The find command in Ubuntu is a versatile command-line utility used to locate files and directories based on various criteria.

Basic Syntax:

find [path] [options] [expression]

[path]:	Specifies the directory where the search should begin.
[options]:	Refine the search criteria.
[expression]:	Defines the conditions for the search (e.g., filename, size, modification time).

/: Searches the entire file system from root.

. (dot): Searches the current working directory.

~: Searches your home directory.

Common Options and Examples:

==========

Finding by Name: find . -name *.log”

Case-sensitive: find /path/to/directory -name "filename.ext"

Case-insensitive: find /path/to/directory -iname "filename.ext"

echo "file" > file1

echo "file2" > file2

- find . “ *.*” -type f // show all files
- find . -type f -name "file1"

To search directories : find . -type d -name "my_directory"

Finding by Type:

- **Greater than 10MB:** find . -size +10M
- **Less than 5KB:** find . -size -5k

Finding by Modification Time:

Modified within the last 7 days: find . -mtime -7

Delete all .log files in a directory:

```
find . -name "*.log" -exec rm -f {} \;
```

Note: {} acts as a placeholder for each found item, and \; terminates the -exec command.

Delete all directory that start with test:

```
find . -type d -name "*test*" -exec rm -r {} \;
```

Finding by Permissions:

- Files with 755 permissions: find . -perm 755
- Files owned by a specific user: find /home -user username
- Files belonging to a specific group: find /home -group groupname

What is grep?

grep (Global Regular Expression Print) is used to **search for patterns** in files using **regular expressions**.

Basic Syntax

```
grep [OPTIONS] PATTERN [FILE...]
```

Search a pattern in a file

grep "hello" file.txt

- Searches for the word "hello" in file.txt.

-i → Ignore case

grep -i "hello" file.txt

- Matches "Hello", "HELLO", "heLLo", etc.

-v → Invert match (show lines not matching)

grep -v "error" logfile.log

- Shows all lines **except** those with "error".

-r or -R → Recursive search

grep -r "password" /etc/

-n → Show line numbers

grep -n "main" program.c

- Displays the matching line **with line number**.

-c → Count of matching lines

grep -c "404" access.log

- Returns the **number of lines** containing "404".

-L → Show file names without matches

grep -L "import" *.py

- Files that **do not** contain the word "import".

-w → Match whole words only

grep -w "get" code.py

- Matches **only "get"**, not "getter" or "forget".

Extended regex using -E (or use egrep)

grep -E "ERR|WARN|FAIL" log.txt

- Match any of the three words.

Match line starting with "ERROR"

grep "^ERROR" log.txt

Redirection in Linux (Ubuntu)

Redirection is used to control where **input comes from** and where **output goes to.**

📌 1. Standard Streams

Stream	File Descriptor	Description
Standard Input	`0`	Input (keyboard)
Standard Output	`1`	Output (screen)
Standard Error	`2`	Error messages

1️⃣ > → Redirect stdout (overwrite)

echo "Hello Linux" > output.txt

✓ Creates (or overwrites) output.txt with the line.

2️⃣ >> → Redirect stdout (append)

echo "More text" >> output.txt

✓ Adds to the end of output.txt without deleting existing content.

3️⃣ 2> → Redirect stderr

ls non_existing_file 2> error.log

✓ Error message is written to error.log.

4️⃣ 2>> → Append stderr

ls /fake_dir 2>> error.log

✓ Appends error to existing error.log.

Example: Save both output and error to a file

(ls /etc && ls /tmp) > out.txt 2>&1

/dev/null → "Black hole" for output

- Used when you don't want any output or error on screen.

/dev/null is a **special file** that discards **anything written to it**. It's like a **trash bin** for unwanted output.

Useful when:

- You **don't care** about the output or error.
- You want to **silence commands** in cron jobs or scripts.
- You want a command to **run quietly**.

Discard standard output (stdout)

```
ls /etc > /dev/null
```

✓ Output of `ls` goes to `/dev/null`, so nothing is shown on screen.

- ⚠️ **2. Discard standard error (stderr)**

```
ls /fake_dir 2> /dev/null
```

✓ Suppresses the error message: `ls: cannot access '/fake_dir': No such file or directory`

Suppress warnings or noisy tools

```
grep "value" data.txt 2> /dev/null
✓ If data.txt doesn't exist, no error message is shown.
```

Use in scripts to clean output

```
ping -c 1 google.com > /dev/null && echo "Internet OK"
✓ Only prints "Internet OK" if ping succeeds — no ping output shown.
```

wc - Word Count Command in Linux

wc stands for Word Count, but it can also count lines, characters, bytes, and words.

Syntax

```
wc [OPTIONS] [FILE...]
```

📦 Common Options

Option	Description
`-l`	Count **lines**
`-w`	Count **words**
`-c`	Count **bytes**
`-m`	Count **characters**
`-L`	Length of **longest line**

Basic word count of a file

wc file.txt

Output:

10 20 120 file.txt

Meaning:

- **10 lines**
- **20 words**
- **120 bytes**

Count only lines (-l)

wc -l file.txt

✓ Only number of lines shown.

Count only words (-w)

wc -w file.txt

✓ Counts number of words.

Count only characters (-m)

wc -m file.txt

✓ Counts number of characters (including whitespace).

Count only bytes (-c)

wc -c file.txt

✓ Useful for file size in bytes.

Longest line length (-L)

wc -L file.txt

✓ Shows length of the longest line in the file.

Multiple files at once

wc file1.txt file2.txt

Use with pipe (|)

cat file.txt | wc -l

✓ Count number of lines from cat.

Use with find

find . -name "*.java" | wc -l

✓ Counts how many .java files are in current directory tree.

Count words from a string (no file)

echo "Linux is awesome" | wc -w

Linux Pipe Operator (|)

The pipe (|) operator is used to connect the output of one command to the input of another. It allows you to chain commands together in a powerful and efficient way.

Syntax

```
command1 | command2 | command3 ...
```

- **Takes stdout of command1 → passes as stdin to command2.**
- **You can chain multiple commands together.**
- **Often used with: grep, sort, uniq, wc, cut, awk, sed, head, tail, etc.**

Count number of files in a directory

ls | wc -l

✓ ls lists files, wc -l counts them.

Find processes related to "nginx"

ps aux | grep nginx

✓ Shows all processes with "nginx" in them.

Show top 5 largest files

du -ah . | sort -rh | head -n 5

- **du -ah → disk usage of files**
- **sort -rh → reverse sort by human-readable size**
- **head -n 5 → top 5 results**

Search lines with "ERROR" and count them

cat log.txt | grep "ERROR" | wc -l

✓ Filters lines with "ERROR" and counts them.

Extract usernames from /etc/passwd

cat /etc/passwd | cut -d: -f1

✓ Cuts first field (username) from :-separated lines.

Find duplicate lines in a file

sort file.txt | uniq -d

✓ sort arranges lines; uniq -d shows duplicates.

Show disk usage of subdirectories

du -sh * | sort -h

✓ Human-readable disk usage, sorted.

Monitor logs in real-time and filter

tail -f /var/log/syslog | grep "eth0"

✓ Live view of logs filtered to show only eth0.

Count number of users currently logged in

who | wc -l

Best Practice: Avoid cat when not needed

Instead of:

cat file.txt | grep "error"

Use:

grep "error" file.txt

✓ More efficient and avoids unnecessary use of cat.

Show only the first 5 files in a directory

ls | head -n 5

ls lists all files → head shows the first 5.

Show only the last 3 files in a directory

ls | tail -n 3

Useful for checking recently created files (if sorted by time).