#90DaysOfDevOps Challenge - Day 3 - Basic Linux Commands (2)

Welcome back to Day 3 of the #90DaysOfDevOps challenge! In today's session, I will continue exploring basic Linux commands that are essential for any DevOps engineer. These commands will help you navigate and manage your Linux system efficiently. So, let's dive in!

1. Viewing the content of a file

To view the content of a file, you can use the `cat` command followed by the filename. For example:

```
cat filename
```

This will display the entire contents of the file on your terminal. It's a simple and effective way to quickly check the contents of a file.

2. Changing the access permissions of files

The `chmod` command is used to change the access permissions of files in Linux. It allows you to modify the read, write, and execute permissions for the owner, group, and others.

```
chmod [options] filename
```

You can specify different options with the `chmod` command to set specific permissions. Some common options include:

- `u` (user/owner): Changes the permissions for the file owner.
- `g` (group): Changes the permissions for the group associated with the file.
- `o` (others): Changes the permissions for users who are neither the owner nor in the group.
- `+` (add): Adds the specified permission.
- `-` (remove): Removes the specified permission.
- `=` (assign): Assigns the specified permission.

For example, to give the owner read and write permissions on a file, you can use:

```
chmod u+rw filename
```

3. Checking command history

To check the commands you have previously run in your terminal session, you can use the `history` command. Simply type `history` in your terminal, and it will display a list of commands along with their respective line numbers.

```
history
```

This command is especially useful when you want to recall and reuse previously executed commands.

4. Removing a directory/folder

To remove a directory in Linux, you can use the `rmdir` command followed by the directory name.

```
rmdir directory
```

However, please note that the `rmdir` command can only be used to remove empty directories. If you want to remove a non-empty directory, you should use the `rm` command with the `-r` option, which stands for recursive.

The main differences between `rmdir` and `rm` are:

- `rmdir` can only remove empty directories, whereas `rm -r` can remove both empty and non-empty directories.
- `rmdir` is a safer option as it prevents accidental removal of non-empty directories.
- `rm -r` is more powerful but should be used with caution to avoid unintended data loss.

5. Creating a fruits.txt file and viewing the content.

To create a new file in Linux, you can use the `touch` command followed by the desired filename. For example:

```
touch fruits.txt
```

This command will create an empty file with the specified name. To view the content of a file, you can use the `cat` command we discussed earlier.

6. Adding content in fruits.txt (One in each line) - Apple, Mango, Banana, Cherry, Kiwi, Orange, Guava.

To add content to a file, you can use a text editor like `vi` or `nano`. However, if you want to append content from the command line, you can use the `echo` command and redirect the output to the file.

For example, to add the following fruits to a file named `fruits.txt`, one fruit per line:

```
Apple
Mango
Banana
Cherry
Kiwi
Orange
Guava
```

You can use the following command:

```
echo -e "Apple\nMango\nBanana\nCherry\nKiwi\nOrange\nGuava" > fruits.txt
```

- `echo`: The `echo` command is used to display text or variables on the terminal.
- `-e`: This option enables the interpretation of backslash escapes. It allows us to include special characters, such as newline (`\n`), in the output.
- `"Apple\nMango\nBanana\nCherry\nKiwi\nOrange\nGuava"`: This part of the command represents the text or content that will be added to the `fruits.txt` file. Each fruit name is separated by the newline character (`\n`), ensuring that each fruit appears on a new line in the file.
- `>`: This symbol is a redirection operator that directs the output of the `echo` command to a file.
- `fruits.txt`: This is the filename of the file where the echoed content will be saved. In this case, it's `fruits.txt`.

7. Showing the top three items from a file

To display the top three items from a file, you can use the `head` command. By default, it shows the first ten lines of a file, but you can specify the number of lines using the `-n` option.

```
head -n 3 filename
```

The `head` command is particularly useful when you want to get a quick preview of the contents of a large file.

8. Showing the bottom three items from a file

To display the bottom three items from a file, you can use the `tail` command. Similar to `head`, the `tail` command also displays the last ten lines of a file by default. You can use the `-n` option to specify the number of lines to display.

```
tail -n 3 filename
```

The `tail` command is often used to monitor log files or track real-time changes in files.

9. Creating another file names Colors.txt and viewing the content.

To create a new file and view its content, you can use the `touch` command to create the file, and then use the `cat` command to view the contents. For example:

```
touch Colors.txt
cat Colors.txt
```

This will create an empty file named `Colors.txt` and display its content, which, in this case, will be empty.

10. Add content in Colors.txt (One in each line) - Red, Pink, White, Black, Blue, Orange, Purple, Grey.

Similar to adding content to `fruits.txt`, you can use the `echo` command to append content to the `Colors.txt` file.

```
echo -e "Red\nPink\nWhite\nBlack\nBlue\nOrange\nPurple\nGrey" > Colors.txt
```

This command will add the listed colors to the `Colors.txt` file, with each color on a new line.

11. Finding the difference between fruits.txt and colors.txt files

To find the difference between two files, you can use the `diff` command followed by the filenames. For example:

```
diff fruits.txt Colors.txt
```

The `diff` command will show the lines that are different between the two files, highlighting any changes made.

These are some basic Linux commands that will prove helpful throughout your DevOps journey. Understanding and practicing these commands will enable you to navigate and manage your Linux system with ease.

That's all for Day 3 of the #90DaysOfDevOps challenge. Stay tuned for Day 4 of the #90DaysOfDevOps challenge, where I'll explore basic Linux Shell scripting for DevOps Engineers.