Linux Notes

Working With Directories: `'pwd' & 'clear'`

Linux provides a `CLI` (Command Line Interface) to communicate with the operating system. The
`CLI` excels at tasks that cannot be performed through the GUI.

Common Linux Commands

Command	Explanation
`pwd`	Displays the **current working directory** of the terminal.
`echo`	Writes its **arguments** to standard output (prints text or variables).
`su`	Switches to the **root user**, enabling **superuser permissions** for subsequent commands.
`su username`	Switches to a **specified user account**, running a new shell as that user.
`sudo`	Executes a **single command** with **root/superuser privileges** (prompts for your password).
`clear`	**Clears the terminal screen**, scrolling previous output up; no data is deleted.

Note: The root directory is the top-level directory in the Linux filesystem hierarchy.

Working With Directories: `'ls'` Command

$ ls [options] [file | Directory]

Command	Purpose
`ls`	Basic listing of current directory.
`ls /var/log`	List contents of the `/var/log` directory.
`ls -l`	Detailed view: permissions, owner, size, timestamp.
`ls -a`	Show hidden files (`.bashrc`, `.git/`, etc.).
`ls --author`	Show file author alongside long listing.
`ls -S`	Sort results by file size (largest first).
`ls -laS --author /etc`	Comprehensive listing of `/etc`, including hidden files, sorted by size.

Command	Purpose
`ls *.html`	Display only files ending with `.html`.
`ls -l > directory.txt`	Save a long-format listing of the current directory into `directory.txt`.

Working With Directories: `'cd’`

$ `cd` [directory]

Command	Effect/Explanation
`cd`	Changes to the home directory of the current user.
`cd -`	Switches to the **previous working directory** (last directory you were in).
`cd /`	Changes to the **root directory** (`/`), which is the top of the filesystem hierarchy.
`cd ..`	Moves up **one directory level** to the parent of the current directory.
`cd 'xx yy'`	Changes to a directory named `xx yy` (with a space in its name). Quotation marks handle the space.

`cd          #` /home/username
`cd -         #`
switches back to previous directory
`cd /          # /`
`cd ..         #`
up one level
`cd 'My Folder'    #`
switches into a directory named 'My Folder'

Working With Files: `'cat'` Command

$ cat [options] filet [file21

Command	What It Does
`cat file.txt`	Display file contents
`cat file1 file2`	Concatenate/display multiple files
`cat -b file.txt`	Number non-blank lines
`cat -n file.txt`	Number all lines (including blank lines)

Command	What It Does
`cat -s file.txt`	Squeeze multiple consecutive blank lines into one
`cat -E file.txt`	Show `$` at the end of each line

Command	Purpose
`cat > file.txt`	Create or overwrite `file.txt` with terminal input
`cat >> file.txt`	Append terminal input to the end of `file.txt`
`cat file.txt`	Display the contents of `file.txt`

$ `cat file1.txt`

The “>” flag can be used to create a new file and enter text contents from the terminal

$

`cat file1, fxt`

The “>>” flag can be used to append text contents to an existing file from the terminal

Working With `'grep'` Command

We use the `'grep'` command to search for a particular string/ word in a text file.

This is similar to

`"Ctrl+F",` but executed via a `CLI`

Command	Description
`grep options file1.txt`	Searches `file1.txt` for lines containing the exact string **"options"** (case-sensitive).
`grep -i options file1.txt`	Searches case-insensitively, so lines with **"Options"**, **"OPTIONS"**, etc., will match.
`grep -n options file1.txt`	Displays matching lines along with their **line numbers** in `file1.txt`.
`grep -v options file1.txt`	Shows only the lines **that do not contain** the string "options".
`grep -c options file1.txt`	Returns the **number of lines** in which "options" appears (counts matching lines only).

Option	Description	Example Command
None	Simple match (case-sensitive)	`grep options file1.txt`
`-i`	Case-insensitive search	`grep -i options file1.txt`
`-n`	Show line numbers	`grep -n options file1.txt`

Option	Description	Example Command
`-v`	Invert match (show lines without the pattern)	`grep -v options file1.txt`
`-c`	Count matching lines	`grep -c options file1.txt`

Working With `'sort'` Command

We use the `"sort"` command to sort the results of a search either alphabetically or numerically. Files, file contents and directories can be sorted.

Command	What It Does
`sort filet.txt`	Sorts `filet.txt` alphabetically (A to Z).
`sort File1.txt File2.txt`	Combines lines from `File1.txt` and `File2.txt`, sorts them as one list alphabetically.
`sort -r file1.txt`	Sorts `file1.txt` in reverse order (Z to A).
`sort -f filet.txt`	Sorts `filet.txt` alphabetically, **case-insensitive** (so "Apple" and "apple" are equal).
`sort -n file1.txt`	Sorts `file1.txt` **numerically** (e.g., 2 comes before 10).

Working With `'|'` Command

The `'|'` command a.k.a `'pipe'` command is used to output the result of one command as input to another command.
Tare used to perform two operations in the same command

Command	Description
`grep dh File1.txt File2.txt`	Search for `'dh'` in both files; outputs matches as they appear.
`grep dh File1.txt File2.txt | sort`	Same search, but results are sorted alphabetically.
`grep dh File1.txt File2.txt | sort -r`	Same search, but results are sorted in reverse order (Z–A).

Working With Files & Directories: `'cp'` Command

$ `cp` [options] source destination

The `cp` (copy) command in Linux is used to copy files and directories from one location to another. Below is a detailed guide to its most commonly used options, presented in a clear, structured format.

Command/Option	Description
`cp file1.txt file2.txt`	Copies `file1.txt` to `file2.txt` (creates/replaces `file2.txt`).
`cp -i`	**Interactive:** Prompts before overwriting files at the destination.
`cp -n`	**No-clobber:** Does *not* overwrite existing files at the destination.
`cp -u`	**Update:** Copies only when the source file is newer than the destination or destination is missing.
`cp -R` or `cp -r`	**Recursive:** Copies directories (and their contents), including hidden files.
`cp -v`	**Verbose:** Displays detailed messages about what files are being copied.

Command/Option	What It Does
`cp`	Basic file or directory copy
`cp -i`	Asks before overwriting files
`cp -n`	Skips overwriting if destination exists
`cp -u`	Copies only if source is newer or missing
`cp -R`	Copies directories and everything within
`cp -v`	Prints what is being copied (verbose output)

Working With Files & Directories: `'mv'` Command

$ `mv` [options] source destination

The `mv` (move) command in Linux is used to move or rename files and directories. Below is a detailed, organized summary of the command along with its most common options.

Command/Option	Description
`mv file1.txt /tmp/`	Moves `file1.txt` to the `/tmp/` directory.
`mv -i`	**Interactive mode:** Prompts before overwriting an existing file at the destination.
`mv -u`	**Update:** Moves files only when the source file is newer than the destination or the destination missing.
`mv -v`	**Verbose:** Displays the source and destination file paths as each move or rename happens.

Option	What It Does
(none)	Move or rename files and directories

Option	What It Does
`-i`	Prompt before overwriting
`-u`	Move only if source is newer or missing at dest.
`-v`	Print each move operation

Working With Directories: `"mkdir"` Command

$ `mkdir` directory-path

Common Usage

Command	Description
`mkdir projects`	Creates a directory named `projects` in the current directory.
`mkdir test1 test2`	Creates multiple directories `test1` and `test2` at once.

Useful Options

Option	Purpose	Example Usage
`-p`	**Creates parent directories as needed.** If the path contains non-existent parents, all will be created.	`mkdir -p projects/2025/july`
`-v`	**Verbose mode.** Prints a message for each directory created.	`mkdir -v logs`
`-m MODE`	**Set permissions** for the new directory immediately.	`mkdir -m 755 scripts`

Command Example	What It Does
`mkdir newdir`	Create a directory named `newdir`
`mkdir dir1 dir2 dir3`	Create three directories at once
`mkdir -p projects/2025/scripts`	Make nested directories, creating any missing parent
`mkdir -m 755 shared`	Set directory permissions at creation
`mkdir -v myfolder`	See terminal messages confirming directory creation

Working With Files & Directories: `'rm' & 'rmdir'`

Managing files and directories often involves deleting them when they are no longer needed. Linux provides two main commands for this task: `rm` and `rmdir`. Here's a clear, structured overview.

Common Usage

Command	Description
`rm filename.txt`	Deletes `filename.txt` in the current directory.
`rm file1 file2`	Deletes multiple files at once.

Useful Options

Option	Purpose	Example Usage
`-i`	Prompts before every removal.	`rm -i important.txt`
`-f`	Forces removal without prompts (ignores missing files).	`rm -f junk.txt`
`-r` / `-R`	Recursively deletes directories and their contents.	`rm -r old_folder`
`-v`	Verbose output, lists each deleted file.	`rm -v temp.txt`

Common Usage

Command	Description
`rmdir olddir`	Deletes the directory `olddir` if it is empty.
`rmdir dir1 dir2`	Deletes multiple empty directories at once.

Key Differences

Feature	`rm`	`rmdir`
Deletes files?	Yes	No
Deletes directory	Yes (`-r` / `-rf` required)	Only if empty
Prompts?	No, unless `-i` is used	No, unless error
Recursion?	Yes with `-r`	No

Working With User Permissions: `'r', 'w' & 'x'`

In Linux, permissions control **who** can do **what** with a file or folder. The three key permissions are:

Symbol	Name	What It Lets You Do
`r`	Read	Look at or open the file or see inside a folder
`w`	Write	Change or delete the file, or add/delete inside a folder
`x`	Execute	Run the file as a program, or enter the folder

```
-rw-r--r-- 1 username groupname 1024 Jul 19 17:10 filename.txt
```

`chmod`: To change the access permissions of files and directories

`chown`: To change the owner of files and directories

`chgrp`: To change the group ownership of file and directories

Command	Explanation
`chmod g+wx` filename	Add write & execute for group
`chmod u=rwx,o-wx` filename	Set owner to `rwx`; remove `w & x` from others
`chown` username filename	Change file owner
`chown` username:group name filename	Change owner **and** group
`chgrp` group name filename	Change file's group only

Working With Linux Repositories

Stable versions of most software's will already be available in Linux repositories. Command to install them:

$

`sudo yum install package-name` // For RHEL based systems

$

`sudo apt-get install package-name:` // For Debian based systems

$

`sudo dnf install package-name` // For Fedora based systems.

Example for Installing `Nginx`

`sudo apt update`

`sudo apt install nginx` # install from enabled repository

`sudo apt remove nginx` # uninstall

`sudo apt upgrade` # upgrade all packages

Working With Tar Files

When you download a package from the internet, the downloaded file comes in compressed form.
Commands to decompress and compress files in Linux:

`gzip:` To compress files with `.gz` format

`gunzip:` To decompress `.gz` files

`tar:` To compress and decompress files with tar format

Command Syntax	Purpose
`gzip file.txt`	Compresses `file.txt` → produces `file.txt.gz`
`gunzip file.txt.gz`	Decompresses `file.txt.gz` → restores `file.txt`
`gzip -d file.txt.gz`	Same as `gunzip file.txt.gz`
`gzip -l file.txt.gz`	Lists compressed file information (uncompressed size, ratio)

Environment Variables

Environment variables control the behavior of the software packages installed in Linux.
The path where the packages have been installed will be specified in environment variables.

Command	Explanation
`printenv`	Prints all environment variables and their values.
`echo $HOME`	Displays the **home directory** path of the current user (environment variable `HOME`).
`echo $PATH`	Shows the **colon-separated list** of directories the shell searches for executable commands (`PATH`).

Command	Explanation
`echo $HOSTNAME`	Prints the system's **`hostname`** (environment variable `HOSTNAME`).
`echo $USER`	Displays the **username** of the current user (environment variable `USER`).
`echo $LANG`	Prints the **locale/language** setting in use (environment variable `LANG`).
`echo $BASH_VERSION`	Shows the **version** of the Bash shell currently running (`BASH_VERSION`).

```
# List all environment variables
printenv

# Get the home directory of the current user
echo $HOME
# Example output: /home/alice# View the command search path

echo $PATH
# Example output: /usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin#

Display the system hostname
echo $HOSTNAME
# Example output: my-server# Print the current username

echo $USER
# Example output: alice# Show the language/locale setting

echo $LANG
# Example output: en_US.UTF-8# Check the Bash shell version

echo $BASH_VERSION
# Example output: 5.1.8(1)-release
```

Regular Expressions

Using Regular Expressions `(RegEx)` with `grep` in Linux

Regular Expressions `(RegEx)` are **powerful patterns** that let you match complex text strings. Combined with the `grep` family of tools, they enable you to **search, filter, and extract** data from files with precision.

Metacharacter	Meaning
`.`	Any single character (except newline)
`*`	Previous token **zero or more** times
`+`	Previous token **one or more** times
`?`	Previous token **zero or one** time
`^`	**Start** of line
`$`	**End** of line
`[]`	Any **one** character inside brackets
`[^ ]`	Any **one** character *not* inside brackets
`()`	Grouping; capture subpatterns
`\`	Escape metacharacter to literal

Creating Users

Command	Explanation
`sudo useradd user-name`	Create a new user account named `user-name`.
`sudo passwd user-name`	Set or update the password for `user-name`.
`sudo userdel user-name`	Delete the user account `user-name` (does not remove home dir).
`sudo groupadd group-name`	Create a new group named `group-name`.
`sudo groupdel group-name`	Delete the group named `group-name`.
`sudo usermod -g group-name user-name`	Set `group-name` as the primary group for `user-name`.

Note:

- When deleting a user with `userdel`, use `r` (e.g., `sudo userdel -r user-name`) to also remove their home directory and mail spool.
- To add a user to a secondary (supplementary) group, use `sudo usermod -aG group-name user-name`.
- Ensure you have `sudo` privileges to run these commands.