Shell, scripting& permission

Alx Egypt Discord Server! ©

# Shell, scripting & permission

By Alx Egypt Discord Server

**Shell, scripting, and permission** are important concepts in Unix-like operating systems such as Linux and macOS.
The shell is a command-line interface that allows users to interact with the operating system and run commands.

**Scripting** is the process of writing scripts, which are programs that automate tasks and can be run from the shell.

**Permissions** are a set of rules that determine which users and processes can access and modify files and directories on the system. Understanding these concepts is essential for working with Unix-like systems and developing efficient and secure workflows.

## Commands Recap!

- **cd:**
    - This command changes the current working directory in the command line interface to a specified directory or path.
- **ls:**
    - This command lists the files and directories in the current directory.
- **Pwd:**
    - This command prints the current working directory.

- **Less:**
    - This command opens a file in a pager, which allows you to view the contents of the file one page at a time.

- **file:**
    - This command displays the type of a file, such as whether it is a text file, binary executable, or image file.

- **cp:**
    - This command copies files or directories from one location to another.

- **mv**:
    - This command moves files or directories from one location to another, and can be used to rename a file.

- **rm:**
    - This command removes files or directories. Be careful when using this command as it permanently deletes files and directories.

- **mkdir:**
    - This command creates a new directory.

- **type:**
    - This command displays the type of a command, such as whether it is a built-in command or an external command.

- **which**:
    - This command displays the location of a command or executable file.

- **help**:
    - This command displays help information for a specific command.

- **man**:
    - This command displays the manual page for a specific command, which provides detailed information about the command and its options.

## Shell Scripting

In Linux and macOS. Bash scripts are written in the Bash programming language, which is a scripting language that provides a set of commands and syntax for performing various tasks.

Here are some key features and benefits of Bash scripting in Linux:

1. Automation: Bash scripts can be used to automate repetitive tasks, such as file management, system administration, and software installation. This can save time and reduce errors caused by manual intervention.

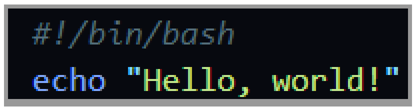
2. Customization: Bash scripting allows users to customize their Linux environment and create workflows that fit their specific needs.

3. Portability: Bash scripts can be run on any Unix-like system, making them portable and flexible.

4. Integration: Bash scripts can be integrated with other Linux tools and utilities, such as awk, sed, and grep, to perform complex tasks.

5. Debugging: Bash scripts can be easily debugged using the built-in debugging tools, such as the set -x command, which prints each command as it is executed.

Here's an example of a simple Bash script that displays a message:



```
#!/bin/bash
echo "Hello, world!"
```

In this example, the #!/bin/bash line specifies that the script should be run using the Bash shell. The echo command is used to display the message "Hello, world!" in the terminal.

Bash scripting can become increasingly complex, with conditionals, loops, functions, and more. However, with practice and patience, Bash scripting can be a powerful tool for automating tasks and improving productivity in Linux environments.

## Shebangs

Shebang (also known as a hashbang or a pound bang) is a special sequence of characters at the beginning of a script file in Unix-like operating systems, such as Linux and macOS. The shebang consists of two characters: the hash symbol (#) followed by an exclamation mark (!).

The shebang is used to specify the interpreter that should be used to execute the script file. The interpreter is a program that reads and executes the commands in the script file. By default, Unix-like systems do not know which interpreter to use for a script file, so the shebang is used to specify it.

Here's an example of a shebang line in a Bash script:

#!/bin/bash
In this example, the shebang specifies that the script file should be executed using the Bash shell (/bin/bash).

Here are some examples of shebang lines for various interpreters:

1. Bash shell:
   #!/bin/bash

2. Python 3:
   #!/usr/bin/env python3

3. Perl:
   #!/usr/bin/perl

4. Ruby:
   #!/usr/bin/env ruby

5. Node.js:
   #!/usr/bin/env node

## Shell Permission

In Linux and other Unix-like operating systems, permissions are a set of rules that determine which users and processes can access and modify files and directories on the system. Permissions are an important aspect of Linux security and help to prevent unauthorized access and modification of sensitive files and directories.

Linux permissions are divided into three categories: owner, group, and others. Each category can have three types of permissions: read, write, and execute. The read permission allows a user to view the contents of a file or directory, the write permission allows a user to modify the contents of a file or directory, and the execute permission allows a user to run a file as a program or script.

### Here are some key features and benefits of Linux permissions:

- Security:
  - Linux permissions help to ensure the security of the system by controlling access to files and directories.

- Flexibility:
  - Linux permissions are flexible and can be customized to fit the needs of different users and applications.

- Granularity:
    - Linux permissions are granular, allowing for fine-grained control over file and directory access.

- Compatibility:
    - Linux permissions are compatible with other Unix-like operating systems, making it easy to share files and directories between systems.

- Accountability:
    - Linux permissions allow for accountability by keeping track of who has accessed or modified a file or directory.

Understanding Linux permissions is essential for working with Linux systems and administering them securely. It is important to properly set permissions on sensitive files and directories to prevent unauthorized access, and to regularly audit and review permissions to ensure that they are still appropriate.

## Commands you have to know:

1. chmod:
    a. This command changes the permissions of files and directories. Example: chmod 755 myfile.txt will set the permissions of "myfile.txt" to read, write, and execute for the owner, and read and execute for the group and others.

2. sudo:
    a. This command allows you to run a command with administrative privileges. Example: sudo apt-get update will update the system's package repositories with administrative privileges.

3. su:
    a. This command allows you to switch to another user account. Example: su root will switch to the root user account.

4. chown:
    a. This command changes the ownership of files and directories. Example: chown user myfile.txt will change the owner of "myfile.txt" to "user".

5. chgrp:
    a. This command changes the group ownership of files and directories. Example: chgrp group myfile.txt will change the group ownership of "myfile.txt" to "group".

6. id:
    a. This command displays information about the current user, such as user ID and group ID. Example: id will display the user ID and group ID of the current user.

7. groups:
    a. This command displays the groups a user belongs to. Example: groups user will display the groups that the user "user" belongs to.

8. Whoami:
    a. This command displays the username of the current user. Example: whoami will display the username of the current user.

9. Adduser:
    a. This command adds a new user account to the system. Example: adduser newuser will add a new user account called "newuser" to the system.

10. useradd:
    a. This command adds a new user account to the system. Example: useradd newuser will add a new user account called "newuser" to the system.

11. addgroup:
    a. This command adds a new group to the system. Example: addgroup newgroup will add a new group called "newgroup" to the system.

## Linux permissions Calculating:

In Linux, permissions are represented by a three-digit number that specifies the access rights for the file or directory. Each digit in the number represents a different category of user: owner, group, and others. The digits correspond to the read, write, and execute permissions, respectively.

Here's a breakdown of how the permission number is calculated:

1. Read permission is assigned a value of 4.
2. Write permission is assigned a value of 2.
3. Execute permission is assigned a value of 1.

To calculate the permission number, add up the values for the desired permissions for each category of user. For example, if the owner has read and write permissions, the group has read permission, and others have no permissions, the permission number would be calculated as follows:

```
Owner:   Read (4) + Write (2) = 6 .
Group:   Read (4)             = 4.
Others:  No permissions       = 0 .
-----------------------------------
Total:                        640
```

In this example, the permission number is 640, which corresponds to the following permissions:

- Owner: read and write (6 = 4 + 2)
- Group: read (4)

- Others: no permissions (0)

To set permissions on a file or directory in Linux, you can use the chmod command followed by the permission number. For example, to set the permissions for the file "example.txt" to 640, you would use the following command: `chmod 640 example.txt`

Overall, understanding how to calculate and set permissions in Linux is essential for managing file and directory access and ensuring the security of the system.

Made with love not skill, Good luck!
Alx Egypt Discord Server, @moe

Click Here To Join Our Discord Server!
Or use this link: https://discord.gg/alx