## Text File Processing

### Finding lines: grep

The **grep** command searches a file for lines matching a pattern. In other unix-based systems, there are two other grep commands: **egrep** (extended grep) and **fgrep** (fixed string grep). Linux combined them into one command. The **egrep** command runs **grep** with the **-extended-regexp** (or **-E**) switch, and the **fgrep** command runs grep with the **-fixed-strings** (or **-F**) switch.

The Linux **locate** command consults a database and returns a list of all of the pathnames containing a specific group of characters, much like a fixed-string grep.

The Linux **find** command searched for files that meet specific conditions, such as files with certain names or files greater than a specific size, for example.

Using this exercise, you will learn how to:

* Use **grep** for searching logs.
* Use **wc -l** or grep -c for counting matches.
* Use **locate** to find a file in the filesystem quickly.
* Use **find** for flexible file searches (by name, type, or time).

The **grep** command searches a file for lines matching a pattern. In other unix-based systems, there are two other grep commands: **egrep** (extended grep) and **fgrep** (fixed string grep). Linux combined them into one command. The egrep commands runs grep with the -extended-regexp (OR -E) switch, and the fgrep commands runs grep with the -fixed-strings (or -F) switch.

**Exercise: Investigating a Security Log**

**Scenario:**  
You are a junior cybersecurity analyst. Your system has generated a suspicious log file called security.log. It contains mock IP addresses, computer names, usernames, and alerts. Your task is to investigate the log using common Linux tools.

**Step 1. Create the Mock Log File**

Create a file called security.log with this content:

cat > security.log << "EOF"

[2025-08-26 09:15:23] ALERT: Failed login from 192.168.1.20 on host WIN-SEC01 user=admin

[2025-08-26 09:20:05] INFO: User guest logged in from 10.0.0.5 on host WEB-SRV01

[2025-08-26 09:22:44] ALERT: Failed login from 172.16.0.12 on host DB-SRV03 user=root

[2025-08-26 09:30:01] INFO: User alice logged in from 192.168.1.45 on host WIN-CLT07

[2025-08-26 09:45:33] ALERT: Suspicious activity from 203.0.113.77 on host MAIL-SRV02

[2025-08-26 09:50:17] INFO: User bob logged in from 192.168.1.100 on host WIN-CLT09

[2025-08-26 10:00:55] ALERT: Failed login from 198.51.100.23 on host APP-SRV05 user=admin

EOF

A screenshot of a computer screen

AI-generated content may be incorrect.

**Use grep to filter suspicious activity**

* Find all lines containing the word ALERT.
* Count how many times Failed login appears.
* Find which user accounts were attacked (Extract only lines mentioning user=)
* Pretend you don’t know where the file is. Use locate (install it is missing from the system) to find it:
* Find files modified today in the current directory (Use find to locate all .log files updated within the last day.)
* Extract all the IP addresses from the file.
* Extract only the IP addresses from all ALERT lines (hint: use grep -E with regex).

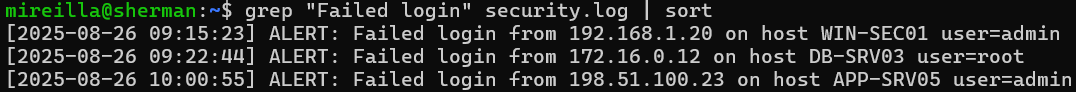
A black rectangular box with red blue and green text

AI-generated content may be incorrect.

### The sort command in Cybersecurity

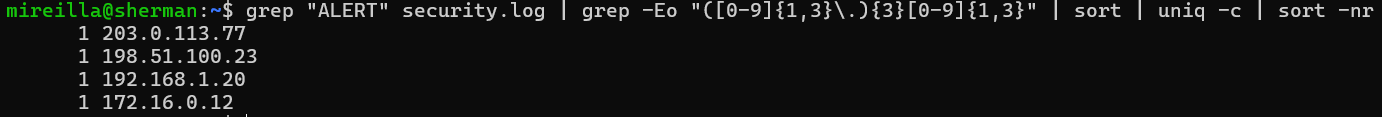
**Log Analysis**

Security logs (auth logs, firewall logs, intrusion detection logs) often have repeated entries. Using **sort** will group similar entries together, making patterns easier to spot. It turns messy raw logs into structured, analysable data.



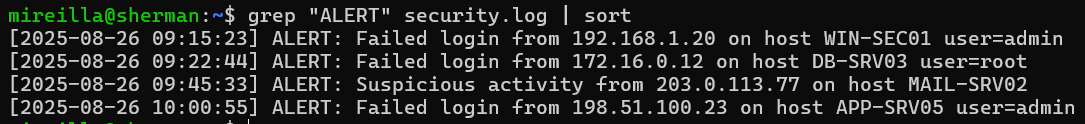
**Detecting Repeated IP Addresses (Brute-force / DoS attempts)**

Combine **sort** with **uniq -c** to count how many times each IP appears:

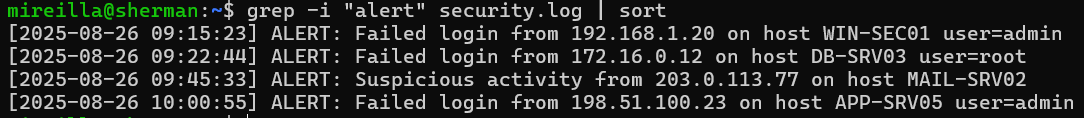


**Sorting Alerts by Time or Severity**

If log entries contain timestamps or severity levels (INFO, WARN, ALERT), sorting helps reorganise the data.







**Exercise:**

You are a junior cybersecurity analyst. Your team suspects that a brute-force attack is coming from one IP address that appears **most often** in the security log file. Your job is to identify which IP address is the **top attacker** (the one with the highest count).

**Instructions**

1. **Look for suspicious lines:** Use grep to extract only the lines in the log that contain ALERT.
2. **Extract the IP addresses:** From those ALERT lines, use grep -Eo with a regex to capture just the IPv4 addresses.
3. **Sort the IPs**: Sort the extracted IP addresses so that identical ones are grouped together.
4. **Count unique occurrences**: Use uniq -c to count how many times each IP appears.
5. **Find the top attacker (s)**

Use **sort -nr** to order the counts from highest to lowest.

Use **head -n 1** to display only the single most frequent IP.

Use **head -n 3** to find the top three frequent IP addresses that are the source of the brute-force attack.

**Task**: Create this security log file:

cat > security1.log << "EOF"

[2025-08-26 09:15:23] ALERT: Failed login from 192.168.1.20 on host WIN-SEC01 user=admin

[2025-08-26 09:20:05] INFO: User guest logged in from 10.0.0.5 on host WEB-SRV01

[2025-08-26 09:22:44] ALERT: Failed login from 172.16.0.12 on host DB-SRV03 user=root

[2025-08-26 10:05:00] ALERT: Failed login from 203.0.113.77 on host MAIL-SRV02 user=admin

[2025-08-26 09:30:01] INFO: User alice logged in from 192.168.1.45 on host WIN-CLT07

[2025-08-26 10:05:00] ALERT: Failed login from 203.0.113.77 on host MAIL-SRV02 user=admin

[2025-08-26 09:45:33] ALERT: Suspicious activity from 203.0.113.77 on host MAIL-SRV02

[2025-08-26 10:05:00] ALERT: Failed login from 203.0.113.77 on host MAIL-SRV02 user=admin

[2025-08-26 09:50:17] INFO: User bob logged in from 192.168.1.100 on host WIN-CLT09

[2025-08-26 10:05:00] ALERT: Failed login from 203.0.113.77 on host MAIL-SRV02 user=admin

[2025-08-26 10:00:55] ALERT: Failed login from 198.51.100.23 on host APP-SRV05 user=admin

EOF

Now, write a command to show the top 3 attackers.

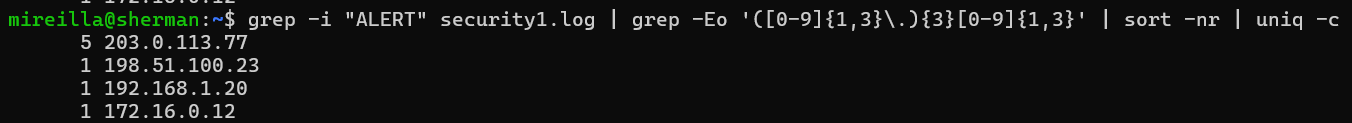
***Reverse sort***

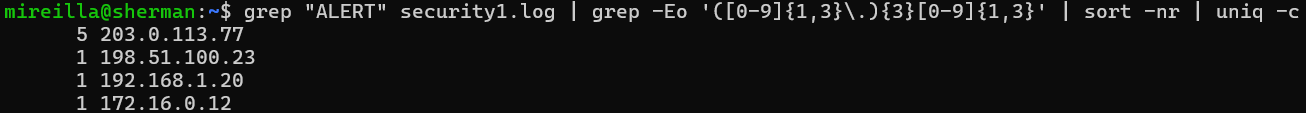
Reverse sorting **(sort -r)** is especially useful when analysing logs with counts. It is used to quickly spot top attackers, largest files, or most frequent events:

Run this command to extract all IPs from ALERT lines, group them, and count how often each appears:

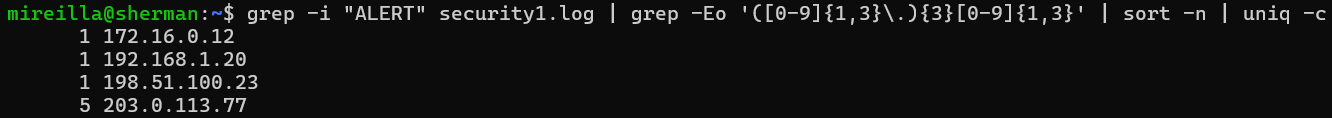


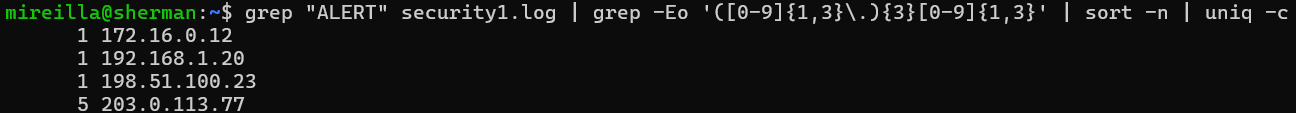
Sort in Reverse Order (largest → smallest)





Sort in Ascending Order (smallest → largest)





### Sorting Data in C and Linux

Why might we prefer Linux sort for large log files instead of writing our own sorter in C?

**Performance & optimisation**:  
Linux sort is highly optimised in C already, using efficient algorithms and memory handling. Writing your own sorter from scratch will almost always be slower and less memory-efficient.

**Scalability**:  
sort can handle **gigabytes of logs** directly from disk, streaming line by line. A custom C program would need careful design to avoid running out of memory.

**Convenience**:  
One line of shell (sort file.log) replaces dozens of lines of C code. Easy to combine with other tools (grep, uniq, awk) in pipelines.

In log analysis, why is sorting before counting **(uniq -c)** necessary?

The uniq command only removes adjacent duplicates. If the log is not sorted, the same IP or username might appear scattered in different places, and **uniq** will not group them. So, always **sort before uniq** to get correct counts.

Use Linux sort for **big logs** (faster, proven).

Imagine these are **IP addresses from a log file**. We want to find **the most frequent attackers**.

A screen shot of a computer

AI-generated content may be incorrect.

Explanation:

* grep -Eo → extract IP addresses.
* sort → alphabetically sort them.
* uniq -c → count occurrences.
* sort -nr → sort by count (highest first).
* head → show top results.

This is the real-world application of sorting in cybersecurity

### Using word count (wc)

**wc** stands for word count, prints counts of lines, words, and characters/bytes in files.

**Common Options**

* wc -l file: **line count** (how many lines in the file)
* wc -w file: **word count** (how many words in the file)
* wc -c file: **byte count** (file size in bytes)
* wc -m file: **character count** (useful with UTF-8 text)
* wc file: prints all 3 (lines, words, bytes)

A screen shot of a computer program

AI-generated content may be incorrect.

**Task**: Count lines from grep output - Counts how many ALERT entries are in the log



### Using **tr** (translate)

The Linux **tr** (translate) command will substitute or delete characters on standard input, writing the results to standard output. The **-d** will delete a special character.

A black and white screen with white text

AI-generated content may be incorrect.

Without options, **tr** will map one set of characters to another. In this example, lowercase ‘aeiou’ are replaced by uppercase ‘AEIOU’.



We can also use the **-complement** (or **-c**) switch to reverse the sense matching. In this example, the characters in the first parameter are not mapped to the second; instead, the characters NOT in the first parameter are replaced with the indicated character.



Notice that the output does not end with a newline (**\n**). This is because the output does not end with a newline character. This is because my pipeline replaced all non-vowels with ‘**?**’. Here is the output on a separate line.



### File editing with **sed**

The Linux **sed (stream editor)** command processes text line by line. Unlike normal text editors such as vim or nano, you cannot interactively edit files. Instead, you give **sed** instructions, for example, *find a pattern and replace it with something else*. Compared to grep, which finds patterns, **sed** can find patterns and edit them automatically (substitute, delete, insert, etc).

Here is a test file

"EOF"

Sherlock Holmes lived at 221B Baker Street.

Dr. Watson often visited Holmes at his home.

One day, Holmes said: "Elementary, my dear Watson."

EOF

A screen shot of a computer

AI-generated content may be incorrect.

Use **grep** to *find* text (Holmes, in this case). In this output, we can see the lines with the match.

A black background with white text

AI-generated content may be incorrect.

Use **sed** to *replace* text. In this example, ‘Holmes’ was replaced with ‘Detective’.

A black screen with white text

AI-generated content may be incorrect.

We can send the results to a new file called newtest.txt

A screen shot of a computer

AI-generated content may be incorrect.

### Curl

**curl** stands for **Client URL**. It is used to transfer data to or from a server using a variety of protocols (HTTP, HTTPS, FTP, etc.). A command-line tool to interact with servers over the internet — download, upload, send requests, inspect responses. In Cybersecurity, it can be used for:

* 1. Reconnaissance: For example, to inspect HTTP headers, cookies, and server information.
  2. Testing: For example, to send requests to APIs or login form (often used for penetration testing)
  3. Automation: For example, to automate downloads, scans, or brute-force attempts
  4. Incident response: For example, to quickly check connectivity or grab Indicators of Compromise from threat feed.

**Task:**

<https://www.bbc.co.uk/news/articles/czxwll89e79o>

Download this webpage to example.html using curl.

Count the number of words in example.html. Make sure it is silent (does not print out the progression information as it downloads the data.) (Use **-s** (silent) and **-o** (output to file)