**Essential Linux Commands for Developers**

**I. File and Directory Navigation**

1. **ls**:
   * Lists files and directories in the current directory.
   * ls -l: Lists files with detailed information (permissions, size, etc.).
   * ls -a: Lists all files, including hidden files (those starting with a dot).
   * ls -lh: Lists files in human readable format, and long format.
2. **cd <directory>**:
   * Changes the current directory.
   * cd ..: Moves up one level to the parent directory.
   * cd ~: Moves to the user's home directory.
3. **pwd**:
   * Prints the current working directory.

**II. File and Directory Management**

1. **mkdir <directory>**:
   * Creates a new directory.
   * mkdir -p <directory/subdirectory>: Creates parent directories as needed.
2. **rm <file>**:
   * Removes a file.
   * rm -r <directory>: Removes a directory and its contents recursively (use with caution).
   * rm -rf <directory>: Forcefully removes a directory and its contents (extremely dangerous).
3. **touch <file>**:
   * Creates an empty file or updates the timestamp of an existing file.
4. **cp <source> <destination>**:
   * Copies a file or directory.
   * cp -r <source\_directory> <destination\_directory>: Copies a directory recursively.
5. **mv <source> <destination>**:
   * Moves (renames) a file or directory.

**III. File Content and Searching**

1. **cat <file>**:
   * Displays the contents of a file.
2. **less <file>**:
   * Displays the contents of a file one page at a time (use q to exit).
3. **nano <file> or vim <file>**:
   * Opens a text editor to create or edit files.
   * nano is user-friendly for beginners; vim is powerful but requires a learning curve.
4. **grep <pattern> <file>**:
   * Searches for a pattern within a file.
   * grep -r <pattern> <directory>: Recursively searches for a pattern within a directory.
5. **find <directory> -name <filename>**:
   * Finds files based on their name.
   * find . -name "\*.txt": Finds all .txt files in the current directory and its subdirectories.

**IV. Permissions and System Information**

1. **chmod <permissions> <file>**:
   * Changes file permissions.
   * chmod +x <file>: Makes a file executable.
2. **ssh <user>@<host>**:
   * Establishes a secure shell connection to a remote server.
3. **scp <file> <user>@<host>:<destination>**:
   * Securely copies files between local and remote systems.
4. **sudo <command>**:
   * Executes a command with superuser (administrator) privileges.
5. **history**:
   * Shows a list of previously executed commands.
6. **clear**:
   * Clears the terminal screen.
7. **df -h**:
   * Displays disk space usage in human-readable format.
8. **du -sh <directory>**:
   * Displays the disk usage of a directory in summary format.

**V. Process Management and Networking**

1. **ps aux | grep <process\_name>**:
   * Lists all running processes and filters the list by the process name.
2. **kill <process\_id>**:
   * Terminates a process by its process ID.
3. **wget <url>**:
   * Downloads a file from a URL.
4. **curl <url>**:
   * Transfers data from or to a server.

**VI. Essential Utilities**

1. **man <command>**:
   * Displays the manual page for a command.
2. **echo <text>**:
   * Displays text in the terminal.
3. **export <VARIABLE>=<value>**:
   * Sets or modifies environment variables.
4. **source ~/.bashrc or source ~/.zshrc**:
   * Reloads the bash or zsh configuration file.
5. **alias <alias\_name>='command'**:
   * Creates an alias for a frequently used command.
6. **tar -czvf <archive\_name>.tar.gz <directory>**:
   * Creates a compressed archive of a directory.
7. **tar -xzvf <archive\_name>.tar.gz**:
   * Extracts files from a compressed archive.

**Linux Monitoring Tools and Techniques**

**I. System Resource Monitoring**

1. **top or htop**:
   * Displays a real-time, dynamic view of running processes and system resource usage (CPU, memory, swap).
   * htop is an interactive, enhanced version of top.
   * Press q to exit.
2. **vmstat**:
   * Reports virtual memory statistics, including processes, memory, paging, I/O, and CPU activity.
   * vmstat 1: Updates the statistics every second.
3. **iostat**:
   * Reports CPU utilization and disk I/O statistics.
   * iostat -xz 1: Provides extended statistics for each device, updating every second.
4. **df**:
   * Displays disk space usage.
   * df -h: Displays disk space usage in a human-readable format.
5. **du**:
   * Estimates file and directory space usage.
   * du -sh <directory>: Displays the total size of a directory in a summary format.
6. **free**:
   * Displays the amount of free and used physical and swap memory in the system.
   * free -m: Displays memory usage in megabytes.
7. **uptime**:
   * Shows how long the system has been running, the number of logged-in users, and the system load averages.
8. **mpstat**:
   * Reports processor-related statistics.
   * mpstat -P ALL 1: Displays statistics for each processor, updating every second.
9. **netstat or ss**:
   * Displays network connections, routing tables, interface statistics, masquerade connections, and multicast memberships.
   * netstat -tulnp: Lists listening TCP and UDP ports with process IDs.
   * ss -tulnp is a more modern and faster replacement for netstat.
10. **sar (System Activity Reporter)**:
    * Collects, reports, and saves system activity information.
    * Requires installation (usually sysstat package).
    * sar -u 1: Reports CPU utilization every second.

**II. Process Monitoring**

1. **ps**:
   * Displays information about running processes.
   * ps aux: Displays all processes with detailed information.
   * ps -ef | grep <process\_name>: Filters processes by name.
2. **pgrep**:
   * Looks up processes based on their names or other attributes.
   * pgrep <process\_name>: Returns the process IDs of matching processes.
3. **lsof (List Open Files)**:
   * Lists open files and the processes that are using them.
   * lsof -i :<port\_number>: Lists processes using a specific port.
4. **strace**:
   * Traces system calls and signals.
   * Useful for debugging and understanding how a process interacts with the kernel.
   * strace <command>: Traces the system calls of the given command.

**III. Log Monitoring**

1. **/var/log/ Directory:**
   * The standard location for system and application log files.
   * Common logs include:
     + /var/log/syslog or /var/log/messages: System-wide messages.
     + /var/log/auth.log: Authentication-related messages.
     + /var/log/kern.log: Kernel messages.
2. **tail**:
   * Displays the end of a file.
   * tail -f <log\_file>: Follows the log file in real-time.
3. **less or more**:
   * Displays the contents of a file one page at a time.
   * less is more powerful, allowing backward scrolling.
4. **grep**:
   * Searches for patterns in files.
   * grep "error" <log\_file>: Searches for lines containing "error".
5. **journalctl**:
   * Queries the systemd journal for log messages.
   * journalctl -u <service\_name>: Displays logs for a specific service.
   * journalctl -f: Follows the journal in real-time.
6. **awk and sed**:
   * Powerful text processing tools for analyzing and manipulating log files.
7. **logrotate**:
   * Utility for managing log files.
   * Automates log rotation, compression, and deletion.

**IV. Monitoring Tools**

1. **Nagios, Zabbix, Prometheus**:
   * Advanced monitoring systems for network and system monitoring.
   * Provide dashboards, alerts, and historical data.
2. **Grafana**:
   * A data visualization and monitoring tool that works well with Prometheus and other data sources.
3. **Checkmk**:
   * Comprehensive IT monitoring solution.

**Example Monitoring Scenarios:**

* **CPU Usage:** Use top or mpstat to monitor CPU utilization.
* **Disk Space:** Use df -h to check available disk space.
* **Memory Usage:** Use free -m to monitor memory usage.
* **Log Analysis:** Use tail -f and grep to monitor log files for errors.
* **Network Monitoring:** use ss -tulnp to check listening ports.
* **Process Monitoring:** Use ps aux to check running processes.

**Linux Job Management**

**I. Foreground and Background Jobs**

1. **Foreground Jobs:**
   * A foreground job is a process that runs in the terminal and occupies the shell until it completes.
   * You interact directly with foreground jobs.
   * Example: nano my\_file.txt (nano editor runs in the foreground).
2. **Background Jobs:**
   * A background job is a process that runs independently of the terminal.
   * You can continue using the terminal while background jobs are running.
   * To run a command in the background, append an ampersand (&) to the end of the command.
   * Example: long\_running\_script.sh &

**II. Job Control**

1. **jobs**:
   * Lists the currently running background jobs.
   * Displays job IDs and status.
   * Example output:
   * [1]+ Running long\_running\_script.sh &
2. **fg %<job\_id>**:
   * Brings a background job to the foreground.
   * %<job\_id> refers to the job ID displayed by the jobs command.
   * Example: fg %1
3. **bg %<job\_id>**:
   * Resumes a suspended background job.
   * If a job is stopped (e.g., by pressing Ctrl+Z), bg will restart it in the background.
   * Example: bg %1
4. **Ctrl+Z**:
   * Suspends the currently running foreground job.
   * The job is paused and can be resumed using fg or bg.
5. **Ctrl+C**:
   * Terminates the currently running foreground job.
6. **kill %<job\_id>**:
   * Terminates a background job.
   * Example: kill %1
7. **kill <process\_id>**:
   * Terminates a process by its process ID (PID).
   * Use ps or top to find the PID.
   * kill -9 <process\_id>: Forcefully terminates a process (use with caution).
8. **ps**:
   * Lists running processes.
   * ps aux: Lists all processes with detailed information.
   * ps -ef | grep <process\_name>: Filters processes by name.
9. **top or htop**:
   * Displays a real-time view of system processes and resource usage.
   * htop is an interactive process viewer with more features than top.
10. **nohup <command> &**:
    * Runs a command in the background and prevents it from being terminated when you log out.
    * The output is redirected to nohup.out by default.
    * Example: nohup long\_running\_script.sh &
11. **screen or tmux**:
    * Terminal multiplexers that allow you to create persistent terminal sessions.
    * You can detach from a session and reattach later, even if you log out.
    * Useful for long-running processes that need to continue running.
    * screen -S <session\_name>: Creates a new screen session.
    * Ctrl+a d: Detaches from a screen session.
    * screen -r <session\_name>: Reattaches to a screen session.
12. **systemd (systemctl)**
    * systemd is a system and service manager for Linux. It is used to control and manage services.
    * systemctl status <service\_name>: shows the status of a service.
    * systemctl start <service\_name>: starts a service.
    * systemctl stop <service\_name>: stops a service.
    * systemctl restart <service\_name>: restarts a service.
    * systemctl enable <service\_name>: enables a service to start at boot.
    * systemctl disable <service\_name>: disables a service from starting at boot.

**Essential Linux Commands for Developers: Scheduling and Automation**

**I. Scheduling and Automation**

Linux provides powerful tools for automating tasks and scheduling jobs, which are crucial for developers and system administrators.

1. **cron**:
   * The cron daemon is a time-based job scheduler in Unix-like operating systems.
   * It allows you to schedule commands or scripts to run automatically at specific intervals.
   * **crontab -e**: Edits the user's crontab file (where scheduled jobs are defined).
   * **Crontab Syntax:**
     + \* \* \* \* \* command
     + Minute (0-59) Hour (0-23) Day of month (1-31) Month (1-12) Day of week (0-7, 0 and 7 are Sunday)
     + Example: 0 0 \* \* \* /path/to/script.sh (Runs the script at midnight every day).
     + Special strings:
       - @reboot run once, at startup.
       - @yearly run once a year, "0 0 1 1 \*".
       - @monthly run once a month, "0 0 1 \* \*".
       - @weekly run once a week, "0 0 \* \* 0".
       - @daily run once a day, "0 0 \* \* \*".
       - @hourly run once an hour, 1 "0 \* \* \* \*".
2. **at**:
   * The at command schedules a command to be executed once at a specified time.
   * at 10:00: Schedules a command to run at 10:00 AM.
   * at now + 1 hour: Schedules a command to run one hour from now.
   * After typing the at command, you will be prompted to enter the command to be executed. Press Ctrl+D to finish.
3. **nohup**:
   * The nohup command allows a command to continue running even after you log out of the terminal.
   * nohup /path/to/script.sh &: Runs the script in the background and prevents it from being terminated when you log out.
   * The output of the command is typically redirected to a file named nohup.out.
4. **systemd timers**:
   * A more modern alternative to cron, especially in systems using systemd.
   * Timers are defined as systemd unit files, providing more flexibility and control.
   * This is more complex than cron, but more powerfull.
5. **Bash scripting:**
   * Creating bash scripts to automate repetitive tasks is a very common practice.
   * Example:

Bash

#!/bin/bash

# This is a sample script

date >> log.txt

ls -l >> log.txt

* + Make the script executable: chmod +x script.sh
  + Run the script: ./script.sh

1. **watch**:
   * Executes a command repeatedly and displays the output, allowing you to monitor changes over time.
   * watch -n 1 ls -l: Executes ls -l every second.
2. **awk and sed**:
   * These are powerful text processing tools that can be used in scripts to automate data manipulation.
   * sed is for stream editing, and awk is for pattern scanning and processing.
3. **Redirecting output:**
   * >: Redirects output to a file, overwriting existing content.
   * >>: Appends output to a file.
   * 2>: Redirects error output to a file.
   * &>: Redirects both standard output and error output to a file.