Day 8-File Operations

File Compression and Archiving in Linux

Linux provides powerful tools for managing disk space through **compression** and **archiving**. These tools help reduce file sizes and group multiple files into a single archive for storage or transfer.

Disk Usage

- `du` stands for **disk usage**
- Common options include:

```
du -sk <file/dir>      # Show size in kilobytes
du -sh <file/dir>      # Show size in human-readable format
ls -lh                 # Show file size in human-readable form along with other metadata
```

tar (Tape Archive)

- `tar` is used to **group multiple files/directories** into a single archive file (called a **tarball**)
- Common options include:

```
tar -cf archive.tar <files>        # Create a tarball
tar -tf archive.tar                # List contents of a tarball
tar -xf archive.tar                # Extract contents
tar -czf archive.tar.gz <files>    # Create and compress with gzip
tar -xzf archive.tar.gz            # Extract a gzipped tarball
```

- `-c` = create, `-f` = specify filename, `-t` = list contents, `-x` = extract, `-z` = use gzip compression

	Tool	Compress Command	Uncompress Command	File Extension
1	compress	`compress file`	`uncompress file.Z`	`.Z`
2	gzip	`gzip file`	`gunzip file.gz`	`.gz`
3	bzip2	`bzip2 file`	`bunzip2 file.bz2`	`.bz2`
4	xz	`xz file`	`unxz file.xz`	`.xz`

Viewing Compressed Files Without Extracting

You can read the contents of compressed files directly using the following tools:

```
zcat file.gz      # View a gzip-compressed file
bzcat file.bz2    # View a bzip2-compressed file
xzcat file.xz     # View an xz-compressed file
```

Searching for Files

Linux provides several tools to **locate files** and **search within their content**.

Using `locate`

- The `locate` command is used to quickly find files by name. For example:

- `locate hello.txt`

- It uses a prebuilt database called `mlocate.db`, so it's much faster than scanning the entire file system.
- To update the database (run as root):
  - `updatedb`

Using `find`

- The `find` command searches for files in real-time and does not depend on any database. For example:
  - `find /home/usera -name hello.txt`
  - `-name` matches the exact filename (case-sensitive)
  - Add `-iname` for case-insensitive file name matching

Using `grep` to Search Inside Files

- The `grep` command is used to search for text **within files**. For example:
  - `grep "text_to_search" file.txt`
- Useful flags include:

```
-i        # Case-insensitive search
-r        # Recursively search within directories
-v        # Print lines that do NOT match the pattern
-w        # Match whole words only
-A <N>    # Print N lines after a matching line
-B <N>    # Print N lines before a matching line
```

- For example:

```
grep -r "new_dir" /home/usera         # Recursively search for 'new_dir' inside /home/usera
grep -i "error" log.txt               # Case-insensitive search for 'error'
grep -v "test" file.txt               # Show lines that do not contain 'test'
grep -w "fail" file.txt               # Match whole word 'fail' only
grep -A1 "status" file.txt            # Print matching line and 1 line after
grep -B2 "failed" file.txt            # Print matching line and 2 lines before
```

I/O Redirection

When a Linux command is executed, three data streams are created:

- **Standard Input (stdin)** → Accepts input (default: keyboard)
- **Standard Output (stdout)** → Displays normal output (default: screen)
- **Standard Error (stderr)** → Displays error messages (default: screen)

Redirecting Output

- `>` redirects **stdout** to a file (overwrites existing content)
  - `ls > output.txt`
- `>>` appends **stdout** to a file (without overwriting)
  - `ls >> output.txt`
- `2>` redirects **stderr** to a file
  - cat file1.txt 2> error.txt
  - If the file does not exist, it will be created. If it exists, it will be overwritten unless `>>` is used.
- To **discard error messages**, redirect to `/dev/null`

- `cat file2.txt 2> /dev/null`
- `/dev/null` acts like a trash bin and anything written to it is ignored.

Pipes

- Pipes (`|`) connect the **stdout of one command** to the **stdin of another**
  - `command1 | command2`
  - `ls -l | grep ".txt"  #Lists only .txt files by passing ls -l output into grep.`

tee Command

- The `tee` command reads from stdin and writes to **stdout and a file** simultaneously. For example:
  - `ls -l | tee output.txt`
- To **append** instead of overwrite, use the `-a` flag. For example:
  - `ls -l | tee -a output.txt`