Date:
Roll no:

Experiment 5: Creating, Modifying, and Redirecting Files and Directories in Linux

Aim:

To demonstrate the creation, modification, and redirection of files and directories using text editors, file readers, output redirection, text processing tools, and various file and directory **operations in a Linux environment.**

Theory:

File and directory operations are fundamental to managing a Linux system. Understanding how to create, modify, and redirect files and directories is crucial for efficient system administration and scripting. This experiment covers basic and advanced file operations, including the use of text editors, file readers, output redirection, and text processing tools.

1. **Text Editors:**

   Text editors like Nano, vim, and Gedit are used to create and modify text files.

   ### 1. Nano

   `nano` is a simple, user-friendly text editor that's good for beginners. It operates in the terminal and provides on-screen shortcuts for common commands.

   ```
   nano example.txt
   ```

2. **File Readers:**

   Commands like cat, less, more, and head/tail are used to read and display file contents.

   Cat-Concatenate and display file content
   Syntax:cat[file]
   Example: cat myfile.txt

   Less –View file content page by page
   Syntax:less[file]
   Example: less largefile.txt

   head – View the first few lines of a file
   Syntax:head[file]
   Example: head myfile.txt

   Tail – View the last few lines of a file
   Syntax:tail[file]
   Example: tail myfile.txt

3. **Output Redirection:**

Output redirection using >, >>, and | is used to direct the output of commands to files or other commands.

1. `>` - Redirect Output to a File

The `>` operator is used to redirect the output of a command to a file, overwriting the file if it already exists.

```
echo "Hello, World!" > file.txt
```

- This command will write "Hello, World!" to `file.txt`. If `file.txt` already contains some data, it will be replaced with "Hello, World!".

2. `>>` - Append Output to a File

The `>>` operator appends the output of a command to the end of a file, rather than overwriting it.

```
echo "Another line" >> file.txt
```

- This command will add "Another line" to the end of `file.txt` without removing the existing content.

4. **Text Processing:**

Tools like grep, awk, sed, cut, and sort are used for processing and manipulating text data.

1. `grep` - Search for Patterns in Text

`grep` is used to search for specific patterns within files. It prints lines that match the given pattern.

```
grep "error" logfile.txt
```

- This command searches for the string "error" in `logfile.txt` and prints all lines containing that string.

2. `awk` - Pattern Scanning and Processing Language

`awk` is a versatile language for pattern scanning and processing. It can perform operations on fields within lines of text.

```
awk '{print $1}' file.txt
```

- This command prints the first field (column) of each line in `file.txt`. By default, `awk` uses whitespace as the field separator.

2. `awk` - Pattern Scanning and Processing Language

`awk` is a versatile language for pattern scanning and processing. It can perform operations on fields within lines of text.

Example:

bash Copy code

```
awk '{print $1}' file.txt
```

- This command prints the first field (column) of each line in `file.txt`. By default, `awk` uses whitespace as the field separator.

3. `sed` - Stream Editor for Filtering and Transforming Text

`sed` is a stream editor that can perform basic text transformations on an input stream (a file or input from a pipeline).

Example:

bash Copy code

```
sed 's/foo/bar/g' file.txt
```

- This command replaces all occurrences of "foo" with "bar" in `file.txt` and prints the result to standard output.

4. `cut` - Remove Sections from Each Line of Files

`cut` is used to remove sections from each line of files based on delimiter or byte positions.

Example:

```bash
cut -d, -f1 file.csv
```

- This command extracts the first field (column) from `file.csv`, using a comma as the delimiter.

5. `sort` - Sort Lines of Text Files

`sort` sorts lines of text files in a specified order.

Example:

```bash
sort file.txt
```

- This command sorts the lines in `file.txt` in ascending alphabetical order.

Example with numeric sorting:

```bash
sort -n numbers.txt
```

- This command sorts the lines in `numbers.txt` numerically.

Example to sort in reverse order:

```bash
sort -r file.txt
```

- This command sorts the lines in `file.txt` in reverse order.

5. **File and Directory Operations:**

Basic commands like touch, mkdir, rm, mv, and cp are used to create, delete, move, and copy files and directories.

1. `touch` - Create or Update Files

The `touch` command is used to create empty files or update the timestamp of existing files.

Example:

```bash
touch file.txt
```

- This command creates an empty file named `file.txt` if it does not already exist. If `file.txt` does exist, its modification timestamp is updated to the current time.

2. `mkdir` - Create Directories

The `mkdir` command is used to create new directories.

Example:

```bash
mkdir new_directory
```

- This command creates a directory named `new_directory`.

3. `rm` - Remove Files and Directories

The `rm` command is used to delete files or directories. Be cautious with `rm`, as deleted files are not easily recoverable.

Example to remove a file:

```bash
rm file.txt
```

- This command deletes the file named `file.txt`.

4. `mv` - Move or Rename Files and Directories

The `mv` command is used to move or rename files and directories.

Example to move a file:

```bash
mv file.txt /path/to/destination/
```

- This command moves `file.txt` to the directory `/path/to/destination/`.

5. `cp` - Copy Files and Directories

The `cp` command is used to copy files or directories from one location to another.

Example to copy a file:

```bash
cp file.txt /path/to/destination/
```

- This command copies `file.txt` to the directory `/path/to/destination/`.

Conclusion:

This experiment demonstrates the essential operations for creating, modifying, and redirecting files and directories in a Linux environment.