**GREP– 15 Practical Use Cases with examples**

📌 grep stands for:

Global Regular Expression Print

✅ In Simple Terms:

grep is a command-line tool that searches text or files for lines that match a pattern.

It’s like:

CTRL + F on steroids for the command line 🔥

💡 What Can grep Do?

| Capability | Example |
| --- | --- |
| 🔎 Search for words/patterns | Find "error" in logs |
| ✅ Filter specific lines | Show only 200 OK responses |
| ❌ Exclude lines | Hide lines with "DEBUG" |
| 🔢 Show line numbers | Locate pattern in large files |
| 🎯 Match using regular expressions | Match email, IP, etc. |
| 📊 Count occurrences | How many times "Fail" appears |
| 📁 Search multiple files | Find keyword across .log files |

🧪 Basic Syntax:

grep [options] pattern [file]

**🧪 Sample File: log.txt**

[ERROR] Database connection failed

[INFO] Server started

[WARNING] Low disk space

[INFO] Backup completed

[ERROR] Unable to read config file

**✅ 1. Find all lines that contain the keyword "ERROR"**

grep "ERROR" log.txt

📤 **Output:**

[ERROR] Database connection failed

[ERROR] Unable to read config file

🔍 **Explanation:**

* grep: The tool used to search text using a pattern.
* "ERROR": The string we’re searching for.
* log.txt: The file we’re searching in.

🧠 **Use Case:** You’re analyzing a production log and need to extract error messages for debugging or reporting.

**✅ 2. Case-insensitive search for "error"**

grep -i "error" log.txt

📤 Output is the same as above.

🔍 **Explanation:**

* -i: Ignores case while matching. Useful when logs may use ERROR, error, or Error.

🧠 **Use Case:** Your team uses mixed cases in logs, and you want to catch all instances of error-related messages.

**✅ 3. Show line numbers where "INFO" appears**

grep -n "INFO" log.txt

📤 Output:

2:[INFO] Server started

4:[INFO] Backup completed

🔍 **Explanation:**

* -n: Displays the line number of each match.
* This helps correlate logs with line number in the file.

🧠 **Use Case:** Developers want to trace where in a large log file an event started.

**✅ 4. Count how many lines contain "INFO"**

grep -c "INFO" log.txt

📤 Output:

2

🔍 **Explanation:**

* -c: Counts how many lines match the pattern.
* Doesn’t print the lines—just gives a total.

🧠 **Use Case:** You want a daily count of how many info-level logs were generated.

**✅ 5. Exclude lines containing "INFO"**

grep -v "INFO" log.txt

📤 Output:

[ERROR] Database connection failed

[WARNING] Low disk space

[ERROR] Unable to read config file

🔍 **Explanation:**

* -v: Inverts the match. Shows lines that **do not** contain the pattern.

🧠 **Use Case:** You want to analyze only warnings and errors, ignoring verbose info logs.

**✅ 6. Use regex to find lines starting with "[ERROR]"**

grep "^\\[ERROR\\]" log.txt

📤 Output:

[ERROR] Database connection failed

[ERROR] Unable to read config file

🔍 **Explanation:**

* ^: Anchors the search to the **start of the line**.
* \[ERROR\]: Escaped brackets to match literally.

🧠 **Use Case:** Some lines might say INFO [ERROR] something, but you want only those where error is at the very beginning.

**✅ 7. Print only matched word, not entire line**

grep -o "INFO" log.txt

📤 Output:

pgsql

CopyEdit

INFO

INFO

🔍 **Explanation:**

* -o: Outputs only the portion of the line that matches.
* If multiple matches per line, each is printed on a new line.

🧠 **Use Case:** Extract just keywords for tagging, alert labels, etc.

**✅ 8. Search for multiple patterns**

grep -E "ERROR|WARNING" log.txt

📤 Output:

[ERROR] Database connection failed

[WARNING] Low disk space

[ERROR] Unable to read config file

🔍 **Explanation:**

* -E: Enables extended regular expressions.
* |: Acts like logical OR.

🧠 **Use Case:** You want to alert only on error and warning messages from logs.

**✅ 9. Recursive search in a directory**

grep -r "password" /etc/

📤 Output:

/etc/app.conf:password=abc123

/etc/db.yaml:admin\_password=xyz

🔍 **Explanation:**

* -r: Recursively searches through all files and subfolders.
* Useful in security audits or config scanning.

🧠 **Use Case:** Check all system config files for exposed passwords or sensitive info.

**✅ 10. Match full word only**

grep -w "error" log.txt

📤 Output:

[ERROR] Database connection failed

[ERROR] Unable to read config file

🔍 **Explanation:**

* -w: Ensures it matches **whole words** only.
* Avoids matching things like "errors" or "serror".

🧠 **Use Case:** Avoids false positives when matching specific terms in logs or scripts.

**✅ 11. Show matched lines with highlighted color**

grep --color=auto "ERROR" log.txt

🧠 Makes debugging easier by highlighting matched parts.

**✅ 12. Extract all IP addresses**

🧪 Sample File: access.log

192.168.1.10 - - [12/Feb] "GET /index.html"

10.0.0.5 - - [12/Feb] "POST /login"

grep -Eo '[0-9]+\.[0-9]+\.[0-9]+\.[0-9]+' access.log

📤 Output:

192.168.1.10

10.0.0.5

🧠 Extracts all IPs using regular expressions.

**✅ 13. Filter out comment lines**

🧪 File: config.env

# This is a comment

PORT=8080

# Another comment

DEBUG=true

grep -v "^#" config.env

📤 Output:

PORT=8080

DEBUG=true

🧠 Cleans config files for processing.

**✅ 14. Find empty lines**

grep "^$" data.txt

📤 Output:

(blank line shown)

🧠 Useful in file cleanup automation.

**✅ 15. Search for specific word in all .sh files**

grep -H "nginx" \*.sh

📤 Output:

deploy.sh: systemctl restart nginx

🧠 -H prints filename too. Useful in CI/CD debugging when many shell files are present.