📁 Linux File Operations & Directory Organization

📘 Scenario Overview

In this challenge, I acted as a system administrator tasked with organizing a small software project using **only core Linux file operation commands**. The goal was to transform an unstructured directory into a clean, maintainable project layout using `cp`, `mv`, and `rm`.

This lab reinforces **essential Linux filesystem skills** used daily by system administrators and DevOps engineers.

🎯 Objectives

- Create a clean project directory structure
- Copy, move, and remove files appropriately
- Use only `cp`, `mv`, and `rm` for file operations
- Verify progress using standard navigation commands

📂 Initial Project Structure

Starting in the `~/project` directory:

```
project/
├── old_stuff/
│   ├── deprecated_script.sh
│   └── outdated_notes.txt
├── temp/
│   ├── draft_readme.md
│   └── config_backup.json
├── app.js
├── styles.css
```

```
└── data.json
```

🏗️ Step 1: Create Required Directories

Purpose

Prepare directories to organize source code and configuration files.

Commands Used

```
mkdir src config
```

📦 Step 2: Move Source Files into `src`

Purpose

Organize application code into a dedicated source directory.

Commands Used

```
mv app.js src/
mv styles.css src/
```

⚙️ Step 3: Move Configuration File into `config`

Purpose

Place configuration data into a centralized configuration directory.

Commands Used

```
mv data.json config/config.json
```

📝 Step 4: Create the README File

Purpose

Use existing documentation drafts to create a project README.

Commands Used

```
mv temp/draft_readme.md README.md
```

🧹 Step 5: Remove Unnecessary Files and Directories

Purpose

Clean up deprecated and temporary files no longer needed.

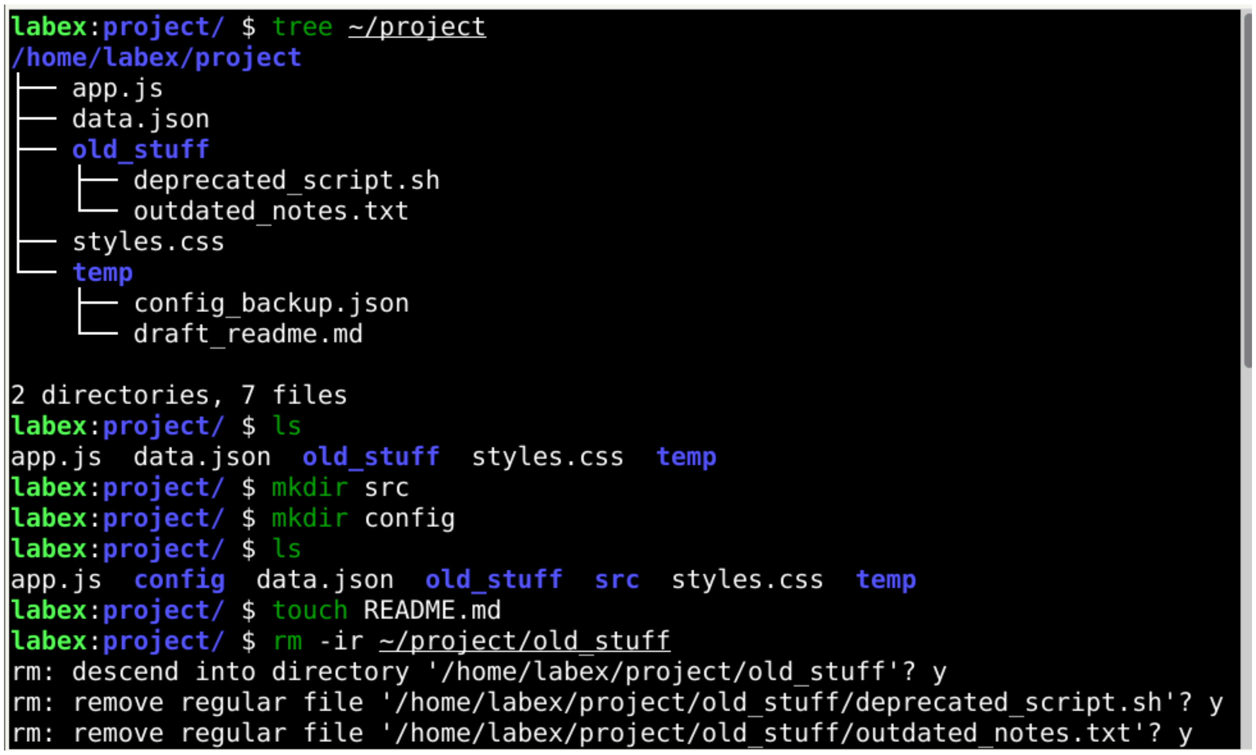
Commands Used

```
rm -r old_stuff
rm -r temp
```

✅ Final Project Structure

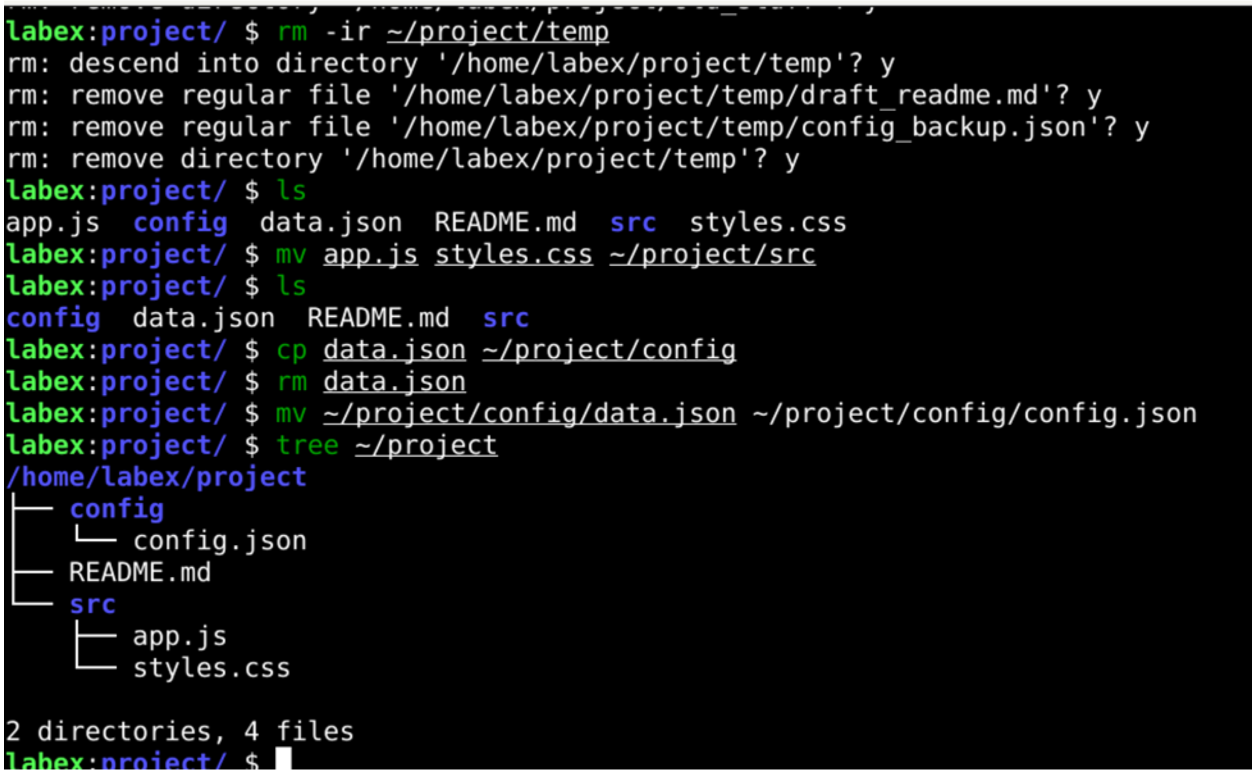
After completing all tasks, the directory structure was successfully reorganized:

```
project/
├── src/
│   ├── app.js
│   └── styles.css
├── config/
│   └── config.json
└── README.md
```



```
labex:project/ $ tree ~/project
/home/labex/project
├── app.js
├── data.json
├── old_stuff
│   ├── deprecated_script.sh
│   └── outdated_notes.txt
├── styles.css
└── temp
    ├── config_backup.json
    └── draft_readme.md

2 directories, 7 files
labex:project/ $ ls
app.js  data.json  old_stuff  styles.css  temp
labex:project/ $ mkdir src
labex:project/ $ mkdir config
labex:project/ $ ls
app.js  config  data.json  old_stuff  src  styles.css  temp
labex:project/ $ touch README.md
labex:project/ $ rm -ir ~/project/old_stuff
rm: descend into directory '/home/labex/project/old_stuff'? y
rm: remove regular file '/home/labex/project/old_stuff/deprecated_script.sh'? y
rm: remove regular file '/home/labex/project/old_stuff/outdated_notes.txt'? y
```



```
labex:project/ $ rm -ir ~/project/temp
rm: descend into directory '/home/labex/project/temp'? y
rm: remove regular file '/home/labex/project/temp/draft_readme.md'? y
rm: remove regular file '/home/labex/project/temp/config_backup.json'? y
rm: remove directory '/home/labex/project/temp'? y
labex:project/ $ ls
app.js  config  data.json  README.md  src  styles.css
labex:project/ $ mv app.js styles.css ~/project/src
labex:project/ $ ls
config  data.json  README.md  src
labex:project/ $ cp data.json ~/project/config
labex:project/ $ rm data.json
labex:project/ $ mv ~/project/config/data.json ~/project/config/config.json
labex:project/ $ tree ~/project
/home/labex/project
├── config
│   └── config.json
├── README.md
└── src
    ├── app.js
    └── styles.css

2 directories, 4 files
labex:project/ $ 
```

🧠 Skills Demonstrated

- Linux file manipulation (`cp`, `mv`, `rm`)
- Directory organization best practices
- Safe removal of obsolete files
- Project structure standardization
- Command-line navigation and verification
- Attention to constraints and requirements