AIX30011

Jaeyoung Chun | School of Applied Artificial Intelligence | Handong Global University

# Linux Shell Commands

## 1. `ls` (list): View file list

```
ls                # Display file list
ls -l             # Show detailed information
ls -t             # Display files sorted by modification time
ls -lh            # Show file sizes in a human-readable format
```

## 2. `pwd` (print working directory): Print current path

```
pwd               # Print current path
```

- `/` : root directory
- `/home` : home directory under root
- `/home/user` : user directory under home directory

## 3. `mkdir` (make directory): Create a new directory

```
mkdir dir1          # Make a directory named dir1 in the current directory
mkdir ./dir1        # Make a directory named dir1 in the current directory
mkdir ./path/dir1   # Make a directory named dir1 at a specific path
```

## 4. `cd` (change directory): Change working directory

```
cd ..           # Move to the parent directory
cd /            # Move to the root directory
cd ~            # Move to the home directory
cd -            # Move to the previous directory
cd dir2         # Move to dir2 in the current directory
cd ./dir1/dir2  # Move using a relative path
cd ~/dir1/      # Move to dir1 inside the home directory
cd /dir1/dir2   # Move using an absolute path
```

## 5. File creation and editing

```

```

```
touch text.txt          # Create an empty file
emacs -nw text.txt      # Edit file using Emacs
nano text.txt           # Edit file using Nano
vim text.txt            # Edit file using Vim
```

Emacs basic shortcuts

- Save: `Ctrl + x` then `Ctrl + s`
- Exit: `Ctrl + x` then `Ctrl + c`

Nano basic shortcuts

- `CTRL + X`: Exit file (prompts for saving changes before exiting)

## 6. `echo` and redirection

```
echo "hello"              # Print a string
echo "hello" > text.txt   # Save the string to a file (overwrite)
echo "hello" >> text.txt  # Append the string to a file
```

## 7. `cp` (copy): Copy files/directories

```
cp text.txt ./dir1             # Copy a file
cp text.txt ./dir1/change.txt  # Copy a file with a new name
cp ./*.pdf dir1/               # Copy all PDF files
```

## 8. `mv` (move/rename): Move files and rename them

```
mv oldname.txt newname.txt  # Rename a file
mv file.txt ./dir2/         # Move a file
```

## 9. `rm` (remove): Delete files

```
rm text.txt     # Delete a file
rm -r dir1      # Delete an entire directory
rmdir dir1      # Delete an empty directory
```

## 10. Display file contents

```

```

```
cat text.txt  # Show entire file contents
head text.txt # Show first part of the file (default: 10 lines)
tail text.txt # Show last part of the file (default: 10 lines)
more text.txt # Display file page by page (Enter: 1 line, Space: one page)
less text.txt # Similar to `more` but allows scrolling with arrow keys
```

## 11. `tar`: Compression and extraction

A `.tar` file, short for **Tape Archive**, is a file format used to bundle multiple files and directories into a single archive file. It is commonly used in Linux and Unix systems for packaging and distributing files. However, a `.tar` file itself is not compressed—it simply groups files together. Compression can be added using tools like `gzip` or `bzip2`, resulting in files like `.tar.gz` or `.tar.bz2`.

### Key Features of `.tar` Files:

1. **Archiving**: It combines multiple files and directories into one file, preserving the directory structure and metadata (e.g., permissions, timestamps).
2. **No Compression**: By default, `.tar` does not reduce file size; it only groups files.
3. **Compression Add-ons**: Tools like `gzip` or `bzip2` are often used alongside `tar` to compress the archive.

### Common `tar` Commands:

- **Create an archive**:

  ```
  tar -cvf archive.tar file1 file2 directory/
  ```

  - `-c`: Create a new archive.
  - `-v`: Verbose mode (shows progress).
  - `-f`: Specify the archive file name.

- **Extract an archive**:

  ```
  tar -xvf archive.tar
  ```

  - `-x`: Extract files from the archive.

- **Compress with gzip**:

  ```
  tar -zcvf archive.tar.gz file1 file2
  ```

  - `-z`: Compress with gzip.

- **Extract a gzip-compressed archive**:

```
tar -xvzf archive.tar.gz
```

- `-z`: Decompress with gzip during extraction.

## 12. `gzip`, `gunzip`

```
gzip test.txt      # Compress a file
zcat test.txt.gz   # Display contents of a compressed file
gunzip test.txt.gz # Decompress a file
```

## 13. `grep`: Search for strings

```
grep "Hello" test.txt      # Find lines containing the string
grep -ni "hello" text.txt  # Search with line numbers (case insensitive)
```

## 14. Pipes (`|`)

```
head abc.txt | grep xyz       # Search for "xyz" in the first 10 lines
zcat abc.txt.gz | grep "xyz"  # Search within a compressed file
```

## 15. `chmod` (change mode)

The `chmod` command is used to change the permissions of a file or directory. Permissions are represented as a combination of read (`r`), write (`w`), and execute (`x`) for three categories: owner, group, and others.

### Common Permission Values:

- `r` (read): Allows reading the file or listing the directory contents.
- `w` (write): Allows modifying the file or adding/removing files in a directory.
- `x` (execute): Allows executing the file (if it's a script or program) or accessing the directory.

### Numeric Representation:

Permissions can also be represented numerically:

- `4`: Read (`r`)
- `2`: Write (`w`)
- `1`: Execute (`x`)

The sum of these values determines the permission for each category. For example:

- `7` = `4 (read) + 2 (write) + 1 (execute)` = Full permissions
- `5` = `4 (read) + 1 (execute)` = Read and execute

- `6` = `4 (read) + 2 (write)` = Read and write

Examples:

```
chmod 777 text.txt     # Grant all permissions
                       # (read, write, execute) to everyone
chmod 755 script.sh    # Full permissions for owner,
                       # read and execute for group and others
chmod 644 file.txt     # Read and write for owner,
                       # read-only for group and others
chmod +x script.sh     # Add execute permission for everyone
chmod -w file.txt      # Remove write permission for everyone
chmod u+x script.sh    # Add execute permission for the owner only
chmod g-w file.txt     # Remove write permission for the group
chmod o+r file.txt     # Add read permission for others
```

Symbolic Representation:

Instead of numeric values, you can use symbolic notation:

- `u`: Owner
- `g`: Group
- `o`: Others
- `a`: All (owner, group, and others)

Use Cases:

- Granting execute permissions to a script:

  ```
  chmod +x script.sh
  ```

- Restricting write access to a file:

  ```
  chmod -w file.txt
  ```

- Setting specific permissions for a directory:

  ```
  chmod 755 /path/to/directory
  ```

## 16. Search files and directories

The `find` command is a powerful utility for searching files and directories based on various criteria such as name, type, size, permissions, and more.

Syntax:

```
find [path] [expression]
```

Common Options:

- `-name`: Search for files or directories by name (case-sensitive).
- `-iname`: Search for files or directories by name (case-insensitive).
- `-type`: Specify the type of file to search for:
  - `f`: Regular file
  - `d`: Directory
- `-size`: Search for files based on size.
  - `+`: Greater than (e.g., `+1M` for files larger than 1 MB).
  - `-`: Less than (e.g., `-500k` for files smaller than 500 KB).
- `-perm`: Search for files with specific permissions.
- `-exec`: Execute a command on the matching files.

Examples:

- Search for a file named "Handong" in the current directory:

  ```
  find . -name "Handong"
  ```

- Search for a file named "handong" (case-insensitive):

  ```
  find . -iname "handong"
  ```

- Search for all directories named "config":

  ```
  find /path/to/search -type d -name "config"
  ```

- Search for files larger than 10 MB:

  ```
  find /path/to/search -type f -size +10M
  ```

- Search for files with `777` permissions:

  ```
  find /path/to/search -perm 777
  ```

- Delete all `.tmp` files in a directory:

```
find /path/to/search -type f -name "*.tmp" -exec rm {} \;
```

- Search for files modified in the last 7 days:

```
find /path/to/search -type f -mtime -7
```

- Search for files accessed more than 30 days ago:

```
find /path/to/search -type f -atime +30
```

- Use `-maxdepth` to limit the search depth:

```
find . -maxdepth 2 -name "*.txt"
```

- Use `-mindepth` to skip shallow levels:

```
find . -mindepth 3 -name "*.txt"
```

## 17. Process and system monitoring

```
top             # Real-time system resource monitoring
htop            # Enhanced `top` command
ps aux          # List running processes
kill 1234       # Terminate a process whose ID is 1234
```

## 18. Network-related commands

### 18.1 `ssh` (Secure Shell)

The `ssh` command is used to securely log in to a remote machine over a network.

```
ssh user@192.168.1.100
```

- `user`: The username on the remote machine.
- `192.168.1.100`: The IP address or hostname of the remote machine.

### 18.2 `scp` (Secure Copy)

The `scp` command is used to securely copy files between a local and a remote machine or between two remote machines.

```
scp file.txt user@remote:/path/
```

- `file.txt`: The file to be copied.
- `user@remote`: The username and hostname/IP address of the remote machine.
- `/path/`: The destination directory on the remote machine.

### 18.3 `wget` (Web Get)

The `wget` command is used to download files from the web.

```
wget https://example.com/file.zip
```

- `https://example.com/file.zip`: The URL of the file to be downloaded.

### 18.4 `curl` (Client URL)

The `curl` command is a versatile tool for transferring data from or to a server. It supports various protocols, including HTTP, HTTPS, FTP, and more.

**Download a file with the same name:**

```
curl -O https://example.com/file.zip
```

- `-O`: Save the file with its original name from the URL.

**Download a file and save it with a custom name:**

```
curl -o custom_name.zip https://example.com/file.zip
```

- `-o custom_name.zip`: Save the file with the specified name (`custom_name.zip`).

## 19. Disk and file system management

```
df -h         # Check disk usage
du -sh /path/ # Check storage usage of a specific directory
mount         # Mount a disk
```

## 20. Ownership Management

The `chown` command is used to change the ownership of files and directories. Ownership is divided into two categories: the user (owner) and the group.

Syntax:

```
chown [OPTIONS] USER:GROUP FILE
```

Examples:

- Change the owner of a file:

  ```
  chown user file.txt
  ```

- Change the owner and group of a file:

  ```
  chown user:group file.txt
  ```

- Change ownership recursively for a directory and its contents:

  ```
  chown -R user:group /path/to/directory
  ```

Options:

- `-R`: Apply changes recursively to all files and subdirectories.
- `--reference=RFILE`: Use the ownership of `RFILE` as a reference for the target file.

Use Cases:

- Assign a new owner to a file:

  ```
  chown alice report.doc
  ```

- Change ownership of all files in a directory:

  ```
  chown -R bob:staff /project/files
  ```

Notes:

- Only the root user or a user with sufficient privileges can change ownership.
- Use `ls -l` to verify ownership changes:

```
ls -l file.txt
```

- Combine with `sudo` if necessary:

```
sudo chown user:group file.txt
```

## 21. `rsync`: Remote File Synchronization

The `rsync` command is a powerful utility for synchronizing files and directories between local and remote systems. It is commonly used for backups and mirroring.

**Syntax:**

```
rsync [OPTIONS] SOURCE DESTINATION
```

**Examples:**

- Sync files from a local directory to a remote server:

```
rsync -avz /local/ user@remote:/backup/
```

  - `-a`: Archive mode (preserves symbolic links, permissions, timestamps, etc.)
  - `-v`: Verbose mode (displays progress)
  - `-z`: Compress data during transfer

- Sync files from a remote server to a local directory:

```
rsync -avz user@remote:/backup/ /local/
```

- Perform a dry run to preview changes without making them:

```
rsync -avz --dry-run /local/ user@remote:/backup/
```

**Use Cases:**

- Backing up files to a remote server.

- Synchronizing directories between systems.
- Efficiently transferring large datasets.

Notes:

- Use `--delete` to remove files in the destination that are not present in the source:

```
rsync -avz --delete /local/ user@remote:/backup/
```

- Combine with `ssh` for secure transfers:

```
rsync -e ssh -avz /local/ user@remote:/backup/
```

## 22. Other Utilities

```
clear           # Clear terminal screen
stat text.txt   # Check file attributes
wc text.txt     # Count words and lines
file text.txt   # Check file type
history         # View command history
man ls          # View command manual
```

## 23. Quick Reference

File/Directory Management

| Command | Function | Example Usage |
|---|---|---|
| `ls` | Display the list of files and folders in the current directory | `ls -l` |
| `pwd` | Print the current working directory (path) | `pwd` |
| `cd` | Navigate between directories (change path) | `cd /home/user` |
| `mkdir` | Create a new directory | `mkdir new_folder` |
| `rmdir` | Delete an empty directory | `rmdir empty_directory` |
| `mv` | Move or rename a file/directory | `mv old.txt new.txt` |
| `cp` | Copy a file/directory | `cp file.txt /backup/` |
| `rm` | Delete a file or directory (use `-r` option for directories) | `rm -r folder` |
| `touch` | Create an empty file or update its timestamp | `touch file.txt` |

| Command | Function | Example Usage |
| --- | --- | --- |
| `ln` | Create a symbolic (or hard) link | `ln -s target.txt link.txt` |
| `file` | Check the file type (e.g., text, binary) | `file example.pdf` |

### Viewing and Comparing File Contents

| Command | Function | Example Usage |
| --- | --- | --- |
| `cat` | Display the entire content of a file | `cat file.txt` |
| `less` | View file content page by page and scroll | `less /var/log/syslog` |
| `head` | Display the first specified number of lines of a file | `head -n 10 file.txt` |
| `tail` | Display the last specified number of lines of a file | `tail -n 10 file.txt` |
| `diff` | Compare two files line by line | `diff file1.txt file2.txt` |
| `cmp` | Compare two files byte by byte | `cmp file1.txt file2.txt` |
| `comm` | Display common and unique lines between two sorted files | `comm file1.txt file2.txt` |
| `sort` | Sort and display the content of a file | `sort file.txt` |

### Text and Output Processing

| Command | Function | Example Usage |
| --- | --- | --- |
| `echo` | Print text or strings | `echo "Hello World"` |
| `export` | Set environment variables (for the current shell) | `export PATH=$PATH:/new/path` |
| `alias` | Create shortcuts (aliases) for frequently used commands | `alias ll='ls -l'` |

### Compression/Archiving

| Command | Function | Example Usage |
| --- | --- | --- |
| `tar` | Create or extract archives of files/directories | `tar -czvf archive.tar.gz folder/` |
| `zip` | Compress files (ZIP format) | `zip archive.zip file1 file2` |
| `unzip` | Extract ZIP files | `unzip archive.zip` |

### System Information

| Command | Function | Example Usage |
| --- | --- | --- |

| Command | Function | Example Usage |
| --- | --- | --- |
| `uname` | Display system and kernel information | `uname -a` |
| `whoami` | Display the current logged-in username | `whoami` |
| `man` | View the manual page for a command | `man ls` |
| `free` | Check system memory usage and availability | `free -h` |
| `history` | View the history of entered commands | `history` |

### Process Management

| Command | Function | Example Usage |
| --- | --- | --- |
| `top` | Monitor system resources and processes in real-time | `top` |
| `htop` | Interactive process monitoring (alternative to `top`) | `htop` |
| `ps` | Display a list of currently running processes | `ps aux` |
| `kill` | Terminate a process by its ID (or use `killall`) | `kill 1234` or `killall firefox` |
| `jobs` | View background/foreground jobs in the current shell | `jobs` |
| `time` | Measure the execution time of a command | `time ls -l` |

### Remote/Network

| Command | Function | Example Usage |
| --- | --- | --- |
| `ssh` | Securely connect to a remote system (SSH) | `ssh user@192.168.1.100` |
| `scp` | Securely copy files between remote systems using SSH | `scp file.txt user@remote:/path/` |
| `curl` | Transfer data to/from a URL (e.g., API testing) | `curl -O https://example.com/file.zip` |
| `wget` | Download files directly using HTTP/HTTPS/FTP protocols | `wget https://example.com/file.zip` |

### Disk/File System

| Command | Function | Example Usage |
| --- | --- | --- |
| `df` | Check disk usage of file systems (commonly with `-h` option) | `df -h` |

| Command | Function | Example Usage |
| --- | --- | --- |
| `mount` | Mount a file system or device | `mount /dev/sdb1 /mnt/usb` |
| `du` | Summarize disk usage of directories and files | `du -sh /path/to/directory` |

### Permissions/Ownership Management

| Command | Function | Example Usage |
| --- | --- | --- |
| `chmod` | Change file/directory permissions | `chmod 755 script.sh` |
| `chown` | Change file/directory owner and group | `sudo chown user:group file.txt` |

### Miscellaneous Utilities

| Command | Function | Example Usage |
| --- | --- | --- |
| `clear` | Clear the terminal screen | `clear` |
| `whereis` | Locate the binary, source, and manual for a command | `whereis gcc` |
| `whatis` | Display a brief description of a command | `whatis ls` |

### Package/Permission Management

| Command | Function | Example Usage |
| --- | --- | --- |
| `apt` | Package manager for Debian-based systems (install/update, etc.) | `sudo apt update` |
| `sudo` | Execute commands with superuser (admin) privileges | `sudo apt upgrade` |

### Additional Commands

| Command | Function | Example Usage |
| --- | --- | --- |
| `hostname` | Display or set the system hostname | `hostname` or `hostnamectl set-hostname newname` |
| `cal` | Display a calendar in the terminal | `cal` |
| `find` | Search for files/directories matching criteria in a path | `find . -name "file.txt"` |
| `rsync` | Synchronize and back up files/directories | `rsync -avz /local/path/ user@remote:/remote/path/` |

## Additional Tutorials

Videos

- 생활코딩 - Linux @ inflearn
- 생활코딩 - Linux @ YouTube

Documents

- Command Line for Beginners - Ubuntu
- How to Learn Linux Terminal as a Beginner - freeCodeCamp
- Basic Linux Commands - GeeksforGeeks
- Linux Commands - DigitalOcean
- Linux Commands - Hostinger
- Linux Command Line Tutorial - freeCodeCamp
- Learning the Shell - LinuxCommand.org