**Linux**

Linux is a powerful and flexible open-source operating system. Its kernel, which is the core of the system, plays a crucial role in managing system resources and interacting with hardware. With the ability to support many hardware architectures and peripherals, Linux is widely used across a variety of devices and platforms. Its open-source nature makes it highly customizable and adaptable for developers and enthusiasts alike.

**The date Command:**

The date command in Linux is used to display the current date and time. You can format it to show specific parts of the date, like year, month, day, time, etc.

$date

Sun Feb 2 20:54:14 PM JST 2025).

**3. Formatting the Date:**

You can use date with format specifiers to customize the output. For example:

**$date "+%T"**

This command will output the time in the format **HH:MM:SS**.

**Common Format Specifiers:**

Here’s a list of useful specifiers for formatting:

* **+%T**: Time in the format HH:MM:SS (e.g., 14:32:20).
* **+%D**: Date in the format MM/DD/YY (e.g., 02/02/25).
* **+%H**: Hour in 24-hour format (e.g., 14).
* **+%M**: Minute (e.g., 32).
* **+%S**: Second (e.g., 20).
* **+%d**: Day of the month (e.g., 02).
* **+%m**: Month number (e.g., 02 for February).
* **+%y**: Two-digit year (e.g., 25 for 2025).
* **+%Y**: Four-digit year (e.g., 2025).
* **+%a**: Abbreviated weekday name (e.g., Sun for Sunday).
* **+%A**: Full weekday name (e.g., Sunday).
* **+%b**: Abbreviated month name (e.g., Feb for February).
* **+%B**: Full month name (e.g., February).

**4. Additional Help - man Command:**

**man date**

**Basic Directory Commands:**

**mkdir example**: Creates a new directory called **example**.

**cd example**: Changes into the **example** directory.

**cd ..**: Moves you up one directory level (to the parent directory).

**pwd**: Displays the full path of your current directory.

**rm -r example**: Removes the **example** directory and all its contents.

**Basic File Operations**:

**cat demo.txt**: Displays the contents of **demo.txt**.

**cat >> demo.txt**: Appends text to **demo.txt** without overwriting the current contents.

**rm demo.txt**: Deletes the file **demo.txt**.

**touch demo.txt**: Creates an empty file named **demo.txt**.

**Listing Files and Directories:**

**ls**: Lists files and directories.

* **ls -l**: Detailed listing.
* **ls -a**: Lists all files (including hidden).
* **ls -t**: Sort by modification time.
* **ls -al**: All files with detailed information.

**Copying Files**:

**cp**: Copies files or directories.

* **cp demo.txt /path/**: Copy file to a directory.
* **cp -r demo\_folder /path/**: Copy directory recursively.
* **cp -v demo.txt /path/**: Copy with verbose output.
* **cp -i**: Prompt before overwriting

**Moving and Renaming Files**

**mv**: Move or rename files and directories.

* **mv demo.txt demo\_link.txt**: Rename **demo.txt** to **demo\_link.txt**.
* **mv demo.txt sample/**: Move **demo.txt** to the **sample** directory.

**ln**: Create links to files.

* **ln demo.txt demo\_link.txt**: Create a hard link.
* **ln -s demo.txt demo\_symlink.txt**: Create a symbolic (soft) link.

**Word and Line Count:**

**wc**: Word count command.

* **wc demo.txt**: Counts lines, words, and characters in **demo.txt**.
* **wc -l demo.txt**: Shows the number of lines.
* **wc -w demo.txt**: Shows the word count.
* **wc -c demo.txt**: Shows the character count.

**Finding Files:**

**find**: Find files and directories.

* **find . -name "demo.txt"**: Search for **demo.txt** in the current directory.
* **find . -iname "demo.txt"**: Search case-insensitively for **demo.txt**.
* **find . -type f**: Lists all **files**.
* **find . -type d**: Lists all **directories**.
* **find . -size -100k**: Find files smaller than **100 KB**.
* **find . -size +100k**: Find files larger than **100 KB**.
* **find . -mtime -3**: Find files modified in the last 3 days.
* **find . -atime -3** finds all files accessed in the last **3 days**.

 **du**: Disk usage command.

* **du -h demo.txt**: Displays the disk usage of **demo.txt** in human-readable format.

**Understanding File Permissions in Linux**

In Linux, each file has three sets of permissions:

1. **Owner** (u): The user who owns the file.
2. **Group** (g): The group associated with the file.
3. **Others** (o): All other users who are not the owner or part of the group.

**-rw-rw-r--**

This string can be broken down as follows:

* The first character represents the file type (- for a regular file, d for a directory).
* The next three characters represent the permissions for the **owner** (user).
* The next three characters represent the permissions for the **group**.
* The last three characters represent the permissions for **others**.

Each set of permissions can include:

* **r** (read)
* **w** (write)
* **x** (execute)

**Breaking Down the File Permissions String:**

For example: -rw-rw-r--

* **Owner** has **read** and **write** permissions (rw-).
* **Group** has **read** and **write** permissions (rw-).
* **Others** have **read** permission only (r--).

**Using chmod to Change File Permissions**

The **chmod** (change mode) command is used to change the permissions of a file or directory. There are two main ways to use it:

**1. Symbolic Mode:**

* **u** = user (owner)
* **g** = group
* **o** = others
* **a** = all (user, group, others)

You can use +, -, or = to add, remove, or set exact permissions.

* **Add a permission**: chmod u+x demo.txt (Adds execute permission to the owner)
* **Remove a permission**: chmod g-w demo.txt (Removes write permission from the group)
* **Set exact permissions**: chmod u=rw,g=r,o=r demo.txt (Sets owner to read/write, group and others to read)

Examples:

* **chmod u+x demo**: Adds execute permission for the owner of the file **demo**.
* **chmod u-x,g+x,o+x demo**: Removes execute permission for the owner, adds execute permission for the group, and adds execute permission for others.

**2. Numeric (Octal) Mode:**

Permissions are also represented by a number system:

* **r** = 4
* **w** = 2
* **x** = 1

The permissions for each of the owner, group, and others are given a number, and you sum the numbers for each set of permissions.

Example:

* **rw-** = 4 (read) + 2 (write) = 6
* **r--** = 4 (read) = 4
* **---** = 0

**chmod**: Change file permissions.

* **chmod u+x demo**: Adds **execute** permission to the **owner**.
* **chmod u-r demo**: Removes **read** permission from the **owner**.
* **chmod a=x demo**: Gives **execute** permission to **all** (user, group, others).
* **chmod 644 demo**: Sets **read/write** for **owner**, **read** for **group** and **others**.
* **chmod 755 demo**: Sets **read/write/execute** for **owner**, and **read/execute** for **group** and **others**.

**Echo Command:**

The echo command is used to print text or the value of variables to the terminal.

**Basic Usage:**

$ echo "This is Session"

$ numi=30

$ echo $numi

**Read Command:**

The read command is used to get user input during execution.

**Reading Input:**

$ read numi

$ echo $numi

$ numi=30

$ numa=20

$ echo $((numi + numa))

$ echo $(expr $numi + $nume)

**Cat Command:**

The **cat** command is used to display the contents of a file.

$ cat filename

$ cat /home/user/Student

* This will display the content of the file **Student** located in the /home/user/ directory.

**echo**: Prints text or variable values.

* **echo "text"**: Prints text.
* **echo $variable**: Prints the value of a variable.

**read**: Reads input from the user.

* **read variable\_name**: Prompts the user to input a value and stores it in the variable.

**pwd**: Prints the current working directory.

* **pwd**: Displays the absolute path of the current directory.

**cat**: Displays the contents of a file.

* **cat filename**: Prints the contents of the file named filename.

**sort Command:**

The sort command is used to **sort** lines of text in a file or from input.

**Basic Usage:**

**$ sort student**

* This sorts the content of the student file in **ascending order**.

**Sorting in Reverse Order:**

**$ sort -r student**

* The **-r** option sorts the data in **reverse order**.

**Sorting in Version Order (Numerically):**

**$ sort -V student**

* The **-V** option is used to sort data **version-wise** (e.g., sorting numbers with decimals correctly).

**Sorting Numerically:**

$ sort -n student

* This will sort the data based on the numeric values in the file, rather than alphabetically.

**Sort alphabetically**:

$ sort student

**Sort by numbers (scores)**:

$ sort -n student

**Sort in reverse order**:

$ sort -r student

**sort**: Sort text files or input.

* + **sort filename**: Sort the file in ascending order.
  + **sort -r filename**: Sort the file in reverse order.
  + **sort -V filename**: Sort version numbers correctly.
  + **sort -n filename**: Sort numerically (useful for numbers in a file).

**With space as the delimiter**:

* Use the command sort +1 -2 student to sort based on the second column.

**With semicolon or comma as the delimiter**:

* Use sort -t';' +1 -2 student or sort -t',' +1 -2 student respectively, depending on how your data is separated.

**Basic grep Usage:**

**$ grep "run" student**

* This command searches for the **word "run"** in the file student.
* It will display all lines from student where the word **"run"** is present.

$ grep -c "run" student

* The **-c** option counts the number of lines in the file student that contain the word **"run"**.
* **Output**: The number will indicate how many lines have the word **"run"**.

$ grep -i "run" student

* The **-i** option makes the search **case-insensitive**.
* This means it will match **"run"**, **"Run"**, **"RUN"**, and so on, regardless of case.

$ grep -v "run" student

* The **-v** option **inverts** the match.
* It will show all lines in the student file **except** those that contain the word **"run"**.

$ grep -n "run" student

* The **-n** option will show the **line number** along with the matching lines.
* For example, if the match is on the 3rd line, it will output something like:

$ grep "^t" student

* The **^** is used to **match lines** that begin with the letter **"t"**.
* It will display all lines from student where the first character is **"t"**.

$ grep "a$" student

* The **$** is used to match lines that **end with** the letter **"a"**.
* It will return lines where the last character of the line is **"a"**.

**Character Ranges in grep:**

$ grep "[a-d]" student

* The **[a-d]** is a **range** that matches any character **from "a" to "d"** (inclusive).
* This will find any line that contains **letters from "a" to "d"**.
* This will match **"Arun"**, as it contains **"a"**, and **"Sandy"**, because it contains **"a"** and **"d"**.

**2. Case-Insensitive Search with Ranges:**

$ grep -i "[a-d]" student

* The **-i** option makes the search **case-insensitive**.
* This means it will match **letters from "a" to "d"**, regardless of case (e.g., **"A"**, **"b"**, **"D"**, etc.).

**3. Focus on Specific Characters in grep:**

$ grep -i "[d,n]" student

* The **[d,n]** means the search is looking for either the **character "d"** or **"n"** (note: no need to put a comma between characters).
* **-i** makes it case-insensitive.

**4. Using egrep (Extended grep):**

$ egrep "Yunland" student

* **egrep** is an extended version of grep that allows you to use **extended regular expressions**.
* It works the same as grep, but **supports more advanced features** like +, ?, |, and () for grouping.

For example, the following egrep search will match "Yunland" anywhere in the file.

**5. Advanced Matching with egrep (Multiple Words):**

$ egrep "runland|lla" student

* The **|** symbol represents **OR** (in regular expressions).
* This command will find lines that contain either **"runland"** **or** **"lla"**.

**Explanation:**

* The egrep allows you to combine different search patterns using **logical OR** (using |) for a more powerful search.
* For example, if you search for "runland" or "lla", it will match any line containing either one.

**6. Using fgrep (Fixed String Search):**

$ fgrep "run" student

* **fgrep** is a **fixed-string** search utility, which **does not interpret special characters** (like \*, .).
* It is more efficient than grep or egrep when you just want to search for **exact fixed strings**, without worrying about special characters.

The **fgrep** will search for the exact string **"run"**, and will match any line that contains that exact sequence.

**Summary of grep, egrep, and fgrep Differences:**

| **Command** | **Function** | **Notes** |
| --- | --- | --- |
| **grep** | Basic search, interprets regular expressions | Supports basic regular expressions |
| **egrep** | Extended grep, allows extended regular expressions like ` | , +, ?, and ()` |
| **fgrep** | Fixed string search, no interpretation of special characters (\*, ., etc.) | Useful for exact matches without regex processing |

1. Check the directory you are in.  
2. Display a long listing or your directory contents, including hidden files.  
3. Display the contents of the file myfile.  
  "Dolphins understand symbols. Could species less closely related to man also have a capacity for symbolic communication?

The dolphins understand even more complex sign language commands. They comprehend both the meaning of the words and how word order changes meanings. They also seem to retain a mental image of an object whether they are still looking at it or not."

4. Create a sub-directory new.  
5. Copy the file myfile to this directory.  
6. Change the current directory to new.  
7. Rename the file in new as newtemp.  
8. Delete the file myfile.  
9. Open the file newtemp with the current line at  The Dolphins…  
10. Create a link to the file newtemp and call the new file linkfile.  
11. Check the file size of the file linkfile.  
12. Return to the parent directouy.  
13. Display the contents of the file newtemp using the absolute pathname.  
14. Make new you current working directory.  
15. Change to a directory mine that branches off the parent directory of your current directory.  
16. Change back to the directory new.

**Opening & Exiting vi**

| **Command** | **Description** |
| --- | --- |
| vi filename | Open a file in vi |
| :q | Quit (if no changes were made) |
| :q! | Quit without saving |
| :w | Save (write) the file |
| :wq or ZZ | Save and exit |
| :x | Save and exit (only if changes were made) |

**Navigation Commands**

| **Command** | **Description** |
| --- | --- |
| h | Move **left** |
| l | Move **right** |
| j | Move **down** |
| k | Move **up** |
| 0 (zero) | Move to the **beginning** of the line |
| ^ | Move to the **first non-blank** character of the line |
| $ | Move to the **end** of the line |
| w | Move forward one **word** |
| b | Move backward one **word** |
| gg | Go to the **beginning** of the file |
| G | Go to the **end** of the file |
| :n | Go to **line n** (e.g., :10 moves to line 10) |
| Ctrl + d | Move **half-page down** |
| Ctrl + u | Move **half-page up** |

**Editing Commands**

| **Command** | **Description** |
| --- | --- |
| i | Insert mode before the cursor |
| I | Insert at the beginning of the line |
| a | Append after the cursor |
| A | Append at the end of the line |
| o | Open a **new line below** and enter insert mode |
| O | Open a **new line above** and enter insert mode |
| r | Replace a **single character** |
| R | Replace **multiple characters** (overwrite mode) |
| x | Delete the character under the cursor |
| X | Delete the character before the cursor |
| dd | Delete (cut) the entire line |
| D | Delete from cursor to the end of the line |
| yy (Y) | Copy (yank) the entire line |
| p | Paste after the cursor |
| P | Paste before the cursor |
| u | Undo the last action |
| Ctrl + r | Redo an undone action |

**Search Commands**

| **Command** | **Description** |
| --- | --- |
| /word | Search forward for "word" |
| ?word | Search backward for "word" |
| n | Repeat the **last search** (forward) |
| N | Repeat the **last search** (backward) |
| :%s/old/new/g | Replace **all** occurrences of "old" with "new" |
| :s/old/new/g | Replace "old" with "new" **in the current line** |

**Advanced Commands**

| **Command** | **Description** |
| --- | --- |
| :set nu | Show **line numbers** |
| :set nonu | Hide **line numbers** |
| :w filename | Save file as "filename" |
| :r filename | Read and insert contents of "filename" |
| :!command | Run a shell command (e.g., :!ls to list files) |
| v | Start **visual mode** for selecting text |
| V | Start **visual line mode** |
| Ctrl + v | Start **visual block mode** |

Insert Mode Commands  
i → Insert text before the cursor.  
I → Insert text at the beginning of the current line.  
a → Append text after the cursor.  
A → Append text at the end of the current line.  
o → Open a new line below the current line and enter Insert mode.  
O → Open a new line above the current line and enter Insert mode.  
2. Replace Mode Commands  
r → Replace a single character under the cursor.  
R → Replace multiple characters (overwrite mode) until you press ESC.

**Opening & Exiting vi**

| **Command** | **Description** |
| --- | --- |
| vi filename | Open a file in vi |
| vim filename | Open a file in vim (if installed) |
| :q | Quit (if no changes were made) |
| :q! | Quit **without saving** |
| :w | Save (write) the file |
| :wq or ZZ | Save and exit |
| :x | Save and exit **only if changes were made** |
| :w filename | Save file as "filename" |
| :e filename | Open another file in the same session |
| :wq! | Save and exit, overriding permissions |

**vim Modes**

| **Mode** | **How to Enter** | **Description** |
| --- | --- | --- |
| **Normal Mode** | Esc | Default mode for navigation & commands |
| **Insert Mode** | i, a, o | Allows text entry |
| **Visual Mode** | v, V, Ctrl + v | Used for selecting text |
| **Command Mode** | : | Used for file operations & searches |

**Navigation Commands**

| **Command** | **Description** |
| --- | --- |
| h | Move **left** |
| l | Move **right** |
| j | Move **down** |
| k | Move **up** |
| 0 (zero) | Move to the **beginning** of the line |
| ^ | Move to the **first non-blank** character of the line |
| $ | Move to the **end** of the line |
| w | Move forward **one word** |
| b | Move backward **one word** |
| gg | Go to the **beginning** of the file |
| G | Go to the **end** of the file |
| :n | Go to **line n** (e.g., :10 moves to line 10) |
| Ctrl + d | Move **half-page down** |
| Ctrl + u | Move **half-page up** |
| Ctrl + f | Move **one full page down** |
| Ctrl + b | Move **one full page up** |

**Editing Commands**

| **Command** | **Description** |
| --- | --- |
| i | Insert **before** the cursor |
| I | Insert at the **beginning** of the line |
| a | Append **after** the cursor |
| A | Append at the **end** of the line |
| o | Open a **new line below** and enter insert mode |
| O | Open a **new line above** and enter insert mode |
| r | Replace a **single character** |
| R | Replace **multiple characters** (overwrite mode) |
| x | Delete the character **under the cursor** |
| X | Delete the character **before the cursor** |
| dd | Delete (cut) the **entire line** |
| D | Delete from cursor to the **end** of the line |
| yy (Y) | Copy (yank) the **entire line** |
| p | Paste **after** the cursor |
| P | Paste **before** the cursor |
| u | **Undo** the last action |
| Ctrl + r | **Redo** an undone action |

**Copy, Cut, and Paste**

| **Command** | **Description** |
| --- | --- |
| yy or Y | Copy the current line |
| dd | Cut (delete) the current line |
| p | Paste after the cursor |
| P | Paste before the cursor |
| y$ | Copy from cursor to end of line |
| d$ | Cut from cursor to end of line |
| yw | Copy a word |
| dw | Cut a word |
| y} | Copy a paragraph |
| d} | Cut a paragraph |

**Search & Replace**

| **Command** | **Description** |
| --- | --- |
| /word | Search **forward** for "word" |
| ?word | Search **backward** for "word" |
| n | Repeat the **last search** (forward) |
| N | Repeat the **last search** (backward) |
| :%s/old/new/g | Replace **all occurrences** of "old" with "new" |
| :s/old/new/g | Replace "old" with "new" **in the current line** |
| :%s/old/new/gc | Replace with **confirmation** |

**Working with Multiple Files**

| **Command** | **Description** |
| --- | --- |
| :e filename | Open another file |
| :bn | Switch to the **next buffer** |
| :bp | Switch to the **previous buffer** |
| :bd | Close a buffer |
| :sp filename | Open file in **horizontal split** |
| :vsp filename | Open file in **vertical split** |
| Ctrl + w + w | Switch between splits |

**Visual Mode (Selecting Text)**

| **Command** | **Description** |
| --- | --- |
| v | Start **visual mode** (character selection) |
| V | Start **visual line mode** |
| Ctrl + v | Start **visual block mode** |
| y | Copy selected text |
| d | Cut selected text |
| p | Paste selected text |

**Indentation & Formatting**

| **Command** | **Description** |
| --- | --- |
| >> | Indent the current line |
| << | Unindent the current line |
| :%!fmt | Format the entire file |
| :set ai | Enable **auto-indentation** |
| :set noai | Disable **auto-indentation** |

**Advanced Commands**

| **Command** | **Description** |
| --- | --- |
| :set nu | Show **line numbers** |
| :set nonu | Hide **line numbers** |
| :set ic | Ignore case in search |
| :set noic | Case-sensitive search |
| :set hlsearch | Highlight search results |
| :set nohls | Remove search highlighting |
| :!command | Run a shell command (e.g., :!ls) |
| :r filename | Read and insert contents of "filename" |
| :w !sudo tee % | Save file with **sudo privileges** (if permission denied) |

Copy code

set number " Show line numbers

set autoindent " Enable auto-indentation

set tabstop=4 " Set tab width

set shiftwidth=4 " Set indentation width

set expandtab " Convert tabs to spaces

syntax on " Enable syntax highlighting

* Always press **Esc** before using command-mode shortcuts.
* Use **ZZ** as a quick way to **save and quit** (:wq).
* Use **Ctrl + w + w** to switch between multiple splits.
* If you mess up a command, use **u (undo)** immediately.

This guide covers **everything you need to master vi/vim**. Let me know if you need more details! 🚀

**1. Delete Characters**

* x → Delete **the character** under the cursor.
* X → Delete **the character** before the cursor.

**2. Delete Words**

* dw → Delete **from the cursor** to the end of the current word.
* dW → Delete **from the cursor** to the end of the current word (ignores punctuation).

**3. Delete Lines**

* dd → Delete the **entire current line**.
* dG → Delete from the **current line to the end of the file**.
* dgg → Delete from the **current line to the beginning of the file**.
* d$ → Delete **from the cursor to the end of the line**.
* d0 → Delete **from the cursor to the beginning of the line**.