## Linux Essential Commands

1. cd - Change Directory

```
cd [directory]
```

- Examples:

```
cd /home/priyanka          # Go to absolute path
cd Documents               # Go to relative path
cd ..                      # Go up one directory
cd ../..                   # Go up two directories
cd ~                       # Go to current user's home directory
cd /                       # Go to root directory
cd -                       # Go to previous directory
```

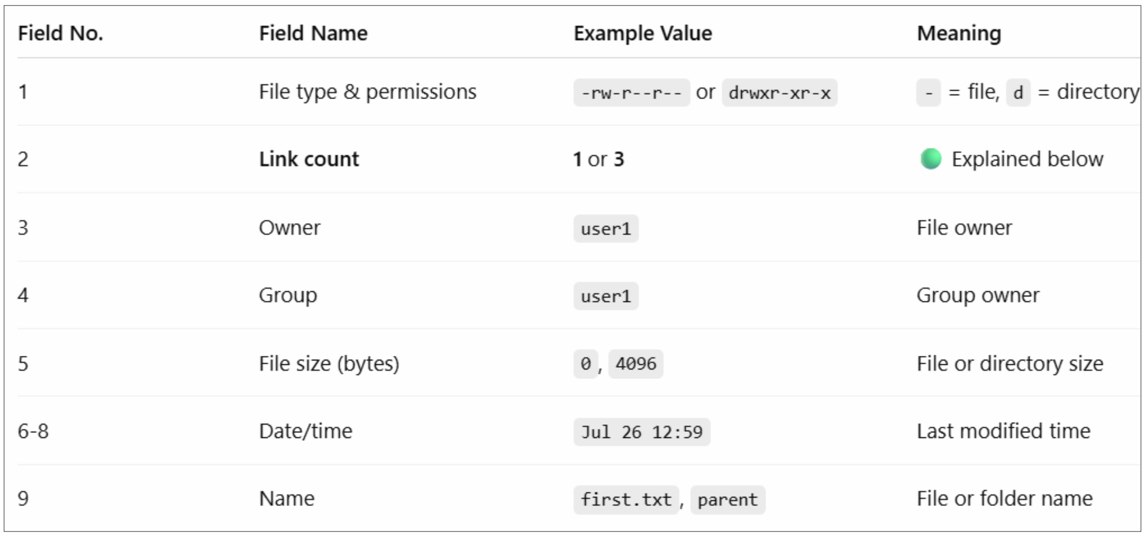
2. LS - List Directory Content

```
ls [options] [directory]
```

Examples:

```
ls                  # List files in current dir
ls -l               # Long listing with permissions, owners, size, date
ls -a               # Include hidden files (those starting with .)
ls -la              # Long listing including hidden files
ls -lh              # Human-readable sizes (KB, MB)
ls -R               # Recursive listing (include subdirectories)
ls /etc             # List files in /etc
ls -ltr             # Sort by modification time, reverse order
```

Example: **ls -l**



| Field No. | Field Name | Example Value | Meaning |
|---|---|---|---|
| 1 | File type & permissions | `-rw-r--r--` or `drwxr-xr-x` | `-` = file, `d` = directory |
| 2 | **Link count** | **1** or **3** | 🟢 Explained below |
| 3 | Owner | `user1` | File owner |
| 4 | Group | `user1` | Group owner |
| 5 | File size (bytes) | `0`, `4096` | File or directory size |
| 6-8 | Date/time | `Jul 26 12:59` | Last modified time |
| 9 | Name | `first.txt`, `parent` | File or folder name |

For a File: The **1** means: **1 hard link** to the file.

For a Directory: 2 + N

- 1 = The . (self) entry inside the directory
- 1 = The .. entry inside **each** of its **subdirectories**

3. rm - Remove Files or Directories

```
rm [options] [file or directory]
```

Examples:

```
rm file.txt              # Remove a file
rm -i file.txt           # Prompt before deletion
rm -f file.txt           # Force remove, no prompt
rm *.log                 # Remove all .log files
rm -r folder/            # Remove directory and its contents recursively
rm -rf /tmp/test/        # Force delete folder recursively (DANGEROUS!)
```

4. sudo - Run Commands as Root (Superuser)

```
sudo [command]
```

Examples:

5. To check what groups are created in linux
   $ groups

6. Users - Normal vs. Sudo (Admin)

```
sudo apt update          # Run system update with root privileges
sudo rm -rf /opt/demo    # Dangerous command with root rights
sudo nano /etc/hosts     # Edit system files
```

```
whoami
```

7. pwd - Print Working Directory

```
pwd
```

8. mkdir - Make directory

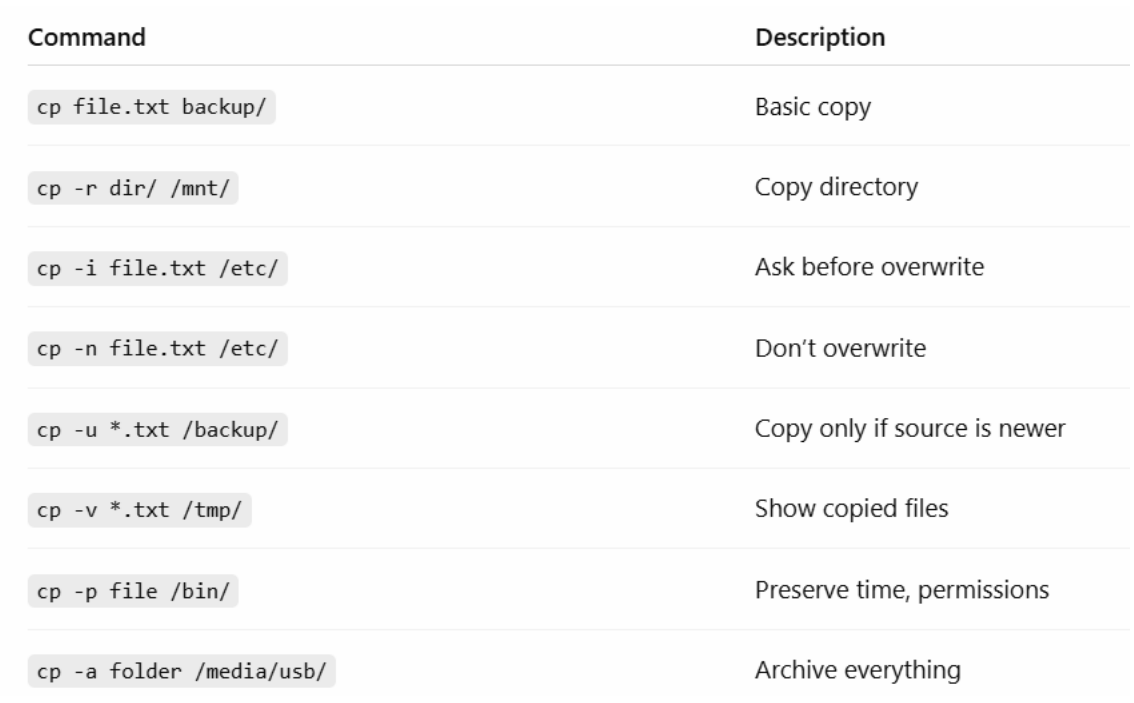
```
mkdir myfolder
mkdir -p parent/child     # Create nested directories
```

9. touch - Create file

```
touch file.txt
touch file{1..5}.txt      # file1.txt to file5.txt
```

10. cp - copy files

```
cp file1.txt backup/      # Copy file
cp -r dir1/ dir2/         # Copy folder recursively
```



| Command | Description |
|---|---|
| `cp file.txt backup/` | Basic copy |
| `cp -r dir/ /mnt/` | Copy directory |
| `cp -i file.txt /etc/` | Ask before overwrite |
| `cp -n file.txt /etc/` | Don’t overwrite |
| `cp -u *.txt /backup/` | Copy only if source is newer |
| `cp -v *.txt /tmp/` | Show copied files |
| `cp -p file /bin/` | Preserve time, permissions |
| `cp -a folder /media/usb/` | Archive everything |

11. mv - move or rename

```
mv file.txt newname.txt
mv file.txt /tmp/
```

Practice on cp command:

```
cp -u *.txt /backup/
```

- **cp** - copy files
- **-u** - **update mode**: only copy the file **if the source is newer than the destination** or if the file does **not exist** in the destination.
- It **compares modification timestamps** of the source and destination files.
- ***.txt** - wildcard to match all .txt files in the current directory
- **/backup/** - target directory where the files will be copied

```
mkdir backup
echo "original" > notes.txt
cp notes.txt backup/
```

```
echo "updated" >> notes.txt
```

```
 cp -u notes.txt backup/
```

This will **overwrite** the one in /backup/ **because it's older**. If you run again **without changes**, it **won't copy again**.

```
cp -p file /bin/
```

- **cp** - Copy a file
- **-p** - **Preserve file attributes**
- **file** - Source file
- **/bin/** - Destination directory (commonly requires sudo)

When you use the -p (preserve) option, it retains the following attributes from the original file:

| **Attribute** | **Description** |
|---|---|
| **Modification time** | The mtime timestamp when the file was last modified |
| **Access time** | The atime timestamp when the file was last read |
| **Ownership** | The file's **user** and **group** owner |
| **Permissions** | Read, write, execute permissions (e.g., rwx) |

**Without -p**

cp file /bin/

- The **new file** in /bin/ will:
    - Have the **current timestamp**
    - Be **owned by the user who ran the command**
    - Use the **default permissions** from the system

**Use Cases for -p**

- ✓ Backups where timestamps matter
- ✓ Deploying files with precise permissions (scripts, binaries)
- ✓ File auditing or compliance needs

As DevOps Engineer, daily basis common tasks like monitoring, troubleshooting, deployment, and system management.

**File and Directory Operations**

```
ls -ltrh                          # List files sorted by time, human-readable
cd /var/log                      # Navigate to log directory
mkdir -p /opt/app/logs          # Create directory and parents if not exist
cp config.yaml /etc/myapp/      # Copy config file to system path
mv app.jar /opt/app/            # Move artifact to deployment folder
rm -rf /tmp/build               # Clean up temporary files
```