**Linux Command Learnings Summary with Examples and Flags**

**1. GREP Command: Pattern Matching and Searching**

**grep** is used to search for specific patterns within files. The -E flag enables extended regular expressions, while -o ensures only the matching part is displayed.

**Key Flags:**

| **Flag** | **Description** | **Example Usage** |
| --- | --- | --- |
| -E | Enable extended regular expressions | `grep -E "error |
| -o | Show only the matching part | grep -Eo '[0-9]{2,}' sample-logs.md |
| -i | Case-insensitive search | grep -i "error" |
| -v | Invert match (exclude pattern) | grep -v "DEBUG" |
| -c | Count matching lines | grep -c "ERROR" sample-logs.md |
| -n | Show line numbers of matches | grep -n "ERROR" sample-logs.md |
| -l | List file names with matches | grep -l "ERROR" \*.log |
| -A N | Show N lines after the match | grep -A 2 "ERROR" sample-logs.md |
| -B N | Show N lines before the match | grep -B 2 "ERROR" sample-logs.md |
| -C N | Show N lines before and after the match | grep -C 2 "ERROR" sample-logs.md |

**Key Examples:**

1. **Find All Email Addresses:**

grep -Eo '[a-zA-Z0-9.\_%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}' sample-logs.md

Output: Extracts valid email addresses from the log.

1. **Find API Requests Taking More Than 1000ms:**

grep -E "completed in [1-9][0-9]{3,}ms" sample-logs.md

Output: Shows API requests that took **1000 milliseconds or more**.

1. **Find High CPU Usage (70% or More):**

grep -E "CPU usage: [7-9][0-9]%" sample-logs.md

Output: Extracts CPU usage that exceeds **70%**.

1. **Show Lines with "ERROR" and 2 Lines After:**

grep -A 2 "ERROR" sample-logs.md

**2. AWK Command: Text Processing and Field Extraction**

**awk** is a text-processing tool that allows you to extract and manipulate specific fields from each line.

**Key Flags:**

| **Flag/Variable** | **Description** | **Example Usage** |
| --- | --- | --- |
| -F | Set field separator | awk -F ',' '{print $1}' for CSV files |
| NR | Line number of the current record | awk '{print NR, $0}' |
| NF | Number of fields in the current record | awk '{print NF, $0}' |
| BEGIN | Actions before processing starts | awk 'BEGIN {print "Start"}' |
| END | Actions after processing ends | awk 'END {print "Done"}' |

**Key Examples:**

1. **Extract the 3rd Field (API Completion Time):**

awk '{print $3}' sample-logs.md

1. **Count Unique Occurrences of a Field:**

awk '{print $3}' sample-logs.md | sort | uniq -c

1. **Filter API Request Completion Times (in ms):**

awk '/API request completed in/ {print $9}' sample-logs.md | sed 's/ms//' | sort -n

1. **Print Line Numbers and Matching Lines:**

awk '/ERROR/ {print NR, $0}' sample-logs.md

**3. TR Command: Character Translation and Deletion**

**tr** is used to translate, delete, or squeeze characters in text.

**Key Flags:**

| **Flag** | **Description** | **Example Usage** |
| --- | --- | --- |
| -d | Delete characters | `echo "123abc" |
| -s | Squeeze repeated characters | `echo "aaabbb" |
| -c | Complement (inverse) character set | `echo "hello" |

**Key Examples:**

1. **Squeeze Repeated Characters:**

echo "aaabbbbccddd" | tr -s 'b'

Output: aaabccddd (Squeezes bbbb to b).

1. **Delete Digits from a String:**

echo "abc123def" | tr -d '0-9'

Output: abcdef

1. **Replace Spaces with Underscores:**

echo "Hello World" | tr ' ' '\_'

Output: Hello\_World

**4. SORT Command: Organizing Output Alphabetically or Numerically**

**sort** arranges lines in ascending or descending order.

**Key Flags:**

| **Flag** | **Description** | **Example Usage** |
| --- | --- | --- |
| -n | Sort numerically | sort -n numbers.txt |
| -r | Reverse order | sort -r names.txt |
| -u | Unique values | sort -u sample.txt |
| -k | Sort by specific column | sort -k2 -n data.txt |

**Key Examples:**

1. **Sort API Request Times (Numerically):**

grep -Eo "API request completed in ([1-9][0-9]{3,})ms" sample-logs.md | sort -n

1. **Sort Lines in Reverse Order:**

sort -r sample-logs.md

1. **Sort by 3rd Column (CPU Usage):**

sort -k3 -n sample-logs.md

**5. CRONTAB Command: Scheduling Tasks**

**crontab** schedules tasks to run automatically at specified times.

**🕰️ Crontab Time Fields:**

| **Field** | **Meaning** | **Allowed Values** |
| --- | --- | --- |
| Minute | Minute of the hour | 0-59 |
| Hour | Hour of the day | 0-23 |
| Day of Month | Day of the month | 1-31 |
| Month | Month of the year | 1-12 |
| Day of Week | Day of the week | 0-7 (0 and 7 = Sunday) |

**Key Examples:**

1. **Run a Backup Daily at 2:00 AM:**

0 2 \* \* \* /home/user/backup.sh

1. **Run Every 10 Minutes:**

\*/10 \* \* \* \* /home/user/monitor.sh

1. **List Existing Cron Jobs:**

crontab -l

1. **Edit Cron Jobs:**

crontab -e

1. **Remove All Cron Jobs:**

crontab -r

**6. Takeaways:**

1. **GREP:** Efficient pattern matching with advanced regular expressions.
2. **AWK:** Field extraction and text processing.
3. **TR:** Character translation, deletion, and squeezing.
4. **SORT:** Sorting lines by alphabet, number, or specific columns.
5. **CRONTAB:** Automating recurring tasks with flexible scheduling.