Linux Filesystem Demo - Hands-on Exercise

Overview

This demonstration will guide you through basic filesystem operations in Linux, including directory creation, file creation using the `echo` command, and managing file permissions with `chmod`. You'll learn practical skills for organizing files and controlling access permissions.

Prerequisites

- Access to a Linux terminal
- Basic understanding of Linux commands
- User account with appropriate permissions

Demo Steps

Step 1: Create a Project Directory

Objective: Create a new directory called `project-a` in your home directory.

```
# Navigate to your home directory (if not already there)
cd ~

# Create the project-a directory
mkdir project-a

# Verify the directory was created
ls -la
```

Explanation:

- `cd ~`: Changes to your home directory (`~` is a shortcut for your home directory path)
- `mkdir project-a`: Creates a new directory named "project-a"
- `ls -la`: Lists all files and directories with detailed information including permissions, ownership, and timestamps

Expected Output: You should see `project-a` listed as a directory (indicated by `d` at the beginning of the permission string).

Step 2: Navigate to the Project Directory

```
# Change into the project-a directory
cd project-a

# Verify your current location
pwd
```

Explanation:

- `cd project-a`: Changes the current working directory to project-a
- `pwd`: Prints the current working directory path (should show `/home/[username]/project-a`)

Step 3: Create Files Using the Echo Command

Objective: Create two text files with initial content using the `echo` command.

```
# Create jan-25.txt with content
echo "January 2025 project tasks and goals" > jan-25.txt

# Create feb-25.txt with content
echo "February 2025 project milestones and deadlines" > feb-25.txt

# Verify both files were created
ls -la
```

Explanation:

- `echo "text" > filename`: Creates a file with the specified content
  - `echo`: Displays text to the terminal
  - `>`: Redirects the output to a file (overwrites if file exists)
- The files contain meaningful content related to project planning

Expected Output: You should see both files listed with their creation timestamps and default permissions.

Step 4: Verify File Contents

```
# Display the contents of both files
cat jan-25.txt
cat feb-25.txt

# Alternative: View both files at once
cat jan-25.txt feb-25.txt
```

Explanation:

- `cat filename`: Displays the entire content of a file
- This step confirms that the files were created with the correct content

Step 5: Check Current File Permissions

```
# View detailed file permissions
ls -la *.txt
```

Explanation:

- `ls -la *.txt`: Lists all .txt files with detailed permissions
- The permission string format is: `[file type][owner permissions][group permissions][other permissions]`
- Example: `-rw-r--r--` means:
  - `-`: Regular file
  - `rw-`: Owner can read and write
  - `r--`: Group can only read
  - `r--`: Others can only read

Step 6: Update File Permissions for Group Access

Objective: Modify file permissions to allow members of the "student" group to read the files.

```
# First, let's check what groups exist and which group we should use
groups

# Method 1: Set specific permissions using octal notation
chmod 640 jan-25.txt feb-25.txt

# Method 2: Alternative using symbolic notation
# chmod g+r jan-25.txt feb-25.txt

# Verify the permission changes
ls -la *.txt
```

Explanation:

- `groups`: Shows which groups the current user belongs to
- `chmod 640`: Sets permissions using octal notation:
  - `6` (110 in binary) = read(4) + write(2) for owner
  - `4` (100 in binary) = read(4) for group
  - `0` (000 in binary) = no permissions for others
- `chmod g+r`: Alternative symbolic method to add read permission for group
- This ensures that any user in the "student" group can read these files

Step 7: Verify Group Permissions

```
# Check the updated permissions
ls -la *.txt

# Test file access (if you have access to another user account)
# su - [another_user_in_student_group]
# cat /home/[your_username]/project-a/jan-25.txt
```

Expected Result: The permission string should now show `-rw-r-----` meaning:

- Owner: read and write
- Group (student): read only
- Others: no access

Step 8: Additional Verification Steps

```
# View file ownership and group information
stat jan-25.txt
stat feb-25.txt

# Check which users belong to the student group
getent group student
```

Explanation:

- `stat filename`: Shows detailed file information including permissions, ownership, and timestamps
- `getent group student`: Lists all users who belong to the "student" group

Summary of Commands Used

Command	Purpose	Example
`mkdir`	Create directory	`mkdir project-a`
`cd`	Change directory	`cd project-a`
`pwd`	Show current directory	`pwd`
`echo >`	Create file with content	`echo "text" > file.txt`
`ls -la`	List files with permissions	`ls -la *.txt`
`cat`	Display file contents	`cat jan-25.txt`
`chmod`	Change file permissions	`chmod 640 file.txt`
`stat`	Show detailed file info	`stat file.txt`
`groups`	Show user's groups	`groups`

Key Learning Points

1. **Directory Organization**: Creating logical folder structures helps organize projects
2. **File Creation**: The `echo` command is a simple way to create files with initial content
3. **Permission Management**: Understanding and controlling file access is crucial for security
4. **Group Permissions**: Proper group permissions enable controlled collaboration
5. **Verification**: Always verify your changes to ensure they work as expected

Troubleshooting Tips

- **Permission Denied**: Ensure you have write permissions in the target directory

- **Group Not Found**: Verify the "student" group exists with `getent group student`
- **Wrong Permissions**: Use `chmod 644` for general read access or `chmod 640` for group-only read access
- **File Not Found**: Use `pwd` and `ls` to verify your current location and file existence

Next Steps

After completing this demo, try:

1. Creating additional files with different content
2. Experimenting with different permission combinations
3. Creating subdirectories within project-a
4. Testing file access from different user accounts