Linux terminal

Introdução Engenharia Informática

Mário Antunes

September 22, 2025

Exercise 1: Finding Your Way Around 🧭

This exercise is about exploring the filesystem without changing anything. It covers `pwd`, `ls`, `cd`, and basic information commands.

1. Open your terminal. Your starting location is your **home directory**. Verify this by printing the working directory.

```
$ pwd
```

2. List the contents of your home directory. Then, list them again, but this time showing **all** files (including hidden ones) in the **long** list format.

```
$ ls
$ ls -la
```

3. Navigate to the **root** directory (`/`). List its contents. You should see key system folders like `etc`, `home`, and `var`.

```
$ cd /
$ ls
```

4. Now, navigate to the system log directory located at `/var/log`. List its contents.

```
$ cd /var/log
$ ls
```

5. Let's get some information. Without moving, find out **who you are** and what the **current date and time** is.

```
$ whoami
$ date
```

6. Return to your home directory using the quickest shortcut. Verify you are back home.

```
$ cd ~
$ pwd
```

Exercise 2: Creating and Managing Files 📂

In this exercise, you'll create, copy, move, and delete files and directories.

1. From your home directory, create a new directory for this class called `IEI`.

   ```
   $ mkdir IEI
   ```

2. Navigate inside your new `IEI` directory.

   ```
   $ cd IEI
   ```

3. Create an empty file called `notes.txt`. You can use the `touch` command. Verify that the file was created.

   ```
   $ touch notes.txt
   $ ls
   ```

4. Add some text to your file using the `echo` command and the `>` **redirection** operator. Then, view its contents with `cat`.

   ```
   $ echo "My first line of text." > notes.txt
   $ cat notes.txt
   ```

5. Make a copy of your file and name it `notes_backup.txt`.

   ```
   $ cp notes.txt notes_backup.txt
   ```

6. Rename the original file from `notes.txt` to `important_notes.txt` using the `mv` (move) command. List the files to see the change.

   ```
   $ mv notes.txt important_notes.txt
   $ ls
   ```

7. Finally, clean up by deleting the backup file.

   ```
   $ rm notes_backup.txt
   ```

Exercise 3: Understanding Permissions 🔐

This exercise focuses on reading and changing file permissions with `chmod`.

1. Inside your `~/IEI` directory, create a new file called `secret_data.txt`.

```
$ touch secret_data.txt
```

2. View the file's permissions using `ls -l`. Note the default permissions for the **user** (you), the **group**, and **others**.

```
$ ls -l secret_data.txt
```

3. Remove **all** permissions (read, write, execute) for **everyone**. A quick way is with the numeric code `000`.

```
$ chmod 000 secret_data.txt
```

4. Try to view the file's contents with `cat`. You should get a **"Permission denied"** error. This is Linux security in action!

```
$ cat secret_data.txt
```

5. Now, give **only yourself** (the user) permission to **read and write** to the file. Then, try to `cat` it again.

```
$ chmod u+rw secret_data.txt
$ cat secret_data.txt
```

6. Create one more file called `my_script.sh`. Use `ls -l` to see that it is not executable by default. Use `chmod` to give **yourself execute** permission. Check the permissions again to see the change (the `x` will appear).

```
$ touch my_script.sh
$ chmod u+x my_script.sh
$ ls -l my_script.sh
```

Exercise 4: Combining Commands 🔗

Let's explore the power of the **pipe (`|`)** and **redirection (`>>`)**.

1. The command `ps aux` lists all running processes on the system. It's a lot of output! Run it to see.
2. Now, use the **pipe** to send that output to the `grep` command to find only the lines containing the word "bash". This will show you your own shell process.

```
$ ps aux | grep "bash"
```

3. Let's create a small log file. First, use `echo` and `>` to create a file named `activity.log` with one entry.

```
$ echo "$(date): Starting my work." > activity.log
```

4. Now, use `echo` and the **append operator (>>)** to add a second line to the file without deleting the first one.

```
$ echo "$(date): Finished exercise 4." >> activity.log
```

5. Verify that your `activity.log` file contains both lines.

```
$ cat activity.log
```

Exercise 5: Customizing Your Environment ✨

Time to edit your `.bashrc` file to create a handy shortcut (an alias).

1. First, navigate to your home directory.
2. Open your `.bashrc` file using the `nano` editor. Be careful not to delete anything!

```
$ nano ~/.bashrc
```

3. Scroll to the very bottom of the file and add the following line. This creates a shortcut `ll` that will run `ls -alF`.

```
alias ll='ls -alF'
```

4. Save the file and exit `nano` (`Ctrl+X`, then `Y`, then `Enter`).
5. The changes are not active yet. You must either open a new terminal or "source" the file to load the changes into your current session.

```
$ source ~/.bashrc
```

6. Test your new alias! It should give you a detailed list of all files.

```
$ ll
```

Exercise 6: Scripting Challenge 🚀

Let's bring it all together by writing a script. Create a file named `setup_project.sh` and add the content for each part. Remember to make it executable with `chmod +x setup_project.sh`!

Part A: The Basic Script

Write a script that creates a new directory named `my_project` inside your `~/IEI` folder.

```
#!/bin/bash
# Part A: Creates a project directory.

echo "Setting up project structure..."
mkdir ~/IEI/my_project
echo "Directory 'my_project' created."
```

Part B: Adding a Check

Improve your script. Use an `if` statement to check if the `my_project` directory **already exists**. If it does, print an error message and exit.

```
#!/bin/bash
# Part B: Checks if directory already exists.

PROJECT_DIR="~/IEI/my_project"

echo "Setting up project structure..."

if [ -d "$PROJECT_DIR" ]; then
  echo "Error: Directory '$PROJECT_DIR' already exists."
  exit 1
fi

mkdir "$PROJECT_DIR"
echo "Directory '$PROJECT_DIR' created."
```

Part C: Adding a Loop

Final step! Modify the script to use a `for` loop to create three subdirectories inside `my_project`: `assets`, `source`, and `docs`.

```
#!/bin-bash
# Part C: Creates subdirectories with a loop.

PROJECT_DIR="~/IEI/my_project"

echo "Setting up project structure..."

if [ -d "$PROJECT_DIR" ]; then
  echo "Error: Directory '$PROJECT_DIR' already exists."
  exit 1
fi

mkdir "$PROJECT_DIR"
```

```
echo "Directory '$PROJECT_DIR' created."

# Loop to create subdirectories
for folder in assets source docs
do
  mkdir "$PROJECT_DIR/$folder"
  echo "-> Created subfolder: $folder"
done

echo "Project setup complete!"
```