Lab 3 – Manipulating File and Directory, Wildcards / Globbering, Links

Instruction and Example: Manipulating files and Directories

1. **Copying Files and Directories**

   - To copy a simple file

   cp filename.extension /destination-path

   - To copy all files with the same extension

   *cp *.extension /destination-path*

   Recursive copy

   - To copy all the contents inside a folder

   cp -r 'FolderName' /destination-path

2. **Moving and Renaming Files and Directories**

   mv oldfile.extension newfile.extension
   *mv *_samename.extension /destination-path*
   mv FolderName /destination-path

3. **Removing Files and Directories**

   rm filename.extension
   rm -d FolderName
   rm -r FolderName

Globbing and Wildcard

- ***What is Globbing?***
  - *Globbing is a powerful feature in Linux that allows you to perform pattern matching on file and directory names.*
- ***What are Wildcards?***
  - *Wildcards are special characters used in globbing patterns to represent one or more characters.*

- ***Asterisk (*)***
  Represents any number of characters (including none).
  *Example: *.txt matches all files with a .txt extension.*
- ***Question Mark (?)***
  Represents a single character.
  Example: file?.txt matches file1.txt, file2.txt, but not file10.txt.

Examples:
*ls *.txt: List all files with a .txt extension in the current directory.*
cp file?.txt destination/: Copy files like file1.txt, file2.txt to the "destination" directory.

Braces ({})
Allows you to specify multiple options within a single wildcard.
Example: file{1,2,3}.txt matches file1.txt, file2.txt, and file3.txt.

Square Brackets ([])
Matches a single character against a range or a list of characters.
Example: [0-9]file.txt matches 0file.txt, 1file.txt, ..., 9file.txt.

Table 4-1: Wildcards

Wildcard	Meaning
*	Matches any characters
?	Matches any single character
`[characters]`	Matches any character that is a member of the set *characters*
`[!characters]`	Matches any character that is not a member of the set *characters*
`[[:class:]]`	Matches any character that is a member of the specified *class*

Table 4-2 lists the most commonly used character classes:

Table 4-2: Commonly Used Character Classes

Character Class	Meaning
`[:alnum:]`	Matches any alphanumeric character
`[:alpha:]`	Matches any alphabetic character
`[:digit:]`	Matches any numeral
`[:lower:]`	Matches any lowercase letter
`[:upper:]`	Matches any uppercase letter

Create Links

ln command is used to create either hard or symbolic links. It is used in one of two ways:

```
ln file link
```

to create a hard link, and:

```
ln -s item link
```

to create a symbolic link where “item” is either a file or a directory.

Hard Links:

1. A hard link cannot reference a file outside its own file system. This means a link cannot reference a file that is not on the same disk partition as the link itself.

2. A hard link may not reference a directory.

Symbolic Links:

Symbolic links work by creating a special type of file that contains a text pointer to the referenced file or directory. In this regard, they operate in much the same way as a Windows shortcut though of course, they predate the Windows feature by many years ;-)

A file pointed to by a symbolic link, and the symbolic link itself are largely indistinguishable from one another.

For example, if you write something to the symbolic link, the referenced file is written to. However, when you delete a symbolic link, only the link is deleted, not the file itself. If the file is deleted before the symbolic link, the link will continue to exist, but will point to nothing. In this case, the link is said to be broken. In many implementations, the ls command will display broken links in a distinguishing color, such as red, to reveal their presence.

Understanding Symbolic Links and Hard Links in Linux

In Linux, links are used to create references to files or directories.

There are two types of links: symbolic links (symlinks) and hard links

What is a Symbolic Link?

- A symbolic link (or symlink) is a reference to a file or directory that points to its target by name.
- It is similar to a shortcut in Windows or a symlink in macOS.

Creating Symbolic Links

Use the ln -s command to create a symbolic link.

Syntax: **`ln -s <target> <link_name>`**

Example: `ln -s /path/to/target /path/to/link`

What is a Hard Link?

- A hard link is a reference to a file that points directly to the file's inode (data structure).
- Changes to the file are reflected in all hard links.

Features:

- ✓ Must be on the same file system.
- ✓ Cannot link to directories.
- ✓ No distinction between original and links.

Creating a Hard Link:

Use the ln command to create a hard link.

Syntax: **`ln <target> <link_name>`**

Example: `ln /path/to/target /path/to/link`

Use Cases:

Symbolic Links:

- ✓ *Used for creating shortcuts.*
- ✓ *Useful when linking to directories outside the current file system.*
- ✓ *Can link to non-existent targets.*

Hard Links:

- ✓ *Used for creating multiple references to the same data.*
- ✓ *Saves disk space when creating multiple references to the same file.*

Practical Exercises:

Exercise 1:

Go to Home directory (Your username directory)

cd ~

Create a folder named 'CopyingFolder', in the folder, create a text file copied.txt.

Now copy this file to 'Documents' folder.

In Documents folder, create 5 text file, one.txt, two.txt, … five.txt.

Copy all these files to 'CopyingFolder'.

Copy 'CopyingFolder' to Desktop.

In Documents folder, create 5 textfiles,

one_document.txt, two_document.txt, three_document.txt, …, five_document.txt

in each file write the number correspondent, 1 in one_document.txt, etc.

rename each file to French words, document_un, document_deux, document_trois, document_quatre, document_cinque.txt

Move these files to a new folder named MoveFolder

create a folder on Desktop named FrenchFolder

*Move all the files with just one command using * from MoveFolder to FrenchFolder.*

From Home Directory, remove a file name document_deux.txt in FrenchFolder.

Go to Documents folder and create a folder name EmptyFolder.

In EmptyFolder, create a textfile named deletefile.txt and another folder named DeleteFolder in EmptyFolder.

1. *What command to be used to delete DeleteFolder?*
2. *What command to be used to delete EmptyFolder?*
3. *What command to be used to delete all text file in FrenchFolder?*

Exercise 2:

Ex1: Using Wildcards and Globbing in Linux

Objective: In this lab, you will practice using wildcards and globbing in Linux to perform various file and directory operations.

Instructions:

1. Navigate to the Lab Directory:

Open a terminal.

Create a directory for this lab session, e.g., wildcard_lab.

Navigate to the lab directory using the cd command.

2. Create Sample Files:

Inside the lab directory, create several sample files with different extensions (e.g., .txt, .csv, .log) using the touch command.

Give them meaningful names.

3. List Files Using Wildcards:

Use the ls command with wildcards to list specific groups of files. Try the following:

List all files with the .txt extension.

List all files with the .log extension.

List files with a single-character name.

4. Copy Files Using Wildcards:

Create a new directory within the lab directory (e.g., backup).

Use the cp command with wildcards to copy files into the backup directory.

Copy all .txt files into the backup directory.

Copy files with a name of exactly three characters into the backup directory.

5. Deleting Files:

Use the rm command with wildcards to delete files.

Delete all files in the lab directory with a single-character name.

Delete all files in the backup directory.

6. Advanced Wildcards:

Create new sample files to experiment with advanced wildcards.

Use curly braces to create files like file1.txt, file2.txt, file3.txt, etc.

Use square brackets to list files like:

file1.txt, file2.txt, file3.txt, etc.

1file.txt, 2file.txt, 3file.txt, etc.

Exercise 3: **Advanced Wildcard Filtering**

Objective: In this exercise, you will practice more advanced wildcard filtering to select files based on specific criteria.

Instructions:

1. Create a directory named logs in your home directory.

   Inside the logs directory, create the following log files with different timestamps in their names (e.g., log20230101.txt, log20230102.txt, etc.):

   log20230101.txt

   log20230102.txt

   log20230103.txt

   log20230104.txt

   log20230105.txt

2. Use a wildcard pattern to list and display only the log files that have timestamps in January 2023 (files like log20230101.txt, log20230102.txt, etc.).

3. Use a wildcard pattern to list and display only the log files that have timestamps on days ending in an odd number (files like log20230101.txt, log20230103.txt, log20230105.txt, etc.).

Exercise 4: Links

- Create a directory named **link_lab**.
- Inside **link_lab,** create a text file named **original.txt** and add some content to it.
- Create a symbolic link to **original.txt** named **symlink.txt**.
- Create a hard link to **original.txt** named **hardlink.txt**.
- Verify that changes to **original.txt** are reflected in both **symlink.txt** and **hardlink.txt**.
- Try creating a symbolic link to a non-existent file or directory.