ls" command.

Change the ownership of "secret.txt" to a different user on your system.

Check the ownership of "secret.txt" to ensure it has changed.

Assignment 3: Working with Processes

List all the running processes on your system using the "ps" command.

Find the process ID (PID) of the "bash" process running in your terminal.

Use the "kill" command to terminate the "bash" process by its PID.

Start a new background process (e.g., "sleep 60 &") and find its PID.

Use the "kill" command to stop the background process you started in the previous step.

Assignment 4: Text Manipulation

Create a new text file named "data.txt" with some sample text in it.

Use the "cat" command to display the contents of "data.txt."

Use the "grep" command to search for a specific word or phrase within "data.txt."

Redirect the output of "cat data.txt" to a new file named "copy.txt."

Count the number of lines in "data.txt" using a command and display the result.

Assignment 5: Archiving and Compression

Create a new directory named "MyFiles" in your home directory.

Inside "MyFiles," create three text files with different content.

Archive the "MyFiles" directory into a compressed tarball (e.g., "myfiles.tar.gz").

Extract the contents of the tarball back into a directory called "ExtractedFiles."

Verify that the contents of "MyFiles" and "ExtractedFiles" are the same.