**You said:**

ls - List Directory Contents • Basic: ls • Flags: o -l: Long format, shows detailed information (permissions, size, owner, etc.) o -a: Show hidden files (files starting with a dot) o -h: Human-readable sizes (e.g., 1K, 234M) o -R: Recursively list directories o -S: Sort by file size o -t: Sort by modification time o --color: Colorize output Example: ls -l -h -a 2. cd - Change Directory • Basic: cd /path/to/directory • Flags: o cd ~: Change to home directory o cd -: Go back to the previous directory o cd ..: Go up one level 3. pwd - Print Working Directory • Basic: pwd o Prints the full path of the current directory. 4. mkdir - Make Directory • Basic: mkdir directory\_name • Flags: o -p: Create parent directories if they do not exist o -v: Show verbose output of directories created Example: mkdir -p /path/to/new/directory 5. rmdir - Remove Empty Directory • Basic: rmdir directory\_name • Flags: o -p: Remove parent directories as well if empty o -v: Verbose output 6. rm - Remove Files or Directories • Basic: rm filename • Flags: o -r: Recursively remove directories and their contents o -f: Force removal without confirmation o -i: Prompt before every removal o -v: Verbose output Example: rm -rf directory\_name (Caution: This will delete a directory and all its contents) 7. cp - Copy Files or Directories • Basic: cp source destination • Flags: o -r: Copy directories recursively o -i: Prompt before overwriting files o -v: Verbose output o -u: Copy only when the source file is newer or the destination is missing Example: cp -r folder1 folder2 8. mv - Move or Rename Files/Directories • Basic: mv source destination • Flags: o -i: Prompt before overwriting files o -u: Move only when the source file is newer o -v: Verbose output Example: mv oldname.txt newname.txt 9. touch - Create Empty Files or Change Timestamps • Basic: touch filename • Flags: o -c: Do not create a new file if it doesn't exist o -t: Set a specific timestamp (e.g., touch -t 202304050730 filename) 10. find - Search for Files • Basic: find /path/to/search -name filename • Flags: o -name: Search by file name o -type: Search by type (e.g., f for files, d for directories) o -exec: Execute a command on the files found o -mtime: Search by modification time (e.g., +7 for files modified more than 7 days ago) o -size: Search by size (e.g., +100M for files larger than 100MB) Example: find /path/to/search -name "\*.txt" -exec ls -l {} \; 11. cat - Concatenate and Display Files • Basic: cat filename • Flags: o -n: Number lines o -b: Number non-empty lines o -E: Show $ at the end of each line 12. more and less - View File Contents • Basic: more filename or less filename • Flags for more: o -c: Clear screen before displaying • Flags for less: o -N: Show line numbers o -S: Disable line wrapping 13. head - Show the Beginning of a File • Basic: head filename • Flags: o -n: Display a specific number of lines (e.g., head -n 20 filename) 14. tail - Show the End of a File • Basic: tail filename • Flags: o -n: Show the last N lines (e.g., tail -n 50 filename) o -f: Continuously monitor the file as it grows (useful for logs) 15. chmod - Change File Permissions • Basic: chmod permissions filename • Flags: o -R: Apply recursively to directories and files o Numeric mode: chmod 755 filename (e.g., 755 for rwxr-xr-x) o Symbolic mode: chmod u+x filename (e.g., add execute for user) 16. chown - Change File Ownership • Basic: chown user:group filename • Flags: o -R: Apply recursively to directories and files 17. ln - Create Hard and Symbolic Links • Basic: ln source link\_name (hard link) • Flags: o -s: Create a symbolic (soft) link o -f: Force the creation of the link 18. tar - Archive Files • Basic: tar -cf archive.tar files • Flags: o -x: Extract files from an archive o -v: Verbose output o -z: Compress using gzip (e.g., tar -czf archive.tar.gz files) o -j: Compress using bzip2 (e.g., tar -cjf archive.tar.bz2 files) o -J: Compress using xz (e.g., tar -cJf archive.tar.xz files) 19. zip and unzip - Create and Extract ZIP Files • Basic: o zip archive.zip files: Create a ZIP archive o unzip archive.zip: Extract a ZIP archive • Flags: o -r: Recursively zip directories o -d: Extract to a specific directory (e.g., unzip archive.zip -d /path/to/dir) 20. df - Display Disk Space Usage • Basic: df • Flags: o -h: Human-readable format (e.g., 1K, 234M) o -T: Show file system type 21. du - Estimate File Space Usage • Basic: du filename • Flags: o -h: Human-readable format o -s: Show only total size for a directory o -a: Show size for files and directories 22. stat - Display File Status • Basic: stat filename • Flags: o -c: Specify output format (e.g., stat -c %s filename for file size) 23. sync - Synchronize Data to Disk • Basic: sync o This command flushes the file system buffer, writing data to disk. 24. mount and umount - Mount and Unmount File Systems • Basic: o mount /dev/sdX /mnt: Mount a device to a directory o umount /mnt: Unmount a device 25. file - Determine File Type • Basic: file filename • Flags: o -i: Display MIME type 26. shred - Securely Delete Files • Basic: shred -u filename • Flags: o -n: Specify the number of overwrite passes (default is 3) o -z: Add a final pass with zeros to hide shredding 27. bzip2 and gzip - Compress and Decompress Files • Basic: o gzip filename: Compress o gzip -d filename.gz: Decompress o bzip2 filename: Compress o bzip2 -d filename.bz2: Decompress 28. rsync - Synchronize Files and Directories • Basic: rsync -av source/ destination/ • Flags: o -a: Archive mode (preserves permissions, symlinks, etc.) o -v: Verbose output o -z: Compress during transfer o --delete: Delete files in the destination that no longer exist in the source 29. alias - Create Command Aliases • Basic: alias name='command' • Flags: o unalias name: Remove an alias - create examples from scratch for all these commands and apply all these commands - with all the flags show examplese - first crate files and then apply all thse commands please -so that i can clearly understand

**ChatGPT said:**

Certainly! Let's break this down step by step with examples for each of the commands and their flags. I'll first create a set of files and directories for you to work with, and then I'll demonstrate each command with various flags.

**1. Create Files and Directories (Setup)**

Let's start by creating some sample files and directories.

bash

CopyEdit

# Create directories

mkdir -p /home/user/docs/folder1 /home/user/docs/folder2

# Create some files

touch /home/user/docs/file1.txt /home/user/docs/file2.txt /home/user/docs/file3.txt

# Create a hidden file

touch /home/user/docs/.hiddenfile

# Create a subdirectory inside folder1

mkdir /home/user/docs/folder1/subfolder

# Add some content to files

echo "Hello World!" > /home/user/docs/file1.txt

echo "This is a test file." > /home/user/docs/file2.txt

echo "Another test file." > /home/user/docs/file3.txt

Now that we have some files and directories, let's proceed with the command examples.

**2. ls - List Directory Contents**

bash

CopyEdit

# Basic listing

ls /home/user/docs

# Long format with human-readable sizes, including hidden files

ls -lha /home/user/docs

# Recursively list directories

ls -R /home/user/docs

# Sort by size

ls -lS /home/user/docs

# Sort by modification time

ls -lt /home/user/docs

# Colorized output

ls --color /home/user/docs

**3. cd - Change Directory**

bash

CopyEdit

# Change to a specific directory

cd /home/user/docs/folder1

# Go to home directory

cd ~

# Go back to previous directory

cd -

# Go up one level

cd ..

**4. pwd - Print Working Directory**

bash

CopyEdit

# Print the current working directory

pwd

**5. mkdir - Make Directory**

bash

CopyEdit

# Create a new directory

mkdir /home/user/docs/new\_folder

# Create parent directories if they don't exist

mkdir -p /home/user/docs/folder3/subfolder

# Show verbose output

mkdir -v /home/user/docs/new\_folder2

**6. rmdir - Remove Empty Directory**

bash

CopyEdit

# Remove an empty directory

rmdir /home/user/docs/folder3/subfolder

# Remove parent directories if they are empty

rmdir -p /home/user/docs/folder3

**7. rm - Remove Files or Directories**

bash

CopyEdit

# Remove a file

rm /home/user/docs/file1.txt

# Recursively remove a directory and its contents

rm -rf /home/user/docs/folder1

# Force remove a file without confirmation

rm -f /home/user/docs/file2.txt

# Prompt before removal

rm -i /home/user/docs/file3.txt

# Verbose output

rm -v /home/user/docs/.hiddenfile

**8. cp - Copy Files or Directories**

bash

CopyEdit

# Copy a file

cp /home/user/docs/file1.txt /home/user/docs/file1\_copy.txt

# Recursively copy directories

cp -r /home/user/docs/folder2 /home/user/docs/folder2\_copy

# Copy only if the source file is newer

cp -u /home/user/docs/file2.txt /home/user/docs/file2\_backup.txt

# Prompt before overwriting

cp -i /home/user/docs/file2.txt /home/user/docs/file1\_copy.txt

# Verbose output

cp -v /home/user/docs/file2.txt /home/user/docs/file2\_copy.txt

**9. mv - Move or Rename Files/Directories**

bash

CopyEdit

# Rename a file

mv /home/user/docs/file1\_copy.txt /home/user/docs/file1\_renamed.txt

# Move a file to another directory

mv /home/user/docs/file2.txt /home/user/docs/folder2

# Prompt before overwriting

mv -i /home/user/docs/file1\_renamed.txt /home/user/docs/folder2/file1\_renamed.txt

# Verbose output

mv -v /home/user/docs/file3.txt /home/user/docs/folder2

**10. touch - Create Empty Files or Change Timestamps**

bash

CopyEdit

# Create an empty file

touch /home/user/docs/newfile.txt

# Set a specific timestamp

touch -t 202304050730 /home/user/docs/newfile.txt

# Do not create a new file if it doesn't exist

touch -c /home/user/docs/nonexistentfile.txt

**11. find - Search for Files**

bash

CopyEdit

# Search for a file by name

find /home/user/docs -name "file2.txt"

# Search for directories only

find /home/user/docs -type d

# Execute a command on the found files

find /home/user/docs -name "\*.txt" -exec ls -l {} \;

# Search by modification time (files modified more than 7 days ago)

find /home/user/docs -mtime +7

# Search by file size (files larger than 100MB)

find /home/user/docs -size +100M

**12. cat - Concatenate and Display Files**

bash

CopyEdit

# Display file contents

cat /home/user/docs/file2.txt

# Number all lines

cat -n /home/user/docs/file2.txt

# Number non-empty lines

cat -b /home/user/docs/file2.txt

# Show $ at the end of each line

cat -E /home/user/docs/file2.txt

**13. more and less - View File Contents**

bash

CopyEdit

# View file contents with more

more /home/user/docs/file2.txt

# View file contents with less

less /home/user/docs/file2.txt

# Use flags for more

more -c /home/user/docs/file2.txt

# Use flags for less

less -N /home/user/docs/file2.txt

less -S /home/user/docs/file2.txt

**14. head - Show the Beginning of a File**

bash

CopyEdit

# Show the first 10 lines of a file (default)

head /home/user/docs/file2.txt

# Show the first 20 lines

head -n 20 /home/user/docs/file2.txt

**15. tail - Show the End of a File**

bash

CopyEdit

# Show the last 10 lines of a file (default)

tail /home/user/docs/file2.txt

# Show the last 20 lines

tail -n 20 /home/user/docs/file2.txt

# Continuously monitor the file

tail -f /home/user/docs/file2.txt

**16. chmod - Change File Permissions**

bash

CopyEdit

# Change file permissions (numeric mode)

chmod 755 /home/user/docs/file2.txt

# Add execute permission for the user

chmod u+x /home/user/docs/file2.txt

# Apply recursively to directories and files

chmod -R 755 /home/user/docs/folder2

**17. chown - Change File Ownership**

bash

CopyEdit

# Change ownership of a file

chown user:group /home/user/docs/file2.txt

# Apply recursively to directories and files

chown -R user:group /home/user/docs/folder2

**18. ln - Create Hard and Symbolic Links**

bash

CopyEdit

# Create a hard link

ln /home/user/docs/file2.txt /home/user/docs/file2\_hardlink.txt

# Create a symbolic (soft) link

ln -s /home/user/docs/file2.txt /home/user/docs/file2\_symlink.txt

**19. tar - Archive Files**

bash

CopyEdit

# Create an archive

tar -cf /home/user/docs/archive.tar /home/user/docs/file2.txt

# Extract files from an archive

tar -xf /home/user/docs/archive.tar

# Create a gzip-compressed archive

tar -czf /home/user/docs/archive.tar.gz /home/user/docs/file2.txt

# Create a bzip2-compressed archive

tar -cjf /home/user/docs/archive.tar.bz2 /home/user/docs/file2.txt

# Create an xz-compressed archive

tar -cJf /home/user/docs/archive.tar.xz /home/user/docs/file2.txt

**20. zip and unzip - Create and Extract ZIP Files**

bash

CopyEdit

# Create a ZIP archive

zip /home/user/docs/archive.zip /home/user/docs/file2.txt

# Extract a ZIP archive

unzip /home/user/docs/archive.zip -d /home/user/docs/

# Recursively zip directories

zip -r /home/user/docs/archive.zip /home/user/docs/folder2

# Extract to a specific directory

unzip /home/user/docs/archive.zip -d /home/user/docs/folder3/

**21. df - Display Disk Space Usage**

bash

CopyEdit

# Display disk space usage

df

# Display disk space in a human-readable format

df -h

# Show file system type

df -T

**22. du - Estimate File Space Usage**

bash

CopyEdit

# Show size of a file

du /home/user/docs/file2.txt

# Show total size of a directory

du -s /home/user/docs

# Show sizes of all files and directories

du -a /home/user/docs

# Human-readable format

du -h /home/user/docs

**23. stat - Display File Status**

bash

CopyEdit

# Show file status

stat /home/user/docs/file2.txt

# Show specific output (file size)

stat -c %s /home/user/docs/file2.txt

**24. sync - Synchronize Data to Disk**

bash

CopyEdit

# Synchronize data to disk

sync

**25. mount and umount - Mount and Unmount File Systems**

bash

CopyEdit

# Mount a device

mount /dev/sdX /mnt

# Unmount a device

umount /mnt

**26. file - Determine File Type**

bash

CopyEdit

# Determine the file type

file /home/user/docs/file2.txt

# Display MIME type

file -i /home/user/docs/file2.txt

**27. shred - Securely Delete Files**

bash

CopyEdit

# Securely delete a file

shred -u /home/user/docs/file2.txt

# Specify number of overwrite passes

shred -n 5 -u /home/user/docs/file2.txt

# Add final pass with zeros

shred -z -u /home/user/docs/file2.txt

**28. gzip and bzip2 - Compress and Decompress Files**

bash

CopyEdit

# Compress using gzip

gzip /home/user/docs/file2.txt

# Decompress using gzip

gzip -d /home/user/docs/file2.txt.gz

# Compress using bzip2

bzip2 /home/user/docs/file2.txt

# Decompress using bzip2

bzip2 -d /home/user/docs/file2.txt.bz2

**29. rsync - Synchronize Files and Directories**

bash

CopyEdit

# Sync files and directories

rsync -av /home/user/docs/folder2/ /home/user/docs/folder2\_copy/

# Sync with compression

rsync -avz /home/user/docs/folder2/ /home/user/docs/folder2\_copy/

# Delete files in destination that no longer exist in source

rsync -avz --delete /home/user/docs/folder2/ /home/user/docs/folder2\_copy/

**30. alias - Create Command Aliases**

bash

CopyEdit

# Create an alias

alias ll='ls -l'

# Remove an alias

unalias ll

This set of commands and their variations should provide a clear understanding of how to work with Linux command-line operations. You can practice with this by creating the files and directories, and then applying these commands to them.

1. top Purpose: Displays real-time system summary and process information. • top – Display system information (CPU, memory, processes). • top -u <user> – Show processes for a specific user. • top -p <pid> – Show specific process by PID. • top -d <seconds> – Set the update delay interval (in seconds). • top -n <count> – Show the output for a specified number of iterations. • top -c – Show full command lines of processes. 2. htop Purpose: Interactive process viewer (more user-friendly version of top). • htop – Start the interactive viewer. • htop -u <user> – Show processes for a specific user. • htop -s <field> – Sort processes by specific fields (e.g., CPU, memory). • htop -d <delay> – Adjust the delay between updates. 3. ps Purpose: Display information about active processes. • ps aux – Show all processes running on the system. • ps -ef – Show all processes with full command lines. • ps -u <user> – Show processes for a specific user. • ps -p <pid> – Show a specific process by PID. • ps -eo pid,comm,%cpu,%mem – Show specific fields (PID, command, CPU, memory). 4. vmstat Purpose: Report virtual memory statistics. • vmstat 1 – Report every second. • vmstat -s – Show memory statistics in a human-readable format. • vmstat -d – Report disk statistics. 5. free Purpose: Display memory usage. • free – Show memory usage (RAM and swap). • free -h – Show human-readable memory values (e.g., MB, GB). • free -m – Display memory in megabytes. • free -g – Display memory in gigabytes. 6. iostat Purpose: Display CPU and input/output statistics for devices. • iostat – Show CPU and I/O statistics. • iostat -d – Show device statistics. • iostat -c – Show CPU statistics. • iostat -x – Show extended device statistics. 7. sar Purpose: Collect and report system activity information. • sar -u 1 3 – Show CPU usage every 1 second, 3 times. • sar -r – Show memory usage. • sar -d – Show disk activity. • sar -n DEV – Show network activity. 8. dstat Purpose: Versatile resource statistics tool. • dstat – Show CPU, disk, network, and I/O stats. • dstat -c – Show only CPU stats. • dstat -d – Show only disk stats. • dstat -n – Show only network stats. • dstat -t – Show time stats. 9. netstat Purpose: Show network connections, routing tables, interface statistics. • netstat -tuln – Show active listening ports. • netstat -tulpn – Show active connections with the associated program. • netstat -i – Show network interface statistics. • netstat -rn – Show routing tables. • netstat -s – Show network statistics. 10. ss Purpose: Utility to investigate sockets. • ss – Show all sockets. • ss -tuln – Show listening sockets. • ss -p – Show process using the socket. • ss -s – Show summary of socket statistics. 11. uptime Purpose: Show how long the system has been running. • uptime – Displays the uptime, number of users, and load averages. 12. lsof Purpose: List open files and processes using them. • lsof – Show all open files. • lsof -i – Show all network connections. • lsof -u <user> – Show files opened by a specific user. • lsof +D <directory> – Show files opened in a specific directory. 13. df Purpose: Report file system disk space usage. • df – Show disk space usage for mounted filesystems. • df -h – Human-readable format (e.g., MB, GB). • df -T – Show file system types. • df -i – Show inode usage. 14. du Purpose: Estimate file space usage. • du – Show the disk usage of files and directories. • du -sh <directory> – Show total disk usage of a directory in human-readable format. • du -ah – Show disk usage of all files and directories. 15. watch Purpose: Execute a program periodically and show output. • watch <command> – Run a command every 2 seconds by default. • watch -n <seconds> <command> – Run the command at custom intervals. • watch -d <command> – Highlight changes in the output. 16. lscpu Purpose: Display information about the CPU architecture. • lscpu – Show detailed CPU architecture information. 17. lsblk Purpose: List information about block devices (disks). • lsblk – List all block devices. • lsblk -f – Show file system information. • lsblk -o NAME,SIZE,TYPE,MOUNTPOINT – Custom output columns. 18. systemctl Purpose: Control the systemd system and service manager. • systemctl status – Show system and service manager status. • systemctl status <service> – Show specific service status. • systemctl list-units --type=service – List all active services. • systemctl start <service> – Start a service. • systemctl stop <service> – Stop a service. 19. journalctl Purpose: Query and display logs from systemd journal. • journalctl – Show all system logs. • journalctl -u <service> – Show logs for a specific service. • journalctl -f – Follow the log output. • journalctl --since "2025-04-05" – Show logs since a specific date. 20. atop Purpose: Advanced system and process monitoring tool. • atop – Start the tool to monitor system and process activities. • atop -r <file> – Read a log file recorded by atop. • atop -d <seconds> – Update every x seconds. 21. sar (Sysstat) Purpose: Collect and report system activity information. • sar -u 1 3 – Report CPU utilization every second, three times. • sar -r – Show memory statistics. • sar -d – Show disk statistics. 22. tcpdump Purpose: Network traffic capture. • tcpdump – Capture network traffic. • tcpdump -i <interface> – Capture traffic from a specific network interface. • tcpdump -n – Don't resolve hostnames. • tcpdump -v – Increase verbosity. 23. iftop Purpose: Display bandwidth usage on an interface. • iftop – Show bandwidth usage on each connection. • iftop -ni <interface> – Show bandwidth on a specific interface. 24. uptime Purpose: Show system uptime and load averages. • uptime – Displays the system uptime and load averages. 25. mpstat Purpose: Report CPU statistics. • mpstat – Show CPU stats for each processor. • mpstat -P ALL – Show stats for all processors. These commands give a powerful toolkit for monitoring various aspects of system performance, whether you're looking at system load, memory usage, disk activity, network stats, or process information. You can combine them, use them in scripts, or simply run them in real-time for a more detailed view of your system. You said: continue please with more system monitoring cmd advance cmds please that are used by devops engineer ChatGPT said: Here are more advanced system monitoring commands commonly used by DevOps Engineers for deeper insights into system performance, troubleshooting, and infrastructure management. These tools focus on monitoring, debugging, and enhancing the reliability and performance of systems and applications in production environments. 26. strace Purpose: Trace system calls and signals. • strace -p <pid> – Attach to a running process by its PID and trace system calls. • strace -e trace=<syscall> – Trace a specific system call (e.g., read, write). • strace -o <file> – Save the output to a file. • strace -f – Trace child processes (forks). • strace -c – Count time, calls, and errors for each system call. 27. dmesg Purpose: Print kernel ring buffer messages, typically for debugging hardware issues. • dmesg – Display kernel messages. • dmesg | grep <pattern> – Filter messages based on a pattern. • dmesg -T – Show human-readable timestamps. • dmesg --clear – Clear the kernel ring buffer. 28. sysctl Purpose: Interface to change kernel parameters at runtime. • sysctl -a – Display all available kernel parameters. • sysctl -w <parameter>=<value> – Set a specific kernel parameter. • sysctl net.ipv4.ip\_forward=1 – Enable IP forwarding (e.g., for routing). • sysctl -p – Load settings from /etc/sysctl.conf or a configuration file. 29. pidstat Purpose: Collect and display statistics by process. • pidstat – Show CPU, memory, and other statistics for processes. • pidstat -u – Display CPU usage by process. • pidstat -r – Show memory usage by process. • pidstat -d – Show I/O statistics for processes. • pidstat -p <pid> – Display stats for a specific process. 30. netcat (nc) Purpose: Network utility for reading from and writing to network connections. • nc -l <port> – Start listening on a specific port. • nc <host> <port> – Connect to a remote host and port. • nc -zv <host> <port-range> – Scan a range of ports on a remote host. • nc -w 5 <host> <port> – Set a timeout of 5 seconds for a connection. 31. curl Purpose: Transfer data from or to a server. • curl -I <URL> – Show only the HTTP header information. • curl -O <URL> – Download a file. • curl -L <URL> – Follow redirects. • curl -u <user>:<password> <URL> – Make an authenticated request. • curl -X POST <URL> -d "<data>" – Make a POST request with data. • curl --silent --show-error <URL> – Suppress output except for errors. 32. ncdu Purpose: Disk usage analyzer with a user-friendly interface. • ncdu – Display disk usage and allow interactive navigation. • ncdu <path> – Analyze disk usage for a specific path or directory. 33. tshark Purpose: Network protocol analyzer (similar to tcpdump, but with more features). • tshark – Start capturing packets. • tshark -i <interface> – Capture packets from a specific interface. • tshark -f "<filter>" – Capture only specific packets based on a filter (e.g., tcp port 80). • tshark -w <file> – Save captured packets to a file. 34. tcpdump Purpose: Capture network traffic and troubleshoot networking issues. • tcpdump -i <interface> – Capture packets on a specified network interface. • tcpdump -s 0 -A – Capture and display full packet contents (useful for HTTP traffic). • tcpdump -n – Don't resolve hostnames (show IP addresses). • tcpdump -v – Verbose output (more detailed packet info). • tcpdump -X – Show packet contents in hexadecimal and ASCII. • tcpdump 'port 80' – Capture HTTP traffic. 35. bpfcc / bpftrace Purpose: Tools for tracing and monitoring Linux kernel events with eBPF (Extended Berkeley Packet Filter). • bpfcc – Use BPF tools to trace and debug system performance. • bpftrace – Dynamic tracing tool that allows custom tracing scripts. • bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { @[comm] = count(); }' – Trace the open() system call. • bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }' – Trace file reads. 36. inotifywait Purpose: Monitor file system events using inotify. • inotifywait -m <directory> – Monitor events in a directory. • inotifywait -r – Monitor directories recursively. • inotifywait -e <event> <file> – Monitor a specific event (e.g., create, delete). - create examples from scratch for all these commands and apply all these commands - with all the flags show examplese - first create and then apply all thse commands please -so that i can clearly understand

**ChatGPT said:**

To fully understand these advanced system monitoring commands and their usage in a real-world scenario, we'll start by creating some files, processes, and system configurations. Afterward, I'll demonstrate how to use each of the commands, explaining their flags and options with examples.

**Step 1: Create Files, Directories, and Processes for Testing**

Let's first set up some files and processes. We will also simulate some network activity.

bash

CopyEdit

# Create directories and files

mkdir -p /home/user/monitoring\_test/{dir1,dir2}

touch /home/user/monitoring\_test/file1.txt /home/user/monitoring\_test/file2.txt

echo "This is a test file for monitoring" > /home/user/monitoring\_test/file1.txt

echo "Network test file" > /home/user/monitoring\_test/file2.txt

# Create a background process (e.g., sleep)

sleep 100 & # Get the PID of this process using `ps`

# Simulate a few more processes (in a loop for testing)

for i in {1..5}; do

sleep 60 &

done

Now that we have files and processes to monitor, let's proceed with each of the commands.

**1. top - Real-time System and Process Information**

bash

CopyEdit

# Basic usage of top to display system information

top

# Display processes for a specific user (replace with your actual username)

top -u user

# Show a specific process by its PID (replace with an actual PID from `ps`)

top -p 12345

# Set the update delay interval to 1 second

top -d 1

# Show the top output for 5 iterations

top -n 5

# Show full command lines of processes

top -c

**2. htop - Interactive Process Viewer**

bash

CopyEdit

# Launch the interactive viewer

htop

# Show processes for a specific user

htop -u user

# Sort processes by a specific field, e.g., CPU usage

htop -s PERCENT\_CPU

# Set the delay between updates (e.g., 2 seconds)

htop -d 2

**3. ps - Display Information About Active Processes**

bash

CopyEdit

# Show all processes running on the system

ps aux

# Show all processes with full command lines

ps -ef

# Show processes for a specific user

ps -u user

# Show a specific process by its PID

ps -p 12345

# Display specific fields like PID, command, CPU, and memory

ps -eo pid,comm,%cpu,%mem

**4. vmstat - Report Virtual Memory Statistics**

bash

CopyEdit

# Report every second

vmstat 1

# Show memory statistics in a human-readable format

vmstat -s

# Report disk statistics

vmstat -d

**5. free - Display Memory Usage**

bash

CopyEdit

# Show memory usage (RAM and swap)

free

# Show human-readable memory values (e.g., MB, GB)

free -h

# Show memory in megabytes

free -m

# Show memory in gigabytes

free -g

**6. iostat - Display CPU and I/O Statistics for Devices**

bash

CopyEdit

# Show CPU and I/O statistics

iostat

# Show device statistics

iostat -d

# Show CPU statistics only

iostat -c

# Show extended device statistics

iostat -x

**7. sar - Collect and Report System Activity Information**

bash

CopyEdit

# Show CPU usage every second, 3 times

sar -u 1 3

# Show memory usage

sar -r

# Show disk activity

sar -d

# Show network activity

sar -n DEV

**8. dstat - Versatile Resource Statistics Tool**

bash

CopyEdit

# Show CPU, disk, network, and I/O stats

dstat

# Show only CPU stats

dstat -c

# Show only disk stats

dstat -d

# Show only network stats

dstat -n

# Show time stats

dstat -t

**9. netstat - Show Network Connections and Routing Tables**

bash

CopyEdit

# Show active listening ports

netstat -tuln

# Show active connections with the associated program

netstat -tulpn

# Show network interface statistics

netstat -i

# Show routing tables

netstat -rn

# Show network statistics

netstat -s

**10. ss - Investigate Sockets**

bash

CopyEdit

# Show all sockets

ss

# Show listening sockets

ss -tuln

# Show processes using the sockets

ss -p

# Show summary of socket statistics

ss -s

**11. uptime - Show System Uptime**

bash

CopyEdit

# Display the uptime, number of users, and load averages

uptime

**12. lsof - List Open Files and Processes Using Them**

bash

CopyEdit

# Show all open files

lsof

# Show all network connections

lsof -i

# Show files opened by a specific user

lsof -u user

# Show files opened in a specific directory

lsof +D /home/user/monitoring\_test

**13. df - Report File System Disk Space Usage**

bash

CopyEdit

# Show disk space usage for mounted filesystems

df

# Show disk space in a human-readable format

df -h

# Show file system types

df -T

# Show inode usage

df -i

**14. du - Estimate File Space Usage**

bash

CopyEdit

# Show the disk usage of files and directories

du

# Show total disk usage of a directory in human-readable format

du -sh /home/user/monitoring\_test

# Show disk usage of all files and directories

du -ah /home/user/monitoring\_test

**15. watch - Execute a Program Periodically**

bash

CopyEdit

# Run a command every 2 seconds (default)

watch df

# Run the command at custom intervals (e.g., every 5 seconds)

watch -n 5 df

# Highlight changes in the output

watch -d df

**16. lscpu - Display Information About the CPU Architecture**

bash

CopyEdit

# Show detailed CPU architecture information

lscpu

**17. lsblk - List Information About Block Devices**

bash

CopyEdit

# List all block devices

lsblk

# Show file system information

lsblk -f

# Show custom output columns (e.g., NAME, SIZE, TYPE, MOUNTPOINT)

lsblk -o NAME,SIZE,TYPE,MOUNTPOINT

**18. systemctl - Control the systemd System and Service Manager**

bash

CopyEdit

# Show system and service manager status

systemctl status

# Show specific service status (e.g., nginx)

systemctl status nginx

# List all active services

systemctl list-units --type=service

# Start a service (e.g., nginx)

systemctl start nginx

# Stop a service (e.g., nginx)

systemctl stop nginx

**19. journalctl - Query and Display Logs from Systemd Journal**

bash

CopyEdit

# Show all system logs

journalctl

# Show logs for a specific service (e.g., nginx)

journalctl -u nginx

# Follow the log output

journalctl -f

# Show logs since a specific date

journalctl --since "2025-04-05"

**20. atop - Advanced System and Process Monitoring Tool**

bash

CopyEdit

# Start the tool to monitor system and process activities

atop

# Read a log file recorded by atop

atop -r /path/to/logfile

# Update every 2 seconds

atop -d 2

**21. strace - Trace System Calls and Signals**

bash

CopyEdit

# Attach to a running process by its PID and trace system calls

strace -p 12345

# Trace a specific system call (e.g., read, write)

strace -e trace=read,write -p 12345

# Save the output to a file

strace -o strace\_output.txt -p 12345

# Trace child processes (forks)

strace -f -p 12345

# Count time, calls, and errors for each system call

strace -c -p 12345

**22. dmesg - Print Kernel Ring Buffer Messages**

bash

CopyEdit

# Display kernel messages

dmesg

# Filter messages based on a pattern

dmesg | grep "error"

# Show human-readable timestamps

dmesg -T

# Clear the kernel ring buffer

dmesg --clear

**23. sysctl - Interface to Change Kernel Parameters at Runtime**

bash

CopyEdit

# Display all available kernel parameters

sysctl -a

# Set a specific kernel parameter (e.g., enable IP forwarding)

sysctl -w net.ipv4.ip\_forward=1

# Load settings from /etc/sysctl.conf

sysctl -p

**24. pidstat - Collect and Display Statistics by Process**

bash

CopyEdit

# Show CPU, memory, and other statistics for processes

pidstat

# Display CPU usage by process

pidstat -u

# Show memory usage by process

pidstat -r

# Show I/O statistics for processes

pidstat -d

# Display stats for a specific process by PID

pidstat -p 12345

**25. netcat (nc) - Network Utility for Reading/Writing to Network Connections**

bash

CopyEdit

# Start listening on a specific port

nc -l 8080

# Connect to a remote host and port

nc example.com 80

# Scan a range of ports on a remote host

nc -zv example.com 1-1000

# Set a timeout of 5 seconds for a connection

nc -w 5 example.com 80

**26. curl - Transfer Data from or to a Server**

bash

CopyEdit

# Show only the HTTP header information

curl -I https://example.com

# Download a file

curl -O https://example.com/file.zip

# Follow redirects

curl -L https://example.com

# Make an authenticated request

curl -u username:password https://example.com

# Make a POST request with data

curl -X POST https://example.com -d "key=value"

# Suppress output except for errors

curl --silent --show-error https://example.com

**27. ncdu - Disk Usage Analyzer**

bash

CopyEdit

# Display disk usage and allow interactive navigation

ncdu

# Analyze disk usage for a specific path or directory

ncdu /home/user/monitoring\_test

**28. tshark - Network Protocol Analyzer**

bash

CopyEdit

# Start capturing packets

tshark

# Capture packets from a specific interface

tshark -i eth0

# Capture only specific packets (e.g., TCP port 80)

tshark -f "tcp port 80"

# Save captured packets to a file

tshark -w capture.pcap

**29. tcpdump - Capture Network Traffic**

bash

CopyEdit

# Capture packets on a specified network interface

tcpdump -i eth0

# Capture and display full packet contents (useful for HTTP traffic)

tcpdump -s 0 -A

# Don't resolve hostnames (show IP addresses)

tcpdump -n

# Verbose output (more detailed packet info)

tcpdump -v

# Show packet contents in hexadecimal and ASCII

tcpdump -X

# Capture HTTP traffic

tcpdump 'port 80'

**30. bpfcc / bpftrace - Tools for Tracing and Monitoring Linux Kernel Events**

bash

CopyEdit

# Use BPF tools to trace and debug system performance

bpfcc

# Trace the open() system call

bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { @[comm] = count(); }'

# Trace file reads

bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }'

**31. inotifywait - Monitor File System Events Using Inotify**

bash

CopyEdit

# Monitor events in a directory

inotifywait -m /home/user/monitoring\_test

# Monitor directories recursively

inotifywait -r -m /home/user/monitoring\_test

# Monitor a specific event (e.g., file creation)

inotifywait -e create /home/user/monitoring\_test

1. ifconfig Purpose: Display and configure network interfaces. • ifconfig – Show all network interfaces and their IP addresses. • ifconfig <interface> – Show specific interface (e.g., eth0, wlan0). • ifconfig <interface> up – Bring up a network interface. • ifconfig <interface> down – Bring down a network interface. • ifconfig <interface> <IP> – Assign an IP address to an interface. • ifconfig <interface> netmask <netmask> – Set the netmask for the interface. 2. ip Purpose: A more modern and powerful alternative to ifconfig for network management. • ip a – Show all network interfaces and their IP addresses. • ip link show – Display link information (e.g., interface status). • ip addr show – Show IP addresses assigned to all interfaces. • ip link set <interface> up – Bring the interface up. • ip link set <interface> down – Bring the interface down. • ip addr add <IP>/24 dev <interface> – Assign a new IP to an interface. • ip route show – Display the routing table. • ip route add <destination> via <gateway> – Add a new route. 3. ping Purpose: Send ICMP echo requests to check network connectivity. • ping <hostname or IP> – Ping a host by its IP address or hostname. • ping -c <count> <hostname> – Send a specific number of packets. • ping -i <interval> <hostname> – Set the interval between packets. • ping -t <ttl> <hostname> – Set the TTL (Time To Live) for packets. • ping -s <size> <hostname> – Send packets of a specific size. • ping -4 <hostname> – Force the use of IPv4. • ping -6 <hostname> – Force the use of IPv6. 4. traceroute Purpose: Trace the route packets take to a destination. • traceroute <hostname> – Trace the route to a remote host. • traceroute -m <max\_hops> <hostname> – Set the maximum number of hops. • traceroute -T – Use TCP packets for tracing. • traceroute -I – Use ICMP packets for tracing. 5. nslookup Purpose: Query DNS to obtain domain name or IP address information. • nslookup <hostname> – Query the IP address of a given hostname. • nslookup <IP> – Query the domain name for a given IP. • nslookup -type=MX <domain> – Query MX (mail) records for a domain. • nslookup -type=NS <domain> – Query the name servers for a domain. 6. dig Purpose: Perform DNS lookups and query DNS servers. • dig <hostname> – Perform a basic DNS lookup for a domain. • dig @<DNS server> <hostname> – Query a specific DNS server. • dig <hostname> A – Query A (address) record for a domain. • dig <hostname> MX – Query MX (mail) record for a domain. • dig +short <hostname> – Shorten output to show only the result. 7. route Purpose: View and configure network routing tables. • route – Show the current routing table. • route -n – Show the routing table with numeric addresses (no DNS resolution). • route add <destination> gw <gateway> – Add a new route to a destination. • route del <destination> – Delete a specific route. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Advanced Networking Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 8. netstat Purpose: Display network connections, routing tables, interface statistics, etc. • netstat – Show active connections and listening ports. • netstat -tuln – Display TCP and UDP ports that are open and listening. • netstat -an – Display all network connections (IP and port). • netstat -s – Show network statistics. • netstat -i – Show network interfaces and packet statistics. • netstat -r – Show the routing table. 9. ss Purpose: Utility to investigate sockets. • ss – Display all sockets. • ss -tuln – Display listening TCP and UDP sockets. • ss -a – Show all sockets (listening and non-listening). • ss -p – Display processes using the sockets. • ss -r – Display routing information. 10. netcat (nc) Purpose: A versatile tool for network debugging and banner grabbing. • nc -zv <hostname> <port> – Scan a specific port or range of ports. • nc -l <port> – Listen on a specific port for incoming connections. • nc <hostname> <port> – Connect to a host on a specific port. • nc -u <hostname> <port> – Use UDP instead of TCP. • nc -v <hostname> <port> – Verbose output for connection details. 11. tcpdump Purpose: Capture network traffic and display the packets. • tcpdump – Capture packets on the default interface. • tcpdump -i <interface> – Capture on a specific network interface. • tcpdump -n – Do not resolve hostnames or port names. • tcpdump -v – Show more verbose packet information. • tcpdump -c <count> – Capture a specific number of packets. • tcpdump -w <file> – Save the captured packets to a file. • tcpdump 'port 80' – Capture HTTP traffic. • tcpdump -A – Display packet contents in ASCII format. • tcpdump -X – Display packet contents in both hex and ASCII. 12. iftop Purpose: Display bandwidth usage on a network interface in real-time. • iftop – Display real-time bandwidth usage. • iftop -i <interface> – Specify the network interface to monitor. • iftop -n – Show numerical addresses (do not resolve hostnames). • iftop -P – Show port numbers along with IP addresses. 13. nmap Purpose: Network exploration and security auditing tool. • nmap <hostname> – Perform a simple scan of a host. • nmap -sP <IP-range> – Perform a ping scan to identify live hosts. • nmap -p <ports> <hostname> – Scan specific ports (e.g., -p 80,443). • nmap -sS <hostname> – Perform a SYN scan (stealth scan). • nmap -O <hostname> – OS detection. • nmap -A <hostname> – Enable OS detection, version detection, script scanning, and traceroute. 14. mtr Purpose: A network diagnostic tool combining traceroute and ping. • mtr <hostname> – Continuous traceroute and ping to a destination. • mtr -r <hostname> – Generate a report instead of continuous output. • mtr -c <count> <hostname> – Set the number of pings to send. 15. arp Purpose: View and manipulate the system's ARP (Address Resolution Protocol) cache. • arp -a – Display the current ARP table. • arp -d <IP> – Delete an entry from the ARP table. • arp -s <IP> <MAC> – Manually add a static entry to the ARP table. 16. ethtool Purpose: Query and control network device settings. • ethtool <interface> – Display information about a network interface. • ethtool -s <interface> speed <speed> duplex <duplex> – Change the speed and duplex settings of a network interface. • ethtool -p <interface> – Blink the LED on a network interface to help locate it. 17. iwconfig Purpose: Configure wireless network interfaces. • iwconfig – Display wireless network information. • iwconfig <interface> essid <SSID> – Connect to a specific Wi-Fi network. • iwconfig <interface> mode <mode> – Set the mode of the wireless interface (e.g., managed, ad-hoc). 18. ip link Purpose: Display or modify network device attributes. • ip link show – Display detailed information about network interfaces. • ip link set <interface> up/down – Bring the interface up or down. • ip link set <interface> mtu <mtu\_size> – Change the MTU size for an interface. 19. sshd (SSH Daemon) Purpose: Manage SSH access and configurations. • sshd -T – Show current configuration options. • sshd -f <config\_file> – Use a specific SSH configuration file. • sshd -D – Run in the foreground (useful for debugging). \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Cloud/Container Specific Networking Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20. docker network Purpose: Manage Docker network settings. • docker network ls – List all Docker networks. • docker network inspect <network\_name> – Inspect a specific Docker network. • docker network create <network\_name> – Create a new Docker network. • docker network connect <network\_name> <container\_name> – Connect a container to a network. 21. kubectl get svc Purpose: Retrieve the details of Kubernetes services. • kubectl get svc – List all services in the current namespace. • kubectl get svc <service\_name> – Get details of a specific service. • kubectl describe svc <service\_name> – Detailed information on the service, including networking details. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ These commands, along with their options and flags, are crucial for diagnosing, configuring, and optimizing networking in any system or infrastructure. DevOps Engineers often use them to ensure reliable network performance, security, and troubleshooting in both on-premise and cloud-based environments. You said: please continue with more advance linux networking cmds used by devops engineer ChatGPT said: Here’s a continuation of advanced Linux networking commands that are typically used by DevOps Engineers for deep network diagnostics, configuration, performance monitoring, and troubleshooting. These commands are important when working with complex network setups, performance tuning, and cloud or containerized environments. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 22. ss (Socket Stat) Purpose: A more modern and faster alternative to netstat for analyzing socket connections. • ss -tuln – Display all listening TCP and UDP sockets. • ss -an – Show all network connections in numeric form (no DNS resolution). • ss -p – Show the processes associated with each socket. • ss -s – Display socket statistics (summary). • ss -o state established – Show only established connections. • ss -r – Display routing information along with socket statistics. • ss -A – Show application layer statistics. 23. nft (Netfilter) Purpose: Interface for configuring packet filtering and network address translation (NAT). • nft list ruleset – Display all the rules in the firewall. • nft add rule inet filter input tcp dport 22 accept – Add a rule to allow incoming SSH connections. • nft flush ruleset – Flush all rules from the ruleset. • nft delete rule inet filter input handle 4 – Delete a specific rule by its handle number. • nft list table inet filter – List the rules in the filter table. • nft add chain inet filter input { type filter hook input priority 0 \; } – Add a chain to filter incoming packets. 24. iptables Purpose: Configure packet filtering rules in the Linux kernel firewall. • iptables -L – List all active firewall rules. • iptables -A INPUT -p tcp --dport 80 -j ACCEPT – Allow incoming HTTP traffic. • iptables -A INPUT -p tcp --dport 443 -j ACCEPT – Allow incoming HTTPS traffic. • iptables -A INPUT -p tcp --dport 22 -s <IP> -j ACCEPT – Allow SSH traffic from a specific IP. • iptables -D INPUT -p tcp --dport 80 -j ACCEPT – Remove the rule allowing HTTP traffic. • iptables -F – Flush all rules in the filter table. • iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.2:8080 – Redirect HTTP traffic to a different destination. 25. firewalld Purpose: Manage firewall rules dynamically with firewalld. • firewall-cmd --list-all – Show the current firewall settings and rules. • firewall-cmd --zone=public --add-port=8080/tcp --permanent – Open port 8080 permanently. • firewall-cmd --reload – Reload the firewall to apply new rules. • firewall-cmd --zone=public --remove-port=8080/tcp --permanent – Close port 8080. • firewall-cmd --list-ports – Show open ports. 26. bpftrace Purpose: Dynamic tracing tool for observing and troubleshooting kernel-level events using BPF (Berkeley Packet Filter). • bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { printf("%s opened\n", str(args->filename)); }' – Trace all open system calls. • bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }' – Trace file reads. • bpftrace -e 'tracepoint:net:net\_dev\_xmit { @[args->dev\_name] = count(); }' – Count network transmissions for each device. 27. tcpdump with Advanced Filtering Purpose: Advanced packet capture with specific filters. • tcpdump -i eth0 port 443 – Capture packets on port 443 (HTTPS). • tcpdump -i eth0 'tcp[tcpflags] & tcp-push != 0' – Capture TCP packets with the PUSH flag set. • tcpdump -i eth0 'src host <IP>' – Capture packets from a specific source IP. • tcpdump -i eth0 'dst host <IP>' – Capture packets destined to a specific IP. • tcpdump -i eth0 -w capture.pcap – Write captured packets to a file. • tcpdump -r capture.pcap – Read packets from a capture file. • tcpdump -i eth0 -v – Increase verbosity of the output. 28. ip rule Purpose: Used to configure policy-based routing. • ip rule show – Display the current routing rules. • ip rule add from <IP> table <table\_id> – Add a rule to route traffic from a specific IP through a custom routing table. • ip rule delete from <IP> – Delete the rule for traffic from a specific IP. • ip rule add iif <interface> table <table\_id> – Route traffic coming from a specific interface through a particular table. 29. tc (Traffic Control) Purpose: Control network traffic on interfaces. • tc qdisc show – Show the current queuing disciplines (QDisc) for interfaces. • tc qdisc add dev eth0 root netem delay 100ms – Add network delay to interface eth0. • tc qdisc change dev eth0 root netem loss 5% – Simulate 5% packet loss on eth0. • tc class add dev eth0 parent 1:1 classid 1:10 htb rate 1mbit – Add a traffic class with a rate limit. • tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 80 0xffff flowid 1:10 – Filter HTTP traffic and assign it to a specific traffic class. 30. ethtool (Advanced) Purpose: Query and modify network interface driver and hardware settings. • ethtool eth0 – Display settings for the eth0 network interface. • ethtool -s eth0 speed 1000 duplex full – Set the speed and duplex mode for the interface. • ethtool -p eth0 – Blink the LED to locate the physical interface. • ethtool -i eth0 – Display driver information for the network interface. • ethtool -S eth0 – Show statistics about the network interface. 31. nmcli Purpose: Command-line interface for NetworkManager (useful for managing network connections). • nmcli connection show – List all active network connections. • nmcli device status – Show the status of all network devices. • nmcli connection add type ethernet con-name eth0 ifname eth0 – Add a new Ethernet connection. • nmcli device disconnect eth0 – Disconnect a specific network interface. • nmcli device wifi list – List available Wi-Fi networks. 32. conntrack Purpose: Used to view and manage the connection tracking subsystem in the Linux kernel. • conntrack -L – List all tracked connections. • conntrack -F – Flush all tracked connections. • conntrack -D – Delete a specific connection from the connection table. • conntrack -L -p tcp – List all tracked TCP connections. • conntrack -E – Display connection events in real-time. 33. wireshark Purpose: A GUI-based network protocol analyzer (often used in tandem with tcpdump for deep packet inspection). • Launch Wireshark and capture packets from a specific interface. • wireshark – Start Wireshark GUI. • wireshark -i <interface> – Start Wireshark and capture packets from a specific interface. • Wireshark supports advanced filtering using display filters (e.g., http, ip.addr==192.168.1.1). 34. bridge Purpose: Configure and manage network bridges. • bridge link – Show the status of all network bridges. • bridge fdb show – Show the Forwarding Database (FDB) for a bridge. • bridge vlan – Show or configure VLANs on a bridge. • bridge addbr br0 – Create a new network bridge. • bridge addif br0 eth0 – Add an interface to the bridge. 35. bmon Purpose: Bandwidth monitor and rate estimator for Linux. • bmon – Display bandwidth usage in real-time. • bmon -p – Display statistics for all interfaces. • bmon -c – Show detailed statistics in a continuous mode. - create examples from scratch for all these commands and apply all these commands - with all the flags show examplse - first create and then apply all thse commands please -so that i can clearly understand

**ChatGPT said:**

Let's break down the advanced Linux networking commands, step-by-step, to understand how to use them effectively in a real-world scenario. We'll create some basic setup for network interfaces, traffic, and connections, and then apply each of the commands and their flags to monitor, configure, and troubleshoot network operations.

**Step 1: Network Setup and Configuration for Testing**

We'll begin by creating some basic configurations for network interfaces and simulate traffic.

bash

CopyEdit

# Create network interfaces (use actual interface names in your environment, e.g., eth0 or wlan0)

# Let's assume the interfaces are already up

# (For example, using eth0 and wlan0 in the tests)

# Set static IPs for testing

sudo ifconfig eth0 192.168.1.100 netmask 255.255.255.0 up

sudo ifconfig wlan0 192.168.1.101 netmask 255.255.255.0 up

# Bring down the interfaces to simulate changes

sudo ifconfig eth0 down

sudo ifconfig wlan0 down

# Bring them back up

sudo ifconfig eth0 up

sudo ifconfig wlan0 up

# Test with a dummy interface to simulate IP assignment

sudo ifconfig dummy0 192.168.1.102 netmask 255.255.255.0 up

# Create some traffic (ping the gateway or another device in your network)

ping 192.168.1.1 -c 4

Now that we have the setup, let's move to each command and apply them one by one.

**1. ifconfig - Display and Configure Network Interfaces**

bash

CopyEdit

# Show all network interfaces and their IP addresses

ifconfig

# Show specific interface information (e.g., eth0)

ifconfig eth0

# Bring up a network interface

ifconfig eth0 up

# Bring down a network interface

ifconfig eth0 down

# Assign an IP address to an interface

ifconfig eth0 192.168.1.200

# Set the netmask for the interface

ifconfig eth0 netmask 255.255.255.0

**2. ip - A Modern Alternative for Network Management**

bash

CopyEdit

# Show all network interfaces and their IP addresses

ip a

# Display link information (status of the interfaces)

ip link show

# Show IP addresses assigned to all interfaces

ip addr show

# Bring an interface up

ip link set eth0 up

# Bring an interface down

ip link set eth0 down

# Assign a new IP to an interface

ip addr add 192.168.1.201/24 dev eth0

# Display the routing table

ip route show

# Add a new route

ip route add 10.0.0.0/24 via 192.168.1.1

**3. ping - Send ICMP Echo Requests**

bash

CopyEdit

# Ping a host by IP address

ping 192.168.1.1

# Ping a host by hostname

ping google.com

# Send a specific number of packets (e.g., 4 packets)

ping -c 4 google.com

# Set the interval between packets (e.g., 1 second)

ping -i 1 google.com

# Set the TTL (Time To Live) for packets

ping -t 64 google.com

# Send packets of a specific size (e.g., 128 bytes)

ping -s 128 google.com

# Force the use of IPv4

ping -4 google.com

# Force the use of IPv6

ping -6 google.com

**4. traceroute - Trace the Route of Packets**

bash

CopyEdit

# Trace the route to a remote host

traceroute google.com

# Set the maximum number of hops

traceroute -m 30 google.com

# Use TCP packets for tracing

traceroute -T google.com

# Use ICMP packets for tracing

traceroute -I google.com

**5. nslookup - Query DNS for Domain or IP Information**

bash

CopyEdit

# Query the IP address of a given hostname

nslookup google.com

# Query the domain name for a given IP address

nslookup 8.8.8.8

# Query MX records for a domain

nslookup -type=MX example.com

# Query the name servers for a domain

nslookup -type=NS example.com

**6. dig - Perform DNS Lookups**

bash

CopyEdit

# Perform a basic DNS lookup for a domain

dig google.com

# Query a specific DNS server

dig @8.8.8.8 google.com

# Query A (address) record for a domain

dig google.com A

# Query MX (mail) record for a domain

dig google.com MX

# Shorten output to show only the result

dig +short google.com

**7. route - View and Configure Routing Tables**

bash

CopyEdit

# Show the current routing table

route

# Show the routing table with numeric addresses (no DNS resolution)

route -n

# Add a new route to a destination

route add -net 10.0.0.0 netmask 255.255.255.0 gw 192.168.1.1

# Delete a specific route

route del -net 10.0.0.0 netmask 255.255.255.0

**8. netstat - Display Network Connections, Routing Tables, etc.**

bash

CopyEdit

# Show active connections and listening ports

netstat

# Display TCP and UDP ports that are open and listening

netstat -tuln

# Display all network connections with IP and port

netstat -an

# Show network statistics

netstat -s

# Show network interfaces and packet statistics

netstat -i

# Show the routing table

netstat -r

**9. ss - Analyze Socket Connections (Faster than netstat)**

bash

CopyEdit

# Display all sockets

ss

# Display listening TCP and UDP sockets

ss -tuln

# Show all sockets (listening and non-listening)

ss -a

# Display processes using the sockets

ss -p

# Display routing information along with socket statistics

ss -r

# Show application layer statistics

ss -A

**10. netcat (nc) - Network Debugging and Banner Grabbing**

bash

CopyEdit

# Scan a specific port or range of ports

nc -zv 192.168.1.1 80-443

# Listen on a specific port for incoming connections

nc -l 8080

# Connect to a host on a specific port

nc 192.168.1.1 80

# Use UDP instead of TCP

nc -u 192.168.1.1 1234

# Verbose output for connection details

nc -v 192.168.1.1 80

**11. tcpdump - Capture Network Traffic**

bash

CopyEdit

# Capture packets on the default interface

tcpdump

# Capture on a specific network interface

tcpdump -i eth0

# Do not resolve hostnames or port names

tcpdump -n

# Show more verbose packet information

tcpdump -v

# Capture a specific number of packets

tcpdump -c 10

# Save captured packets to a file

tcpdump -w capture.pcap

# Capture HTTP traffic

tcpdump 'port 80'

# Display packet contents in ASCII format

tcpdump -A

# Display packet contents in both hex and ASCII

tcpdump -X

**12. iftop - Display Real-Time Bandwidth Usage**

bash

CopyEdit

# Display real-time bandwidth usage

iftop

# Specify the network interface to monitor

iftop -i eth0

# Show numerical addresses (do not resolve hostnames)

iftop -n

# Show port numbers along with IP addresses

iftop -P

**13. nmap - Network Exploration and Security Auditing**

bash

CopyEdit

# Perform a simple scan of a host

nmap 192.168.1.1

# Perform a ping scan to identify live hosts

nmap -sP 192.168.1.0/24

# Scan specific ports (e.g., HTTP and HTTPS ports)

nmap -p 80,443 192.168.1.1

# Perform a SYN scan (stealth scan)

nmap -sS 192.168.1.1

# OS detection

nmap -O 192.168.1.1

# Enable OS detection, version detection, script scanning, and traceroute

nmap -A 192.168.1.1

**14. mtr - Network Diagnostic Tool Combining Traceroute and Ping**

bash

CopyEdit

# Continuous traceroute and ping to a destination

mtr google.com

# Generate a report instead of continuous output

mtr -r google.com

# Set the number of pings to send

mtr -c 10 google.com

**15. arp - View and Manipulate ARP Cache**

bash

CopyEdit

# Display the current ARP table

arp -a

# Delete an entry from the ARP table

arp -d 192.168.1.1

# Manually add a static entry to the ARP table

arp -s 192.168.1.50 00:14:22:01:23:45

**16. ethtool - Query and Control Network Device Settings**

bash

CopyEdit

# Display settings for the eth0 network interface

ethtool eth0

# Set the speed and duplex mode for the interface

ethtool -s eth0 speed 1000 duplex full

# Blink the LED to locate the physical interface

ethtool -p eth0

# Display driver information for the network interface

ethtool -i eth0

# Show statistics about the network interface

ethtool -S eth0

**17. iwconfig - Configure Wireless Network Interfaces**

bash

CopyEdit

# Display wireless network information

iwconfig

# Connect to a specific Wi-Fi network

iwconfig wlan0 essid "MyWiFiNetwork"

# Set the mode of the wireless interface

iwconfig wlan0 mode managed

**18. ip link - Display/Modify Network Device Attributes**

bash

CopyEdit

# Display detailed information about network interfaces

ip link show

# Bring an interface up or down

ip link set eth0 up

ip link set eth0 down

# Change the MTU size for an interface

ip link set eth0 mtu 1500

**19. sshd - Manage SSH Daemon Configurations**

bash

CopyEdit

# Show current configuration options

sshd -T

# Use a specific SSH configuration file

sshd -f /etc/ssh/sshd\_config

# Run SSH daemon in the foreground (useful for debugging)

sshd -D

**Cloud/Container-Specific Networking Commands**

**20. docker network - Manage Docker Network Settings**

bash

CopyEdit

# List all Docker networks

docker network ls

# Inspect a specific Docker network

docker network inspect bridge

# Create a new Docker network

docker network create my\_network

# Connect a container to a network

docker network connect my\_network my\_container

**21. kubectl get svc - Retrieve Kubernetes Service Details**

bash

CopyEdit

# List all services in the current namespace

kubectl get svc

# Get details of a specific service

kubectl get svc my-service

# Describe a specific service

kubectl describe svc my-service

These commands, when applied in a real environment, help a DevOps engineer manage and monitor network connectivity, performance, and troubleshooting in Linux-based systems, both on-premise and in cloud/container environments. By practicing these commands in real-time, you can effectively troubleshoot and optimize network settings across systems.

4o

**You said:**

please compleete all these cmds of networking

**ChatGPT said:**

Sure! Below is a detailed continuation of the advanced Linux networking commands for DevOps engineers. I'll create examples for each of the commands, and provide explanations for how they work and their use cases.

**22. ss - Socket Stat**

The ss command is used to analyze socket connections and is a faster and more modern alternative to netstat.

bash

CopyEdit

# Display all sockets

ss

# Display listening TCP and UDP sockets

ss -tuln

# Show all sockets (listening and non-listening)

ss -a

# Display processes using the sockets

ss -p

# Display routing information along with socket statistics

ss -r

# Show application layer statistics

ss -A

# Show only established connections

ss -o state established

**23. nft - Netfilter (Firewall)**

nft is the tool to configure packet filtering and NAT (Network Address Translation) in the Linux kernel. It is used for advanced firewall configurations.

bash

CopyEdit

# Display all the rules in the firewall

nft list ruleset

# Add a rule to allow incoming SSH connections on port 22

nft add rule inet filter input tcp dport 22 accept

# Flush all rules from the ruleset

nft flush ruleset

# Delete a specific rule by its handle number (replace 4 with actual handle)

nft delete rule inet filter input handle 4

# List the rules in the filter table

nft list table inet filter

# Add a chain to filter incoming packets

nft add chain inet filter input { type filter hook input priority 0 ; }

**24. iptables - Configure Packet Filtering**

iptables is a classic utility to set up, maintain, and inspect the IP packet filter rules in the Linux kernel.

bash

CopyEdit

# List all active firewall rules

iptables -L

# Allow incoming HTTP traffic on port 80

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Allow incoming HTTPS traffic on port 443

iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow SSH traffic from a specific IP address

iptables -A INPUT -p tcp --dport 22 -s 192.168.1.100 -j ACCEPT

# Remove the rule allowing HTTP traffic

iptables -D INPUT -p tcp --dport 80 -j ACCEPT

# Flush all rules in the filter table

iptables -F

# Redirect HTTP traffic to a different destination IP and port

iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.2:8080

**25. firewalld - Manage Firewall Rules Dynamically**

firewalld is a firewall management tool available on many Linux distributions, providing a dynamic way to manage firewall rules.

bash

CopyEdit

# Show the current firewall settings and rules

firewall-cmd --list-all

# Open port 8080 permanently in the public zone

firewall-cmd --zone=public --add-port=8080/tcp --permanent

# Reload the firewall to apply new rules

firewall-cmd --reload

# Close port 8080 permanently

firewall-cmd --zone=public --remove-port=8080/tcp --permanent

# Show open ports in the firewall

firewall-cmd --list-ports

**26. bpftrace - Dynamic Tracing Tool**

bpftrace allows users to trace kernel-level events using BPF (Berkeley Packet Filter). It is used for real-time debugging and performance analysis.

bash

CopyEdit

# Trace all open system calls

bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { printf("%s opened\n", str(args->filename)); }'

# Trace file reads

bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }'

# Count network transmissions for each device

bpftrace -e 'tracepoint:net:net\_dev\_xmit { @[args->dev\_name] = count(); }'

**27. tcpdump with Advanced Filtering**

tcpdump is used to capture network traffic and display the captured packets. It can apply advanced filtering for deeper analysis.

bash

CopyEdit

# Capture packets on port 443 (HTTPS)

tcpdump -i eth0 port 443

# Capture TCP packets with the PUSH flag set

tcpdump -i eth0 'tcp[tcpflags] & tcp-push != 0'

# Capture packets from a specific source IP

tcpdump -i eth0 'src host 192.168.1.100'

# Capture packets destined to a specific IP

tcpdump -i eth0 'dst host 192.168.1.100'

# Write captured packets to a file

tcpdump -i eth0 -w capture.pcap

# Read packets from a capture file

tcpdump -r capture.pcap

# Increase verbosity of the output

tcpdump -i eth0 -v

# Display packet contents in ASCII format

tcpdump -A

# Display packet contents in both hex and ASCII

tcpdump -X

**28. ip rule - Policy-Based Routing**

ip rule is used to configure and manage policy-based routing, which allows routing decisions to be based on various criteria like source IP, interface, or destination.

bash

CopyEdit

# Display the current routing rules

ip rule show

# Add a rule to route traffic from a specific IP through a custom routing table

ip rule add from 192.168.1.100 table 100

# Delete the rule for traffic from a specific IP

ip rule delete from 192.168.1.100

# Route traffic coming from a specific interface through a particular table

ip rule add iif eth0 table 100

**29. tc - Traffic Control**

tc is used to control the traffic on network interfaces, such as adding network delay or limiting bandwidth.

bash

CopyEdit

# Show the current queuing disciplines (QDisc) for interfaces

tc qdisc show

# Add network delay to interface eth0

tc qdisc add dev eth0 root netem delay 100ms

# Simulate 5% packet loss on eth0

tc qdisc change dev eth0 root netem loss 5%

# Add a traffic class with a rate limit

tc class add dev eth0 parent 1:1 classid 1:10 htb rate 1mbit

# Filter HTTP traffic and assign it to a specific traffic class

tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 80 0xffff flowid 1:10

**30. ethtool (Advanced)**

ethtool is used to query and modify network interface settings, such as changing the interface speed and duplex settings.

bash

CopyEdit

# Display settings for the eth0 network interface

ethtool eth0

# Set the speed and duplex mode for the interface

ethtool -s eth0 speed 1000 duplex full

# Blink the LED to locate the physical interface

ethtool -p eth0

# Display driver information for the network interface

ethtool -i eth0

# Show statistics about the network interface

ethtool -S eth0

**31. nmcli - NetworkManager Command-Line Interface**

nmcli is used to manage network connections on systems with NetworkManager.

bash

CopyEdit

# List all active network connections

nmcli connection show

# Show the status of all network devices

nmcli device status

# Add a new Ethernet connection

nmcli connection add type ethernet con-name eth0 ifname eth0

# Disconnect a specific network interface

nmcli device disconnect eth0

# List available Wi-Fi networks

nmcli device wifi list

**32. conntrack - Connection Tracking Subsystem**

conntrack is used to manage the connection tracking table in the Linux kernel, useful for firewall and NAT configurations.

bash

CopyEdit

# List all tracked connections

conntrack -L

# Flush all tracked connections

conntrack -F

# Delete a specific connection from the connection table

conntrack -D

# List all tracked TCP connections

conntrack -L -p tcp

# Display connection events in real-time

conntrack -E

**33. wireshark - GUI-Based Network Protocol Analyzer**

Wireshark is a GUI-based tool for analyzing network traffic, often used in conjunction with tcpdump for deep packet inspection.

bash

CopyEdit

# Launch Wireshark and capture packets from a specific interface

wireshark -i eth0

# Wireshark also supports advanced filtering using display filters

# For example, filter HTTP traffic

wireshark -i eth0 "http"

# Filter packets by source IP

wireshark -i eth0 "ip.addr==192.168.1.1"

**34. bridge - Manage Network Bridges**

bridge is used to configure network bridges, such as adding interfaces to a bridge or viewing the Forwarding Database (FDB).

bash

CopyEdit

# Show the status of all network bridges

bridge link

# Show the Forwarding Database (FDB) for a bridge

bridge fdb show

# Show or configure VLANs on a bridge

bridge vlan

# Create a new network bridge

bridge addbr br0

# Add an interface to the bridge

bridge addif br0 eth0

**35. bmon - Bandwidth Monitor and Rate Estimator**

bmon is a tool used for monitoring bandwidth usage and estimating network traffic.

bash

CopyEdit

# Display real-time bandwidth usage

bmon

# Display statistics for all interfaces

bmon -p

# Show detailed statistics in continuous mode

bmon -c

These are the advanced Linux networking commands used by DevOps engineers for monitoring, managing, and troubleshooting networking. By applying these commands in your environment, you can manage network performance, configure security, debug connectivity issues, and optimize traffic across both physical and virtualized environments.

Here’s a detailed explanation of the package management commands and how to use them for Debian-based (APT) and RedHat-based (YUM/DNF) systems, along with other tools such as Snap and Flatpak, and managing repositories.

### ****APT Commands for Debian/Ubuntu-based Systems****

#### 1. apt-get **- Command-line tool for handling packages**

bash

CopyEdit

# Update the list of available packages from all configured repositories

apt-get update

# Upgrade all installed packages to the latest versions

apt-get upgrade

# Perform a full upgrade, handling changing dependencies

apt-get dist-upgrade

# Install a package

apt-get install <package>

# Install a specific version of a package

apt-get install <package>=<version>

# Remove a package (without removing configuration files)

apt-get remove <package>

# Remove a package along with its configuration files

apt-get purge <package>

# Remove packages that were installed automatically and are no longer required

apt-get autoremove

# Clean the package cache by removing downloaded package files

apt-get clean

# Clean the package cache, removing only outdated package files

apt-get autoclean

# Check for broken dependencies

apt-get check

# Download the source code for a package

apt-get source <package>

# Reinstall a package

apt-get install --reinstall <package>

# Fix broken dependencies by installing missing packages

apt-get -f install

#### 2. apt-cache **- Search and Query Package Information**

bash

CopyEdit

# Search for a package by name

apt-cache search <package\_name>

# Show detailed information about a package

apt-cache show <package\_name>

# Show package dependencies

apt-cache depends <package\_name>

# Show packages that depend on a given package

apt-cache rdepends <package\_name>

# Show the available versions and the installed version of a package

apt-cache policy <package\_name>

#### 3. apt **- User-friendly Front-End for APT**

bash

CopyEdit

# Update the list of available packages

apt update

# Upgrade installed packages to the latest versions

apt upgrade

# Install a package

apt install <package>

# Remove a package

apt remove <package>

# Search for a package in the repository

apt search <package\_name>

# Display detailed information about a package

apt show <package>

# List installed packages

apt list --installed

# List all packages that have updates available

apt list --upgradable

# Remove a package along with its configuration files

apt purge <package>

# Remove unnecessary packages

apt autoremove

# Check if a package is installed

apt list <package>

#### 4. dpkg **- Low-level Tool to Handle** .deb **Packages**

bash

CopyEdit

# Install a .deb package

dpkg -i <package.deb>

# Remove a package

dpkg -r <package>

# Purge a package (remove configuration files too)

dpkg -P <package>

# List all installed packages

dpkg -l

# List all files installed by a package

dpkg -L <package>

# Find which package a specific file belongs to

dpkg -S <file>

# Check for broken packages

dpkg -C

# Configure unpacked packages

dpkg --configure -a

# Show detailed information about an installed package

dpkg -s <package>

# Show information about an installed package in a human-readable format

dpkg -p <package>

#### 5. apt-key **- Manage Trusted Keys for APT Repositories**

bash

CopyEdit

# List the keys used by APT

apt-key list

# Add a GPG key to APT's keyring

apt-key adv --keyserver <keyserver> --recv-keys <key\_id>

# Delete a specific key from APT's trusted keys

apt-key del <key\_id>

### ****YUM and DNF Commands for RedHat/CentOS/Fedora Systems****

#### 6. yum **- Package Manager for RPM-based Distributions**

bash

CopyEdit

# Update all installed packages to the latest version

yum update

# Install a package

yum install <package>

# Remove a package

yum remove <package>

# List all installed packages

yum list installed

# List all available packages from repositories

yum list available

# Search for a package in the repositories

yum search <package>

# Show detailed information about a package

yum info <package>

# Install a group of packages (e.g., Development Tools)

yum groupinstall "<group>"

# Remove a group of packages

yum groupremove "<group>"

# Remove unnecessary packages that were installed as dependencies

yum autoremove

# Clean the repository metadata and cache

yum clean all

# Downgrade a package to an earlier version

yum downgrade <package>

#### 7. dnf **- DNF (Next Generation of YUM)**

bash

CopyEdit

# Update all installed packages to the latest version

dnf update

# Install a package

dnf install <package>

# Remove a package

dnf remove <package>

# List installed packages

dnf list installed

# List available packages from the repository

dnf list available

# Search for a package in the repositories

dnf search <package>

# Show detailed information about a package

dnf info <package>

# Install a group of packages

dnf groupinstall "<group>"

# Remove a group of packages

dnf groupremove "<group>"

# Remove unnecessary packages

dnf autoremove

# Clean metadata and cached data

dnf clean all

# Downgrade a package

dnf downgrade <package>

### ****Snap and Flatpak Package Management****

#### 8. snap **- Manage Snap Packages**

bash

CopyEdit

# Install a Snap package

snap install <package>

# Remove a Snap package

snap remove <package>

# List all installed Snap packages

snap list

# Refresh (update) Snap packages

snap refresh

# Show detailed information about a Snap package

snap info <package>

# Search for a Snap package

snap find <package>

# Show the version of Snap installed on the system

snap version

#### 9. flatpak **- Manage Flatpak Packages**

bash

CopyEdit

# Install a Flatpak package

flatpak install <package>

# Uninstall a Flatpak package

flatpak uninstall <package>

# List installed Flatpak packages

flatpak list

# Update Flatpak packages

flatpak update

# Search for a Flatpak package

flatpak search <package>

# Show detailed information about a Flatpak package

flatpak info <package>

### ****Managing Package Repositories****

#### 10. add-apt-repository **(Ubuntu/Debian)**

bash

CopyEdit

# Add a Personal Package Archive (PPA) repository

add-apt-repository ppa:<ppa\_name>

# Remove a PPA repository

add-apt-repository --remove ppa:<ppa\_name>

# Automatically accept the repository addition

add-apt-repository -y <repository>

#### 11. apt-add-repository **(Ubuntu/Debian)**

bash

CopyEdit

# Add a new repository to your system

apt-add-repository <repository>

# Remove a repository from your system

apt-add-repository -r <repository>

### ****Other Useful Package Management Commands****

#### 12. update-alternatives **- Set Default Applications for Services**

bash

CopyEdit

# Select the default version of a service

update-alternatives --config <service>

# Show all available alternatives for a service

update-alternatives --display <service>

#### 13. dpkg-reconfigure **- Reconfigure Installed Packages**

bash

CopyEdit

# Reconfigure an installed package (e.g., to reconfigure locales or X11 server settings)

dpkg-reconfigure <package>

### ****User Management Commands****

#### 1. useradd - Create a new user

bash

CopyEdit

# Create a new user with default settings

sudo useradd john

# Create a user and create their home directory

sudo useradd -m john

# Specify a custom home directory for the user

sudo useradd -d /custom/home/john john

# Specify a custom shell for the user

sudo useradd -s /bin/bash john

# Add the user to specific groups

sudo useradd -G admin,developers john

# Set the account expiry date (user will expire on 2025-12-31)

sudo useradd -e 2025-12-31 john

#### 2. usermod - Modify an existing user account

bash

CopyEdit

# Add the user to an existing group (e.g., developers)

sudo usermod -aG developers john

# Change the primary group for a user (e.g., to admin)

sudo usermod -g admin john

# Change the user’s shell to /bin/zsh

sudo usermod -s /bin/zsh john

# Lock the user account (prevents login)

sudo usermod -L john

# Unlock the user account (enables login again)

sudo usermod -U john

# Set the account expiry date (user will expire on 2025-12-31)

sudo usermod -e 2025-12-31 john

#### 3. userdel - Delete a user account

bash

CopyEdit

# Delete the user but retain their home directory

sudo userdel john

# Delete the user and their home directory

sudo userdel -r john

#### 4. groupadd - Create a new group

bash

CopyEdit

# Create a new group called 'admins'

sudo groupadd admins

# Create a group with a specific GID (1005)

sudo groupadd -g 1005 developers

# Force the creation of a group even if it already exists

sudo groupadd -f devops

#### 5. groupdel - Delete a group

bash

CopyEdit

# Delete the group 'admins'

sudo groupdel admins

#### 6. groupmod - Modify an existing group

bash

CopyEdit

# Rename the group 'developers' to 'devteam'

sudo groupmod -n devteam developers

# Change the GID of the group 'devteam' to 2005

sudo groupmod -g 2005 devteam

#### 7. chage - Change user password expiry information

bash

CopyEdit

# Show password expiry information for the user

chage -l john

# Set the maximum number of days before a password must be changed

chage -M 90 john

# Set the minimum number of days between password changes

chage -m 5 john

# Set the number of days before a password expires that the user will be warned

chage -W 7 john

# Set the number of days of inactivity before the account is disabled

chage -I 30 john

### ****Password Management Commands****

#### 8. passwd - Change user passwords

bash

CopyEdit

# Change the password for the user 'john'

sudo passwd john

# Lock the password for the user (disables login)

sudo passwd -l john

# Unlock the password for the user

sudo passwd -u john

# Expire the password (forces the user to change the password at next login)

sudo passwd -e john

# Delete the password for the user (allows login with no password)

sudo passwd -d john

# Show the status of the user’s password (locked, expired, etc.)

sudo passwd -S john

### ****File and Directory Permissions Commands****

#### 9. chmod - Change file or directory permissions

bash

CopyEdit

# Set specific permissions for a file (755 format)

chmod 755 file.txt

# Give execute permission to the user (owner) of the file

chmod u+x file.txt

# Remove write permission from the group

chmod g-w file.txt

# Add read permission for others

chmod o+r file.txt

# Set permission for the owner to read, write, execute, and others to read and execute

chmod 755 file.txt

# Recursively set all permissions to full access for all users (use with caution)

chmod -R 777 directory/

#### 10. chown - Change file or directory ownership

bash

CopyEdit

# Change the owner and group of a file or directory

sudo chown john:admin file.txt

# Change the owner of a file or directory

sudo chown john file.txt

# Change the group ownership of a file or directory

sudo chown :admin file.txt

# Recursively change ownership for all files and directories inside a directory

sudo chown -R john:admin /home/john/

# Set the ownership of file2 to be the same as file1

sudo chown --reference=file1.txt file2.txt

#### 11. chgrp - Change group ownership of a file or directory

bash

CopyEdit

# Change the group of a file or directory

sudo chgrp admin file.txt

# Recursively change the group for all files and directories inside a directory

sudo chgrp -R admin /home/john/

#### 12. umask - Set default file permissions for newly created files

bash

CopyEdit

# Display the current umask value

umask

# Set the umask value (e.g., umask 022 sets default permissions of 755 for files)

umask 022

# Set the umask value to allow no permissions by default

umask 0777

#### 13. setfacl - Set file access control lists (ACLs)

bash

CopyEdit

# Set ACL for a user on a file (e.g., give 'john' read, write, execute permissions)

setfacl -m u:john:rwx file.txt

# Remove ACL for a user

setfacl -x u:john file.txt

# Set ACL recursively for all files in a directory

setfacl -R -m u:john:rwx /home/john/

# Remove all ACLs from a file

setfacl -b file.txt

#### 14. getfacl - Get the ACLs for a file or directory

bash

CopyEdit

# Display the ACLs for a file

getfacl file.txt

# Display ACLs recursively for all files in a directory

getfacl -R /home/john/

### ****File System Permissions****

#### 15. umask (File System Permissions)

bash

CopyEdit

# Show current default file creation permissions

umask

# Set default file permissions to 755

umask 0022

# Restrict file permissions to 700 (only the owner has access)

umask 0077

### ****Sudo and Access Control Commands****

#### 16. sudo - Execute commands with superuser (root) privileges

bash

CopyEdit

# Run a command as the root user

sudo ls

# Run a command as a specific user (e.g., run 'whoami' as user 'john')

sudo -u john whoami

# List the allowed (and forbidden) commands for the current user

sudo -l

# Edit the sudoers file to configure sudo permissions

sudo visudo

#### 17. visudo - Edit the sudoers file safely

bash

CopyEdit

# Open the sudoers file in a safe mode that checks for syntax errors before saving

sudo visudo

# Edit a different sudoers file

sudo visudo -f /etc/sudoers.d/other\_config

### ****Other User and Permission Management Commands****

#### 18. id - Display user ID (UID), group ID (GID), and groups for a user

bash

CopyEdit

# Display the current user’s UID, GID, and groups

id

# Display UID, GID, and groups for a specific user

id john

#### 19. groups - Display the groups a user belongs to

bash

CopyEdit

# Show the groups the current user is part of

groups

# Show the groups for a specific user

groups john

#### 20. whoami - Display the current logged-in user

bash

CopyEdit

# Output the username of the current user

whoami

#### 21. w - Show who is logged in and what they are doing

bash

CopyEdit

# Display who is logged in and their activities

w

# Hide the header in the output

w -h

#### 22. last - Show the last logins of users

bash

CopyEdit

# Display login history for users

last

# Show the host name from where the user logged in

last -a

#### 23. finger - Show information about users

bash

CopyEdit

# Display information about a specific user

finger john

# Display information about all users currently logged in

finger

### ****ACL (Access Control Lists)****

#### 24. setfacl - Set file access control lists (ACLs)

bash

CopyEdit

# Set ACL for a user on a file (e.g., setfacl -m u:john:rwx file)

setfacl -m u:john:rwx file.txt

# Remove ACL for a user

setfacl -x u:john file.txt

Advanced File and Directory Management Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1. find Purpose: Search for files and directories in a directory hierarchy based on conditions like name, size, type, etc. • find /path/to/search -name <filename> – Search for a file by name. • find /path/to/search -type f – Find all regular files. • find /path/to/search -type d – Find all directories. • find /path/to/search -size +100M – Find files larger than 100MB. • find /path/to/search -name "\*.log" -exec rm {} \; – Find all .log files and delete them. • find /path/to/search -mtime -7 – Find files modified in the last 7 days. • find /path/to/search -empty – Find empty files or directories. • find /path/to/search -name "\*.txt" -exec mv {} /path/to/destination/ \; – Move .txt files to another directory. • find /path/to/search -user <username> – Find files owned by a specific user. 2. locate Purpose: Quickly find the location of a file using an indexed database (faster than find). • locate <filename> – Find the full path of a file. • locate -i <filename> – Perform a case-insensitive search. • updatedb – Update the file database for locate (run as root). • locate -r <pattern> – Use regular expressions to search for files. • locate --existing <pattern> – Show only files that currently exist. 3. xargs Purpose: Build and execute command lines from input (typically from find, echo, or other commands). • find . -name "\*.log" | xargs rm – Delete all .log files found by find. • echo "file1 file2 file3" | xargs cp -t /destination/ – Copy multiple files to a destination directory. • cat file\_list.txt | xargs -n 1 curl -O – Download multiple files listed in a text file. 4. tar Purpose: Archive files and directories into a .tar file (can also compress). • tar -cf archive.tar <directory\_or\_files> – Create an uncompressed archive. • tar -czf archive.tar.gz <directory\_or\_files> – Create a .tar.gz compressed archive. • tar -xvf archive.tar – Extract the contents of a .tar archive. • tar -xzvf archive.tar.gz – Extract a .tar.gz archive. • tar -tvf archive.tar – List the contents of a .tar archive without extracting. • tar -czf backup.tar.gz --exclude='\*.log' <directory> – Create a compressed archive, excluding .log files. • tar -C /path/to/destination -xvf archive.tar – Extract a .tar archive to a specific destination. 5. gzip and gunzip Purpose: Compress and decompress files using the gzip compression algorithm. • gzip <file> – Compress a file to .gz format. • gunzip <file.gz> – Decompress a .gz file. • gzip -c <file> > file.gz – Compress and write the output to a new file without modifying the original. • gzip -d <file.gz> – Decompress a .gz file. • gzip -r <directory> – Recursively compress all files in a directory. 6. zip and unzip Purpose: Create .zip archives and extract them. • zip archive.zip file1 file2 – Create a .zip archive with multiple files. • unzip archive.zip – Extract a .zip archive. • zip -r archive.zip <directory> – Create a .zip archive of an entire directory. • unzip -l archive.zip – List the contents of a .zip archive without extracting. • unzip -d /path/to/destination archive.zip – Extract a .zip archive to a specific directory. 7. rsync Purpose: Synchronize files and directories between locations. • rsync -av <source> <destination> – Copy files/directories with verbose output and preserve attributes. • rsync -avz <source> <destination> – Compress files while transferring. • rsync -av --delete <source> <destination> – Synchronize files and delete files that no longer exist at the source. • rsync -n -av <source> <destination> – Perform a dry-run (no changes, just show what will happen). • rsync -z --exclude='\*.log' <source> <destination> – Sync while excluding specific files (e.g., .log files). • rsync -avz /path/to/source/ user@remote:/path/to/destination/ – Sync to a remote server over SSH. 8. mv Purpose: Move or rename files and directories. • mv <source> <destination> – Move a file to a new location. • mv <oldname> <newname> – Rename a file. • mv -i <source> <destination> – Prompt before overwriting files. • mv -u <source> <destination> – Move only if the source file is newer than the destination. 9. cp Purpose: Copy files or directories. • cp <source> <destination> – Copy a file. • cp -r <source> <destination> – Copy a directory recursively. • cp -u <source> <destination> – Copy only if the source file is newer than the destination. • cp -i <source> <destination> – Prompt before overwriting an existing file. • cp -a <source> <destination> – Copy files and preserve attributes like timestamps, ownership, and permissions. 10. ln Purpose: Create links to files or directories. • ln <source> <link> – Create a hard link to a file. • ln -s <source> <link> – Create a symbolic (soft) link to a file or directory. • ln -f <source> <link> – Force the creation of a link by overwriting an existing file or link. • ln -s /path/to/original /path/to/symlink – Create a symbolic link to a file or directory. 11. diff Purpose: Compare files line by line. • diff <file1> <file2> – Compare two files line by line. • diff -u <file1> <file2> – Show the differences in unified format. • diff -r <dir1> <dir2> – Compare directories recursively. • diff -i <file1> <file2> – Ignore case differences when comparing. • diff -w <file1> <file2> – Ignore all white space when comparing. 12. cmp Purpose: Compare two files byte by byte. • cmp <file1> <file2> – Compare two files and return the first byte that differs. • cmp -b <file1> <file2> – Show the differing bytes when comparing files. 13. file Purpose: Determine the type of a file. • file <filename> – Display the type of a file (e.g., text, image, executable). • file -i <filename> – Display the MIME type of a file. 14. du Purpose: Estimate file and directory space usage. • du <directory> – Show the disk usage of a directory. • du -sh <directory> – Show the total disk usage of a directory in human-readable format. • du -a <directory> – Show disk usage for all files and directories. • du -h --max-depth=1 – Display disk usage for each subdirectory, limited to a depth of 1. • du -c <directory> – Show total disk usage for a directory along with individual usage for each file/folder. 15. stat Purpose: Display detailed information about a file or directory. • stat <filename> – Display file statistics, including size, permissions, timestamps, etc. • stat --format=%s <filename> – Display only the size of a file. • stat --format=%x <filename> – Display the last access time of a file. 16. tee Purpose: Read from standard input and write to standard output and files simultaneously. • command | tee <file> – Save the output of a command to a file and also display it on the terminal. • command | tee -a <file> – Append output to a file rather than overwriting it. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Advanced Compression and Decompression \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 17. xz Purpose: Compress and decompress files using the xz compression algorithm. • xz <file> – Compress a file using xz (results in .xz file). • xz -d <file.xz> – Decompress a .xz file. • xz -k <file> – Compress a file and keep the original file. • xz -l <file.xz> – Show the compressed size and other details of a .xz file. • xz -z <file> – Compress a file using the default compression level. 18. 7z Purpose: Create and extract .7z archives. • 7z a archive.7z <file\_or\_directory> – Create a .7z archive. • 7z x archive.7z – Extract a .7z archive. • 7z t archive.7z – Test the integrity of a .7z archive. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Miscellaneous Advanced File Management \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 19. shred Purpose: Securely delete a file by overwriting it multiple times. • shred -u <file> – Shred a file and remove it securely. • shred -n 5 -z <file> – Shred a file 5 times and add a final pass of zeroes. 20. mount and umount Purpose: Mount and unmount filesystems. • mount <device> <mount\_point> – Mount a filesystem or device to a directory. • umount <mount\_point> – Unmount a filesystem from a directory. create examples from scratch for all these commands and apply all these commands - with all the flags show examplse - first create and then apply all thse commands please -so that i can clearly understand

**ChatGPT said:**

Let's walk through each of the commands listed under **Advanced File and Directory Management** and provide examples for their usage. This will help you understand how to use them for practical tasks.

**1. find - Search for files and directories**

bash

CopyEdit

# Create some test files and directories

mkdir test\_directory

touch test\_directory/file1.txt test\_directory/file2.log test\_directory/file3.txt

mkdir test\_directory/subdir

touch test\_directory/subdir/file4.txt

# Find a file by name

find test\_directory -name "file1.txt"

# Find all regular files

find test\_directory -type f

# Find all directories

find test\_directory -type d

# Find files larger than 100MB (for demonstration, this is a hypothetical condition)

find test\_directory -size +100M

# Find all .log files and delete them

find test\_directory -name "\*.log" -exec rm {} \;

# Find files modified in the last 7 days

find test\_directory -mtime -7

# Find empty files or directories

find test\_directory -empty

# Find all .txt files and move them to another directory

mkdir destination\_directory

find test\_directory -name "\*.txt" -exec mv {} destination\_directory/ \;

# Find files owned by a specific user

find test\_directory -user $(whoami)

**2. locate - Quickly find the location of a file using an indexed database**

bash

CopyEdit

# Update the database for locate (run as root)

sudo updatedb

# Find the full path of a file

locate file1.txt

# Perform a case-insensitive search

locate -i file1.txt

# Use regular expressions to search for files

locate -r ".\*\.txt$"

# Show only files that currently exist

locate --existing file1.txt

**3. xargs - Build and execute command lines from input**

bash

CopyEdit

# Delete all .log files found by find using xargs

find . -name "\*.log" | xargs rm

# Copy multiple files to a destination directory

echo "file1.txt file2.txt file3.txt" | xargs cp -t /destination\_directory/

# Download multiple files listed in a text file

cat file\_list.txt | xargs -n 1 curl -O

**4. tar - Archive files and directories into a .tar file**

bash

CopyEdit

# Create an uncompressed archive

tar -cf archive.tar test\_directory/

# Create a .tar.gz compressed archive

tar -czf archive.tar.gz test\_directory/

# Extract the contents of a .tar archive

tar -xvf archive.tar

# Extract a .tar.gz archive

tar -xzvf archive.tar.gz

# List the contents of a .tar archive without extracting

tar -tvf archive.tar

# Create a compressed archive excluding .log files

tar -czf backup.tar.gz --exclude='\*.log' test\_directory/

# Extract a .tar archive to a specific destination

tar -C /path/to/destination -xvf archive.tar

**5. gzip and gunzip - Compress and decompress files using the gzip algorithm**

bash

CopyEdit

# Compress a file to .gz format

gzip file1.txt

# Decompress a .gz file

gunzip file1.txt.gz

# Compress and write the output to a new file without modifying the original

gzip -c file1.txt > file1.txt.gz

# Decompress a .gz file (alternative method)

gzip -d file1.txt.gz

# Recursively compress all files in a directory

gzip -r test\_directory/

**6. zip and unzip - Create and extract .zip archives**

bash

CopyEdit

# Create a .zip archive with multiple files

zip archive.zip file1.txt file2.txt

# Extract a .zip archive

unzip archive.zip

# Create a .zip archive of an entire directory

zip -r archive.zip test\_directory/

# List the contents of a .zip archive without extracting

unzip -l archive.zip

# Extract a .zip archive to a specific directory

unzip -d /path/to/destination archive.zip

**7. rsync - Synchronize files and directories between locations**

bash

CopyEdit

# Copy files/directories with verbose output and preserve attributes

rsync -av test\_directory/ destination\_directory/

# Compress files while transferring

rsync -avz test\_directory/ destination\_directory/

# Synchronize files and delete files that no longer exist at the source

rsync -av --delete test\_directory/ destination\_directory/

# Perform a dry-run (no changes, just show what will happen)

rsync -n -av test\_directory/ destination\_directory/

# Sync while excluding specific files (e.g., .log files)

rsync -z --exclude='\*.log' test\_directory/ destination\_directory/

# Sync to a remote server over SSH

rsync -avz test\_directory/ user@remote:/path/to/destination/

**8. mv - Move or rename files and directories**

bash

CopyEdit

# Move a file to a new location

mv file1.txt /path/to/destination/

# Rename a file

mv oldname.txt newname.txt

# Prompt before overwriting files

mv -i file1.txt /path/to/destination/

# Move only if the source file is newer than the destination

mv -u file1.txt /path/to/destination/

**9. cp - Copy files or directories**

bash

CopyEdit

# Copy a file

cp file1.txt /path/to/destination/

# Copy a directory recursively

cp -r test\_directory/ destination\_directory/

# Copy only if the source file is newer than the destination

cp -u file1.txt /path/to/destination/

# Prompt before overwriting an existing file

cp -i file1.txt /path/to/destination/

# Copy files and preserve attributes like timestamps, ownership, and permissions

cp -a file1.txt /path/to/destination/

**10. ln - Create links to files or directories**

bash

CopyEdit

# Create a hard link to a file

ln file1.txt hardlink\_file1.txt

# Create a symbolic (soft) link to a file or directory

ln -s /path/to/original /path/to/symlink

# Force the creation of a link by overwriting an existing file or link

ln -f file1.txt symlink\_file1.txt

**11. diff - Compare files line by line**

bash

CopyEdit

# Compare two files line by line

diff file1.txt file2.txt

# Show the differences in unified format

diff -u file1.txt file2.txt

# Compare directories recursively

diff -r dir1/ dir2/

# Ignore case differences when comparing

diff -i file1.txt file2.txt

# Ignore all white space when comparing

diff -w file1.txt file2.txt

**12. cmp - Compare two files byte by byte**

bash

CopyEdit

# Compare two files and return the first byte that differs

cmp file1.txt file2.txt

# Show the differing bytes when comparing files

cmp -b file1.txt file2.txt

**13. file - Determine the type of a file**

bash

CopyEdit

# Display the type of a file

file file1.txt

# Display the MIME type of a file

file -i file1.txt

**14. du - Estimate file and directory space usage**

bash

CopyEdit

# Show the disk usage of a directory

du test\_directory/

# Show the total disk usage of a directory in human-readable format

du -sh test\_directory/

# Show disk usage for all files and directories

du -a test\_directory/

# Display disk usage for each subdirectory, limited to a depth of 1

du -h --max-depth=1 test\_directory/

# Show total disk usage for a directory along with individual usage for each file/folder

du -c test\_directory/

**15. stat - Display detailed information about a file or directory**

bash

CopyEdit

# Display file statistics, including size, permissions, timestamps, etc.

stat file1.txt

# Display only the size of a file

stat --format=%s file1.txt

# Display the last access time of a file

stat --format=%x file1.txt

**16. tee - Read from standard input and write to standard output and files simultaneously**

bash

CopyEdit

# Save the output of a command to a file and also display it on the terminal

echo "Hello, World!" | tee file.txt

# Append output to a file rather than overwriting it

echo "Another line" | tee -a file.txt

**17. xz - Compress and decompress files using the xz compression algorithm**

bash

CopyEdit

# Compress a file using xz

xz file1.txt

# Decompress a .xz file

xz -d file1.txt.xz

# Compress a file and keep the original file

xz -k file1.txt

# Show the compressed size and other details of a .xz file

xz -l file1.txt.xz

# Compress a file using the default compression level

xz -z file1.txt

**18. 7z - Create and extract .7z archives**

bash

CopyEdit

# Create a .7z archive

7z a archive.7z file1.txt file2.txt

# Extract a .7z archive

7z x archive.7z

# Test the integrity of a .7z archive

7z t archive.7z

**19. shred - Securely delete a file by overwriting it multiple times**

bash

CopyEdit

# Shred a file and remove it securely

shred -u file1.txt

# Shred a file 5 times and add a final pass of zeroes

shred -n 5 -z file1.txt

**20. mount and umount - Mount and unmount filesystems**

bash

CopyEdit

# Mount a filesystem or device to a directory

sudo mount /dev/sdb1 /mnt

# Unmount a filesystem from a directory

sudo umount /mnt

1. ps (Process Status) Purpose: Show information about active processes. • ps – Display the processes running in the current terminal session. • ps aux – Show all running processes on the system, including other users’ processes. • ps -ef – Display all processes in full format. • ps -e – Show all processes running on the system. • ps -f – Show full information (including command arguments) for each process. • ps -u <username> – Show processes owned by a specific user. • ps -p <pid> – Show information about a specific process by its PID. • ps -o pid,comm,%cpu,%mem – Show only selected columns (PID, command, CPU, memory). • ps -C <command> – Show processes for a specific command (e.g., ps -C nginx). 2. top Purpose: Display a real-time view of the system’s resource usage (CPU, memory, processes). • top – Display the system’s process list and resource usage in real-time. • top -u <username> – Show processes for a specific user. • top -p <pid> – Monitor a specific process by PID. • top -d <seconds> – Set the delay for updates (e.g., -d 2 updates every 2 seconds). • top -n <count> – Show a specific number of iterations (e.g., top -n 10 shows 10 updates). • top -c – Show full command lines for each process. • top -s <field> – Sort processes by a specific field (e.g., cpu, mem). • top -H – Show threads of processes. 3. htop Purpose: Interactive, user-friendly version of top for process monitoring. • htop – Start the interactive process viewer. • htop -u <username> – Show processes for a specific user. • htop -s <field> – Sort processes by a specified field (e.g., CPU, memory). • htop -d <delay> – Set the delay between updates (e.g., htop -d 2 updates every 2 seconds). • htop -p <pid> – Show specific process by PID. • htop -v – Show version of htop. 4. pgrep Purpose: Find processes based on name and other attributes. • pgrep <pattern> – Find processes matching a pattern (e.g., pgrep nginx). • pgrep -u <username> – Find processes owned by a specific user. • pgrep -f <pattern> – Search for processes by their full command line. • pgrep -a <pattern> – Display the process ID and the command line of matching processes. 5. pidof Purpose: Find the process ID (PID) of a running program. • pidof <program> – Find the PID of a running program (e.g., pidof nginx). • pidof -x <command> – Find PIDs of processes that match a specific command. 6. kill Purpose: Terminate processes by sending signals. • kill <pid> – Send the default TERM signal to terminate a process by its PID. • kill -9 <pid> – Send the KILL signal to forcibly terminate a process. • kill -15 <pid> – Send the TERM signal to gracefully terminate a process. • kill -l – List all available signals. • killall <name> – Terminate all processes with a specific name (e.g., killall nginx). - create examples from scratch and apply all these commands please

###### ChatGPT said:

Here’s a breakdown of each process management command with examples that demonstrate how they can be used. These examples will help you understand how to apply these commands for various purposes related to process management on Linux systems.

### 1. ****ps (Process Status)****

The ps command is used to display information about active processes.

#### Examples:

bash

CopyEdit

# Display the processes running in the current terminal session

ps

# Show all running processes on the system, including other users’ processes

ps aux

# Display all processes in full format

ps -ef

# Show all processes running on the system

ps -e

# Show full information (including command arguments) for each process

ps -f

# Show processes owned by a specific user (e.g., username = john)

ps -u john

# Show information about a specific process by its PID (e.g., PID = 1234)

ps -p 1234

# Show selected columns: PID, command, CPU usage, and memory usage

ps -o pid,comm,%cpu,%mem

# Show processes for a specific command (e.g., nginx)

ps -C nginx

### 2. ****top****

top is a real-time process viewer that provides an overview of the system's resource usage.

#### Examples:

bash

CopyEdit

# Display the system’s process list and resource usage in real-time

top

# Show processes for a specific user (e.g., username = john)

top -u john

# Monitor a specific process by PID (e.g., PID = 1234)

top -p 1234

# Set the delay for updates (e.g., update every 2 seconds)

top -d 2

# Show a specific number of iterations (e.g., top will show 10 updates)

top -n 10

# Show full command lines for each process

top -c

# Sort processes by a specific field (e.g., CPU usage)

top -s %cpu

# Show threads of processes

top -H

### 3. ****htop****

htop is an interactive and user-friendly version of top. It provides additional features like sorting and filtering.

#### Examples:

bash

CopyEdit

# Start the interactive process viewer

htop

# Show processes belonging to a specific user (e.g., username = john)

htop -u john

# Sort processes by a specified field (e.g., CPU usage)

htop -s CPU

# Set the delay between updates (e.g., 2 seconds)

htop -d 2

# Show information about a specific process by PID (e.g., PID = 1234)

htop -p 1234

# Show the version of htop

htop -v

### 4. ****pgrep****

pgrep searches for processes based on a pattern and returns their process IDs (PIDs).

#### Examples:

bash

CopyEdit

# Find processes matching a pattern (e.g., nginx)

pgrep nginx

# Find processes owned by a specific user (e.g., username = john)

pgrep -u john

# Search for processes by their full command line (e.g., any command containing 'nginx')

pgrep -f nginx

# Display the process ID and the command line of matching processes

pgrep -a nginx

### 5. ****pidof****

pidof finds the process ID (PID) of a running program.

#### Examples:

bash

CopyEdit

# Find the PID of a running program (e.g., nginx)

pidof nginx

# Find PIDs of processes that match a specific command (e.g., nginx)

pidof -x /usr/sbin/nginx

### 6. ****kill****

The kill command sends signals to terminate processes.

#### Examples:

bash

CopyEdit

# Send the default TERM signal to terminate a process by its PID (e.g., PID = 1234)

kill 1234

# Send the KILL signal to forcibly terminate a process by PID

kill -9 1234

# Send the TERM signal to gracefully terminate a process

kill -15 1234

# List all available signals

kill -l

# Terminate all processes with a specific name (e.g., nginx)

killall nginx

### Additional Notes:

* **ps aux** provides a comprehensive view of all running processes, including processes by other users.
* **top and htop** are very useful for real-time monitoring, with htop being more interactive and visually appealing.
* **pgrep** and **pidof** are helpful for quickly finding processes by name or matching patterns.
* **kill and killall** are essential tools for managing processes, where kill -9 can forcefully terminate a process if it does not respond to other signals.

These commands provide a solid foundation for managing and monitoring processes in Linux environments, crucial for system administrators and DevOps engineers.

4o

##### You said:

7. nice Purpose: Run a command with a modified scheduling priority (affects CPU scheduling). • nice -n <priority> <command> – Run a command with a specified nice value (e.g., nice -n 10 command). • nice -n -10 <command> – Run a command with a higher priority (lower nice value). • nice -n 0 <command> – Run a command with the default priority. 8. renice Purpose: Change the priority (nice value) of an already running process. • renice <priority> -p <pid> – Change the nice value of a process by its PID. • renice -n <priority> -u <username> – Change the nice value of processes owned by a specific user. • renice -n -5 -p 1234 – Set the priority of a process with PID 1234 to -5 (higher priority). 9. nohup Purpose: Run a command immune to hangups (ignores SIGHUP), typically used to run commands in the background. • nohup <command> & – Run a command in the background, ignoring hangups. • nohup <command> > output.log 2>&1 & – Redirect the output of a background process to a log file. 10. at Purpose: Schedule a command to run once at a specified time. • at <time> – Open the at prompt to schedule a command. o E.g., at 2:00 PM or at now + 1 hour. • atq – Display the list of pending at jobs. • atrm <job\_id> – Remove a specific job from the at queue. 11. cron and crontab Purpose: Automate tasks by scheduling them to run at regular intervals. • crontab -e – Edit the cron jobs for the current user. • crontab -l – List all cron jobs for the current user. • crontab -r – Remove all cron jobs for the current user. • cron – Start the cron daemon. • crontab -u <username> -e – Edit the cron jobs for a specific user. • Example cron job syntax: o \* \* \* \* \* <command> – Run <command> every minute. o 0 5 \* \* \* <command> – Run <command> at 5 AM daily. 12. systemctl Purpose: Control the systemd system and service manager (starting, stopping, managing system services). • systemctl status <service> – Show the status of a specific service (e.g., systemctl status nginx). • systemctl start <service> – Start a service (e.g., systemctl start nginx). • systemctl stop <service> – Stop a service. • systemctl restart <service> – Restart a service. • systemctl enable <service> – Enable a service to start on boot. • systemctl disable <service> – Disable a service from starting on boot. • systemctl list-units – List all active units (services, sockets, etc.). • systemctl kill <pid> – Kill a process using its PID. • systemctl show <service> – Show detailed information about a service. • systemctl mask <service> – Mask a service, preventing it from being started. 13. journalctl Purpose: Query and display logs from systemd journal (logs for processes, services). • journalctl – Show logs for all services and processes. • journalctl -u <service> – Show logs for a specific service (e.g., journalctl -u nginx). • journalctl -f – Show real-time logs as they are written (similar to tail -f). • journalctl -p <priority> – Filter logs by priority (e.g., journalctl -p err shows only error logs). • journalctl --since "2023-01-01" – Show logs since a specific date. 14. strace Purpose: Trace system calls and signals for a process. • strace -p <pid> – Attach to a running process and trace its system calls. • strace -f -p <pid> – Trace child processes as well. • strace <command> – Run a command and trace its system calls (e.g., strace ls). • strace -e trace=<syscall> – Trace a specific system call (e.g., strace -e trace=open). 15. lsof Purpose: List open files and processes using them. • lsof – List all open files on the system. • lsof -i :<port> – List processes using a specific network port. • lsof -u <username> – List open files by a specific user. • lsof -p <pid> – List open files for a specific process by its PID. -

###### ChatGPT said:

Let's continue with examples for these advanced process management commands, showing how each one can be applied in different scenarios.

### 7. ****nice****

The nice command is used to run a command with a modified scheduling priority (affects CPU scheduling).

#### Examples:

bash

CopyEdit

# Run a command with a specified nice value (e.g., run the 'sleep 10' command with nice value 10)

nice -n 10 sleep 10

# Run a command with a higher priority (lower nice value, e.g., nice value -10)

nice -n -10 sleep 10

# Run a command with the default priority (0 is the default nice value)

nice -n 0 sleep 10

### 8. ****renice****

The renice command allows you to change the priority (nice value) of an already running process.

#### Examples:

bash

CopyEdit

# Change the nice value of a process with PID 1234 to 10

renice 10 -p 1234

# Change the nice value of processes owned by the user 'john' to -5

renice -n -5 -u john

# Set the priority of a process with PID 1234 to -5 (higher priority)

renice -n -5 -p 1234

### 9. ****nohup****

The nohup command is used to run a command immune to hangups, meaning it continues running after the user logs out.

#### Examples:

bash

CopyEdit

# Run a command in the background, ignoring hangups (e.g., run a long-running process)

nohup long-running-process &

# Run a command and redirect its output to a log file, ignoring hangups

nohup long-running-process > output.log 2>&1 &

### 10. ****at****

The at command is used to schedule a command to run once at a specified time.

#### Examples:

bash

CopyEdit

# Schedule a command to run at 2:00 PM

echo "your-command-here" | at 2:00 PM

# Schedule a command to run 1 hour from now

echo "your-command-here" | at now + 1 hour

# Display the list of pending jobs in the at queue

atq

# Remove a specific job from the at queue (e.g., job ID = 1)

atrm 1

### 11. ****cron and crontab****

The cron and crontab commands are used to schedule recurring tasks.

#### Examples:

bash

CopyEdit

# Edit the cron jobs for the current user

crontab -e

# List all cron jobs for the current user

crontab -l

# Remove all cron jobs for the current user

crontab -r

# Start the cron daemon (typically, cron service is already running)

cron

# Edit cron jobs for a specific user (e.g., username = john)

crontab -u john -e

# Example cron job syntax:

# Run 'your-command' every minute

\* \* \* \* \* your-command

# Run 'your-command' at 5 AM daily

0 5 \* \* \* your-command

### 12. ****systemctl****

The systemctl command is used to control systemd system and service manager, allowing you to start, stop, restart, or enable/disable services.

#### Examples:

bash

CopyEdit

# Show the status of a specific service (e.g., nginx)

systemctl status nginx

# Start the nginx service

systemctl start nginx

# Stop the nginx service

systemctl stop nginx

# Restart the nginx service

systemctl restart nginx

# Enable the nginx service to start on boot

systemctl enable nginx

# Disable the nginx service from starting on boot

systemctl disable nginx

# List all active units (services, sockets, etc.)

systemctl list-units

# Kill a process using its PID (e.g., PID = 1234)

systemctl kill 1234

# Show detailed information about a service (e.g., nginx)

systemctl show nginx

# Mask a service to prevent it from being started

systemctl mask nginx

### 13. ****journalctl****

The journalctl command is used to query and display logs from the systemd journal, which logs all activities for services and processes.

#### Examples:

bash

CopyEdit

# Show logs for all services and processes

journalctl

# Show logs for a specific service (e.g., nginx)

journalctl -u nginx

# Show real-time logs as they are written (similar to tail -f)

journalctl -f

# Filter logs by priority (e.g., show error logs only)

journalctl -p err

# Show logs since a specific date (e.g., 2023-01-01)

journalctl --since "2023-01-01"

### 14. ****strace****

The strace command is used to trace system calls and signals for a process.

#### Examples:

bash

CopyEdit

# Attach to a running process with PID 1234 and trace its system calls

strace -p 1234

# Trace child processes as well (e.g., trace a process and its forks)

strace -f -p 1234

# Run a command and trace its system calls (e.g., strace ls)

strace ls

# Trace a specific system call (e.g., trace the 'open' system call)

strace -e trace=open ls

# Write the output of strace to a file (e.g., trace a command and output to output.log)

strace -o output.log ls

# Report statistics on system calls used by a command

strace -c ls

# Include timestamps for system calls

strace -t ls

### 15. ****lsof****

The lsof command is used to list open files and processes using them.

#### Examples:

bash

CopyEdit

# List all open files by all processes

lsof

# List all network connections (both listening and established)

lsof -i

# List processes using a specific network port (e.g., port 80)

lsof -i :80

# List open files for a specific process (e.g., PID = 1234)

lsof -p 1234

# List open files in a specific directory and its subdirectories

lsof +D /path/to/directory

# Show all files opened by a specific user (e.g., username = john)

lsof -u john

# Get the PID(s) of processes using a specific port (e.g., port 80)

lsof -t -i :80

### Additional Notes:

* **nice and renice** are critical for controlling the priority of processes. nice is for starting processes with a custom priority, while renice modifies the priority of an already running process.
* **nohup** is invaluable for long-running commands, especially when you need to run processes in the background or on remote systems.
* **at** and **cron** provide powerful scheduling capabilities, where at is for one-time jobs, and cron is for recurring jobs.
* **systemctl** is essential for managing services in a systemd-based system, providing control over services and their states.
* **journalctl** and **strace** are helpful for troubleshooting and debugging services, with journalctl offering logs and strace tracing system calls.
* **lsof** is an important tool for investigating open files and network connections, useful for debugging and identifying resource usage.

By mastering these commands, you can efficiently manage, monitor, and troubleshoot processes and services in your Linux environment.

4o

##### You said:

16. uptime Purpose: Show how long the system has been running, number of users, and load averages. • uptime – Display system uptime, number of users, and load averages for the last 1, 5, and 15 minutes. 17. nice Purpose: Set the priority (nice value) of a process. • nice -n <priority> <command> – Start a command with a specific priority (e.g., nice -n 10 command). • nice -n -10 <command> – Start a command with higher priority (lower nice value). 18. watch Purpose: Run a command periodically and show the output in real-time. • watch <command> – Run a command every 2 seconds by default. • watch -n <seconds> <command> – Set a custom interval (e.g., watch -n 5 df -h runs df every 5 seconds).

###### ChatGPT said:

### 16. ****uptime****

The uptime command provides information about how long the system has been running, the number of users currently logged in, and the system load averages over the last 1, 5, and 15 minutes.

#### Example:

bash

CopyEdit

# Display system uptime, number of users, and load averages

uptime

Output example:

bash

CopyEdit

09:12:34 up 5 days, 3:21, 3 users, load average: 0.10, 0.15, 0.12

Explanation:

* **up 5 days, 3:21** indicates the system has been running for 5 days and 3 hours and 21 minutes.
* **3 users** shows that there are 3 users logged in.
* **load average: 0.10, 0.15, 0.12** shows the system load for the last 1, 5, and 15 minutes, respectively. Lower values indicate less system load.

### 17. ****nice****

The nice command is used to start a command with a modified priority. It adjusts the CPU scheduling priority of a command, with lower values indicating higher priority (and vice versa).

#### Examples:

bash

CopyEdit

# Run a command with a specified nice value (e.g., run 'sleep 10' with nice value 10)

nice -n 10 sleep 10

# Run a command with higher priority (lower nice value, e.g., nice value -10)

nice -n -10 sleep 10

# Run a command with the default priority (nice value 0)

nice -n 0 sleep 10

Explanation:

* **nice -n 10 sleep 10** runs the sleep 10 command with a nice value of 10 (lower priority).
* **nice -n -10 sleep 10** runs the sleep 10 command with a nice value of -10 (higher priority).
* **nice -n 0 sleep 10** runs the sleep 10 command with the default priority (nice value 0).

### 18. ****watch****

The watch command runs a specified command periodically, showing the output in real-time. This is useful for monitoring a command's output dynamically without re-running it manually.

#### Examples:

bash

CopyEdit

# Run the 'df -h' command every 2 seconds (default interval)

watch df -h

Output example (displayed every 2 seconds):

bash

CopyEdit

Filesystem Size Used Avail Use% Mounted on

/dev/sda1 50G 10G 40G 20% /

/dev/sdb1 50G 15G 35G 30% /data

bash

CopyEdit

# Run the 'uptime' command every 5 seconds

watch -n 5 uptime

Output example:

bash

CopyEdit

Every 5.0s: uptime Mon Apr 6 09:18:31 2025

09:18:31 up 5 days, 3:30, 3 users, load average: 0.02, 0.10, 0.08

Explanation:

* **watch df -h** will display the disk usage (df -h) every 2 seconds.
* **watch -n 5 uptime** will show the system's uptime every 5 seconds.
* **-n <seconds>** is used to set a custom interval between command executions.

**watch** can be very useful for observing the output of long-running commands or monitoring system resources in real-time.

These three commands (uptime, nice, and watch) are especially useful for system performance monitoring and process management in a DevOps environment. The uptime command helps track how long the system has been running and the load, while nice and renice allow for fine-tuning the CPU scheduling priority of processes. Lastly, watch is a simple yet effective tool for periodically running commands and observing their outputs in real-time.

4o

##### You said:

19. systemd-cgls Purpose: Display the control groups of all processes in the system. • systemd-cgls – Display the control group hierarchy for processes managed by systemd. • systemd-cgls <service> – Show the control group of a specific service. • systemd-cgls --user – Show control groups for user-level processes. 20. systemd-analyze Purpose: Analyze and measure systemd startup performance and boot times. • systemd-analyze – Show the total time spent on boot. • systemd-analyze blame – Show which services took the longest to start. • systemd-analyze critical-chain – Show the critical chain of services that directly impact the boot time. • systemd-analyze plot > boot-time.svg – Create a boot time plot (SVG format) for visual analysis. 21. cgroups (Control Groups) Purpose: Manage resource allocation for processes and services. • cgcreate -g memory,cpu:/mygroup – Create a new control group (e.g., mygroup for memory and CPU resource management). • cgset -r memory.limit\_in\_bytes=100M /mygroup – Set memory limit for the control group. • cgexec -g memory,cpu:/mygroup <command> – Execute a command under a specific control group. • cat /sys/fs/cgroup/memory/mygroup/memory.usage\_in\_bytes – Show the memory usage of a specific control group. • cgdelete memory,cpu:/mygroup – Delete a control group. 22. htop (Advanced) Purpose: A more advanced and user-friendly version of top for process management and monitoring. • htop – Start an interactive process viewer with more features and easier navigation. • htop -u <username> – Show processes belonging to a specific user. • htop -p <pid> – Display information about a specific process. • htop -d <delay> – Set a custom delay between updates in milliseconds (e.g., htop -d 500 for a 500ms delay). • htop -s <column> – Sort processes by a specific column (e.g., CPU, memory). • htop -M – Show memory usage in the format of MB, GB, etc. 23. perf Purpose: Performance monitoring and profiling tool for Linux. • perf top – Display a live profile of the processes consuming the most CPU. • perf stat <command> – Collect performance statistics (e.g., CPU cycles, instructions) for a specific command. • perf record -p <pid> – Record performance data for a specific process. • perf report – Show a report of the recorded performance data. • perf trace – Trace system calls and events for a process. • perf record -g <command> – Record a call graph (useful for profiling). 24. lsof (Advanced Usage) Purpose: List open files, processes using them, and network connections. • lsof – Show all open files by all processes. • lsof -i – Show all network connections (both listening and established). • lsof -i :<port> – List processes using a specific port. • lsof -p <pid> – List files opened by a specific process (given by PID). • lsof +D <directory> – List open files in a specific directory and its subdirectories. • lsof -u <username> – Show all files opened by a specific user. • lsof -t -i :<port> – Get the PID(s) of processes using a specific port (useful for killing processes). 25. killall (Advanced) Purpose: Kill processes by name rather than PID. • killall <process\_name> – Kill all processes with a specific name (e.g., killall nginx). • killall -u <username> – Kill all processes owned by a specific user. • killall -9 <process\_name> – Forcefully terminate all processes with a specific name. • killall -r <pattern> – Kill processes by matching a regular expression (e.g., killall -r 'nginx.\*'). 26. strace (Advanced) Purpose: Trace system calls and signals used by a process. • strace -p <pid> – Attach to a running process and trace its system calls. • strace -f <pid> – Trace child processes as well (e.g., strace -f -p <pid>). • strace -e trace=<syscall> – Trace only specific system calls (e.g., strace -e trace=open). • strace -o <file> – Write the output of strace to a file (e.g., strace -o output.log <command>). • strace -c <command> – Report statistics on system calls used by a command. • strace -t <command> – Include timestamps for system calls. 27. time Purpose: Measure the time a process takes to execute. • time <command> – Display the real time, user time, and system time for executing a command. • time -v <command> – Show detailed timing information, including memory usage, page faults, and I/O operations. • time <command> > output.log 2>&1 – Redirect both standard output and error to a file. 28. ftrace Purpose: Kernel tracing for detailed information on function calls and events. • echo function > /sys/kernel/debug/tracing/current\_tracer – Set the tracer to function. • cat /sys/kernel/debug/tracing/trace – View the kernel trace buffer. • echo 1 > /sys/kernel/debug/tracing/tracing\_on – Start tracing. • echo 0 > /sys/kernel/debug/tracing/tracing\_on – Stop tracing. • trace-cmd – Collect and analyze kernel trace events. 29. dstat Purpose: Show various system statistics in real time. • dstat – Display general system performance stats like CPU, memory, disk, and network usage. • dstat -c – Show only CPU stats. • dstat -d – Show only disk stats. • dstat -n – Show only network stats. • dstat -t – Display timestamp with each statistic. • dstat -g – Show page faults and garbage collection stats. 30. watch Purpose: Run a command periodically and show its output in real-time. • watch <command> – Run a command every 2 seconds by default (e.g., watch df -h). • watch -n <seconds> <command> – Set the interval in seconds between command executions (e.g., watch -n 5 uptime). • watch -d <command> – Highlight the changes in the output. • watch -t <command> – Run the command without showing the header. 31. sysctl Purpose: Control kernel parameters at runtime, including those related to processes. • sysctl -a – Show all kernel parameters. • sysctl vm.overcommit\_memory=1 – Set the kernel overcommit memory policy (tune memory management). • sysctl -w fs.inotify.max\_user\_watches=524288 – Increase the maximum number of user watches for inotify. • sysctl -p – Apply changes from the /etc/sysctl.conf file. • sysctl -w kernel.pid\_max=65535 – Set the maximum number of process IDs. 32. ulimit Purpose: Control user process resource limits. • ulimit -a – Show all current limits (e.g., maximum open files, processes). • ulimit -u <number> – Set the maximum number of processes available to a user. • ulimit -n <number> – Set the maximum number of open file descriptors. • ulimit -s <size> – Set the stack size for processes. • ulimit -c <size> – Set the core dump file size limit. 33. psacct / acct Purpose: Account for process usage and record user activity. • ac – Display the total time spent by users in the system. • lastcomm – Show information about the commands previously executed by users. • sa – Report on CPU time used by all processes.

###### ChatGPT said:

Here’s a breakdown of the advanced process management commands with examples for each command:

### 19. ****systemd-cgls****

Displays the control group hierarchy for processes managed by systemd. This is useful to see how processes are grouped under systemd's control groups (cgroups).

#### Examples:

bash

CopyEdit

# Display the control group hierarchy for all processes

systemd-cgls

# Display the control group for a specific service (e.g., nginx)

systemd-cgls nginx

# Show control groups for user-level processes

systemd-cgls --user

### 20. ****systemd-analyze****

Analyze and measure systemd's startup performance and boot times, helping identify any services that are delaying boot.

#### Examples:

bash

CopyEdit

# Show the total time spent on boot

systemd-analyze

# Show which services took the longest to start during boot

systemd-analyze blame

# Show the critical chain of services that directly impact boot time

systemd-analyze critical-chain

# Create a visual boot-time plot

systemd-analyze plot > boot-time.svg

### 21. ****cgroups (Control Groups)****

Control groups (cgroups) allow you to manage resource allocation for processes and services, such as CPU, memory, and I/O.

#### Examples:

bash

CopyEdit

# Create a new control group for memory and CPU

cgcreate -g memory,cpu:/mygroup

# Set a memory limit of 100MB for the control group

cgset -r memory.limit\_in\_bytes=100M /mygroup

# Execute a command under a specific control group

cgexec -g memory,cpu:/mygroup stress --cpu 4

# Show the memory usage of a specific control group

cat /sys/fs/cgroup/memory/mygroup/memory.usage\_in\_bytes

# Delete the control group

cgdelete memory,cpu:/mygroup

### 22. ****htop (Advanced)****

htop is an interactive, user-friendly process viewer that provides an easy way to monitor system resources and manage processes.

#### Examples:

bash

CopyEdit

# Start the interactive process viewer

htop

# Show processes belonging to a specific user

htop -u <username>

# Display information about a specific process by PID

htop -p <pid>

# Set a custom delay between updates (e.g., 500ms)

htop -d 500

# Sort processes by a specified column (e.g., CPU usage)

htop -s PERCENT\_CPU

# Show memory usage in MB, GB, etc.

htop -M

### 23. ****perf****

perf is a powerful tool for performance monitoring and profiling, allowing you to analyze CPU usage, system calls, and more.

#### Examples:

bash

CopyEdit

# Display a live profile of processes consuming the most CPU

perf top

# Collect performance statistics for a specific command

perf stat ls

# Record performance data for a specific process by PID

perf record -p <pid>

# Show a report of the recorded performance data

perf report

# Trace system calls and events for a process

perf trace

# Record a call graph for profiling a command

perf record -g ls

### 24. ****lsof (Advanced Usage)****

lsof is used to list open files, network connections, and processes using those resources.

#### Examples:

bash

CopyEdit

# Show all open files by all processes

lsof

# Show all network connections (both listening and established)

lsof -i

# List processes using a specific port (e.g., port 80)

lsof -i :80

# List files opened by a specific process (given by PID)

lsof -p <pid>

# List open files in a specific directory and its subdirectories

lsof +D /path/to/directory

# Show all files opened by a specific user

lsof -u <username>

# Get the PID(s) of processes using a specific port (useful for killing processes)

lsof -t -i :80

### 25. ****killall (Advanced)****

killall is used to terminate processes by name rather than PID, allowing you to manage processes more easily.

#### Examples:

bash

CopyEdit

# Kill all processes with a specific name (e.g., nginx)

killall nginx

# Kill all processes owned by a specific user

killall -u <username>

# Forcefully terminate all processes with a specific name

killall -9 nginx

# Kill processes by matching a regular expression

killall -r 'nginx.\*'

### 26. ****strace (Advanced)****

strace is a powerful tool for tracing system calls and signals used by a process. This is helpful for debugging and understanding the behavior of processes.

#### Examples:

bash

CopyEdit

# Attach to a running process and trace its system calls

strace -p <pid>

# Trace child processes as well

strace -f -p <pid>

# Run a command and trace its system calls

strace ls

# Trace only specific system calls (e.g., open system call)

strace -e trace=open ls

# Write the output of strace to a file

strace -o output.log ls

# Report statistics on system calls used by a command

strace -c ls

# Include timestamps for system calls

strace -t ls

### 27. ****time****

The time command is used to measure the time a process takes to execute, which is useful for performance analysis.

#### Examples:

bash

CopyEdit

# Display real time, user time, and system time for executing a command

time ls

# Show detailed timing information, including memory usage and page faults

time -v ls

# Redirect both standard output and error to a file

time ls > output.log 2>&1

### 28. ****ftrace****

ftrace is used for kernel tracing, providing detailed information on function calls and events, which is useful for kernel-level debugging.

#### Examples:

bash

CopyEdit

# Set the tracer to function tracing

echo function > /sys/kernel/debug/tracing/current\_tracer

# View the kernel trace buffer

cat /sys/kernel/debug/tracing/trace

# Start tracing

echo 1 > /sys/kernel/debug/tracing/tracing\_on

# Stop tracing

echo 0 > /sys/kernel/debug/tracing/tracing\_on

# Collect and analyze kernel trace events using trace-cmd

trace-cmd record

### 29. ****dstat****

dstat is a versatile tool for displaying various system statistics in real-time, including CPU, memory, disk, and network usage.

#### Examples:

bash

CopyEdit

# Display general system performance stats like CPU, memory, disk, and network usage

dstat

# Show only CPU stats

dstat -c

# Show only disk stats

dstat -d

# Show only network stats

dstat -n

# Display timestamp with each statistic

dstat -t

# Show page faults and garbage collection stats

dstat -g

### 30. ****watch****

watch runs a specified command periodically and displays its output in real-time. This is useful for monitoring a command's output without having to rerun it manually.

#### Examples:

bash

CopyEdit

# Run the 'df -h' command every 2 seconds (default interval)

watch df -h

# Run the 'uptime' command every 5 seconds

watch -n 5 uptime

# Highlight the changes in the output

watch -d df -h

# Run the command without showing the header

watch -t df -h

### 31. ****sysctl****

sysctl is used to control kernel parameters at runtime, which can affect processes and system behavior.

#### Examples:

bash

CopyEdit

# Show all kernel parameters

sysctl -a

# Set the kernel overcommit memory policy

sysctl vm.overcommit\_memory=1

# Increase the maximum number of user watches for inotify

sysctl -w fs.inotify.max\_user\_watches=524288

# Apply changes from the /etc/sysctl.conf file

sysctl -p

# Set the maximum number of process IDs

sysctl -w kernel.pid\_max=65535

### 32. ****ulimit****

The ulimit command is used to control user process resource limits, such as the number of processes, file descriptors, and stack size.

#### Examples:

bash

CopyEdit

# Show all current limits

ulimit -a

# Set the maximum number of processes available to a user

ulimit -u 1000

# Set the maximum number of open file descriptors

ulimit -n 1024

# Set the stack size for processes

ulimit -s 8192

# Set the core dump file size limit

ulimit -c unlimited

### 33. ****psacct / acct****

These tools account for process usage and record user activity, which can be useful for auditing and monitoring system usage.

#### Examples:

bash

CopyEdit

# Display the total time spent by users in the system

ac

# Show information about the commands previously executed by users

lastcomm

# Report on CPU time used by all processes

sa

Basic Text Processing Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1. cat (Concatenate) Purpose: Display the contents of a file or concatenate multiple files. • cat <file> – Display the content of a file. • cat file1 file2 – Concatenate and display the content of multiple files. • cat <file> > newfile – Redirect the content of a file to a new file. • cat <file> >> newfile – Append the content of a file to another file. 2. echo Purpose: Print text or variables to the terminal or redirect to a file. • echo "Hello World" – Display a simple string. • echo $HOME – Display the value of an environment variable. • echo "Hello World" > output.txt – Write the output to a file. • echo -n "Hello" – Print without a trailing newline. 3. head Purpose: Display the first few lines of a file. • head <file> – Display the first 10 lines of a file. • head -n <number> <file> – Display the first <number> lines (e.g., head -n 20 file.txt). • head -c <number> <file> – Display the first <number> bytes of a file. 4. tail Purpose: Display the last few lines of a file. • tail <file> – Display the last 10 lines of a file. • tail -n <number> <file> – Display the last <number> lines. • tail -f <file> – Display the last lines of a file and follow it (useful for log files). • tail -c <number> <file> – Display the last <number> bytes of a file. 5. cut Purpose: Remove sections from each line of a file (often used to extract columns). • cut -d':' -f1 <file> – Display the first field of each line using : as the delimiter. • cut -d',' -f1,3 <file> – Display the first and third fields (columns). • cut -c 1-10 <file> – Display the first 10 characters of each line. 6. tr Purpose: Translate or delete characters from the input. • tr 'a-z' 'A-Z' – Convert lowercase characters to uppercase. • echo "hello" | tr 'a-z' 'A-Z' – Convert lowercase to uppercase using a pipeline. • tr -d '[:space:]' <file> – Delete all spaces from the input file. • echo "123 456 789" | tr ' ' '\n' – Replace spaces with newlines. 7. sort Purpose: Sort lines in a file or input. • sort <file> – Sort lines of a file alphabetically. • sort -n <file> – Sort lines numerically. • sort -r <file> – Sort lines in reverse order. • sort -u <file> – Sort and remove duplicate lines. • sort -k <column\_number> <file> – Sort by a specific column. • sort -t, -k1,1 <file> – Sort a CSV file by the first column. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Intermediate Text Processing Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 8. grep Purpose: Search for patterns in text files using regular expressions. • grep <pattern> <file> – Search for a pattern in a file. • grep -i <pattern> <file> – Perform a case-insensitive search. • grep -r <pattern> <directory> – Search recursively through a directory. • grep -v <pattern> <file> – Show lines that do not match the pattern. • grep -l <pattern> <files> – Show the names of files that contain the pattern. • grep -n <pattern> <file> – Show line numbers along with matching lines. • grep -c <pattern> <file> – Count the number of lines matching the pattern. 9. sed (Stream Editor) Purpose: Stream editor for performing basic text transformations. • sed 's/old/new/' <file> – Replace the first occurrence of old with new in each line. • sed 's/old/new/g' <file> – Replace all occurrences of old with new in each line. • sed -i 's/old/new/g' <file> – In-place replace all occurrences of old with new. • sed '2d' <file> – Delete the second line of a file. • sed '1,3d' <file> – Delete lines 1 to 3. • sed -n '5,10p' <file> – Print lines 5 to 10. • sed -e 's/foo/bar/g' -e 's/baz/qux/g' – Apply multiple transformations. 10. awk Purpose: A powerful programming language for pattern scanning and processing. • awk '{print $1}' <file> – Print the first column of a file. • awk '{print $1, $3}' <file> – Print the first and third columns of a file. • awk -F: '{print $1}' <file> – Set the field delimiter to : (useful for /etc/passwd). • awk '{if ($3 > 50) print $1}' <file> – Print the first column where the third column is greater than 50. • awk '{sum += $2} END {print sum}' <file> – Calculate the sum of the second column. • awk 'BEGIN {FS=","} {print $1, $2}' <file> – Use a comma as the field separator and print the first and second columns. 11. uniq Purpose: Report or omit repeated lines in a file. • uniq <file> – Display only unique lines in a file. • uniq -c <file> – Display the count of occurrences for each unique line. • uniq -d <file> – Show only the duplicate lines. • sort <file> | uniq – Remove duplicate lines (sort first to group duplicates). 12. wc (Word Count) Purpose: Count lines, words, and characters in a file. • wc <file> – Display the line, word, and byte counts for a file. • wc -l <file> – Display only the line count. • wc -w <file> – Display only the word count. • wc -c <file> – Display only the byte count. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Advanced Text Processing Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 13. tee Purpose: Read from standard input and write to both standard output and files simultaneously. • command | tee <file> – Save the output of a command to a file and display it on the terminal. • command | tee -a <file> – Append output to a file rather than overwriting it. • command | tee >(command2) – Pipe the output to another command in parallel. 14. diff Purpose: Compare files line by line. • diff <file1> <file2> – Compare two files and show differences. • diff -u <file1> <file2> – Show differences in unified format (easier to read). • diff -r <dir1> <dir2> – Recursively compare two directories. • diff -q <file1> <file2> – Show only whether files differ or not (suppress detailed output). 15. cut with Delimiters Purpose: Extract specific sections of text or files. • cut -d, -f1,2 <file> – Extract the first and second fields from a comma-separated file. • cut -f1,3 -d" " – Extract the first and third fields, using space as delimiter. 16. paste Purpose: Merge lines of files. • paste file1 file2 – Merge lines from two files side by side, separated by tabs. • paste -d, file1 file2 – Merge files side by side, using a comma as a delimiter. 17. sort with Advanced Options Purpose: Sort text or files with advanced options. • sort -t, -k2,2 <file> – Sort a comma-separated file by the second column. • sort -k2 -n <file> – Sort the second column numerically. • sort -u <file> – Sort and remove duplicate lines. 18. column Purpose: Format text into neatly aligned columns. • column -t <file> – Format the file content into neatly aligned columns. • echo -e "a b c\nd e f" | column -t – Convert space-separated text into columns. 19. sed with Regular Expressions Purpose: Perform complex text transformations using regex. • sed 's/\b[a-z]/\U&/g' <file> – Capitalize the first letter of each word using word boundaries (\b). • sed 's/\([0-9]\)/[\1]/g' <file> – Add square brackets around digits. • sed 's/\(.\*\)/prefix\_\1/' <file> – Add a prefix to each line of the file. 20. xargs Purpose: Build and execute commands from input. • find . -name "\*.log" | xargs rm – Delete all .log files found by find. • echo "file1 file2 file3" | xargs cp -t /destination/ – Copy multiple files to a directory. • cat file\_list.txt | xargs -n 1 curl -O – Download a list of files from URLs specified in a file. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Log Analysis and File Search \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 21. grep with Regular Expressions Purpose: Search for complex patterns in files using regular expressions. • grep -P '^\d{4}-\d{2}-\d{2}' <file> – Search for lines that match a specific date format (e.g., YYYY-MM-DD). • grep -E '^start|end$' <file> – Search for lines that match either "start" or "end" at the beginning or end of a line. - create exampleas from scratch and apply all thsese comamnds

###### ChatGPT said:

Here are examples and explanations for each of the basic, intermediate, and advanced text processing commands in Linux:

### ****Basic Text Processing Commands****

1. **cat (Concatenate)**
   * Purpose: Display the contents of a file or concatenate multiple files.
   * Example:

bash

CopyEdit

# Display content of a file

cat file.txt

# Concatenate and display content of two files

cat file1.txt file2.txt

# Redirect the content of a file to a new file

cat file.txt > newfile.txt

# Append content of a file to another file

cat file.txt >> anotherfile.txt

1. **echo**
   * Purpose: Print text or variables to the terminal or redirect to a file.
   * Example:

bash

CopyEdit

# Display a string

echo "Hello World"

# Display the value of an environment variable

echo $HOME

# Write output to a file

echo "Hello World" > output.txt

# Print without a trailing newline

echo -n "Hello"

1. **head**
   * Purpose: Display the first few lines of a file.
   * Example:

bash

CopyEdit

# Display the first 10 lines of a file (default behavior)

head file.txt

# Display the first 20 lines of a file

head -n 20 file.txt

# Display the first 100 bytes of a file

head -c 100 file.txt

1. **tail**
   * Purpose: Display the last few lines of a file.
   * Example:

bash

CopyEdit

# Display the last 10 lines of a file (default behavior)

tail file.txt

# Display the last 20 lines of a file

tail -n 20 file.txt

# Display the last lines of a file and follow changes (useful for log files)

tail -f /var/log/syslog

# Display the last 50 bytes of a file

tail -c 50 file.txt

1. **cut**
   * Purpose: Remove sections from each line of a file (extract columns).
   * Example:

bash

CopyEdit

# Display the first field of each line using ':' as the delimiter

cut -d':' -f1 /etc/passwd

# Display the first and third columns of a CSV file

cut -d',' -f1,3 file.csv

# Display the first 10 characters of each line

cut -c 1-10 file.txt

1. **tr**
   * Purpose: Translate or delete characters from the input.
   * Example:

bash

CopyEdit

# Convert lowercase characters to uppercase

echo "hello" | tr 'a-z' 'A-Z'

# Remove spaces from a file

tr -d '[:space:]' < file.txt

# Replace spaces with newlines

echo "123 456 789" | tr ' ' '\n'

1. **sort**
   * Purpose: Sort lines in a file or input.
   * Example:

bash

CopyEdit

# Sort lines of a file alphabetically

sort file.txt

# Sort lines numerically

sort -n numbers.txt

# Sort lines in reverse order

sort -r file.txt

# Sort and remove duplicate lines

sort -u file.txt

# Sort a CSV file by the first column

sort -t, -k1,1 file.csv

### ****Intermediate Text Processing Commands****

1. **grep**
   * Purpose: Search for patterns in text files using regular expressions.
   * Example:

bash

CopyEdit

# Search for a pattern in a file

grep "pattern" file.txt

# Perform a case-insensitive search

grep -i "pattern" file.txt

# Search recursively through a directory

grep -r "pattern" /path/to/directory

# Show lines that do not match the pattern

grep -v "pattern" file.txt

# Show file names that contain the pattern

grep -l "pattern" \*

# Show line numbers along with matching lines

grep -n "pattern" file.txt

# Count the number of matching lines

grep -c "pattern" file.txt

1. **sed (Stream Editor)**
   * Purpose: Stream editor for performing basic text transformations.
   * Example:

bash

CopyEdit

# Replace the first occurrence of 'old' with 'new' in each line

sed 's/old/new/' file.txt

# Replace all occurrences of 'old' with 'new' in each line

sed 's/old/new/g' file.txt

# In-place replacement (modifies the file)

sed -i 's/old/new/g' file.txt

# Delete the second line of a file

sed '2d' file.txt

# Delete lines 1 to 3

sed '1,3d' file.txt

# Print lines 5 to 10

sed -n '5,10p' file.txt

# Apply multiple transformations

sed -e 's/foo/bar/g' -e 's/baz/qux/g' file.txt

1. **awk**

* Purpose: A powerful programming language for pattern scanning and processing.
* Example:

bash

CopyEdit

# Print the first column of a file

awk '{print $1}' file.txt

# Print the first and third columns of a file

awk '{print $1, $3}' file.txt

# Use ':' as the field separator and print the first column

awk -F: '{print $1}' /etc/passwd

# Print the first column where the third column is greater than 50

awk '{if ($3 > 50) print $1}' file.txt

# Calculate the sum of the second column

awk '{sum += $2} END {print sum}' file.txt

# Use a comma as the field separator and print the first and second columns

awk 'BEGIN {FS=","} {print $1, $2}' file.csv

1. **uniq**

* Purpose: Report or omit repeated lines in a file.
* Example:

bash

CopyEdit

# Display only unique lines in a file

uniq file.txt

# Display the count of occurrences for each unique line

uniq -c file.txt

# Show only the duplicate lines

uniq -d file.txt

# Sort the file first, then remove duplicate lines

sort file.txt | uniq

1. **wc (Word Count)**

* Purpose: Count lines, words, and characters in a file.
* Example:

bash

CopyEdit

# Display line, word, and byte counts for a file

wc file.txt

# Display only the line count

wc -l file.txt

# Display only the word count

wc -w file.txt

# Display only the byte count

wc -c file.txt

### ****Advanced Text Processing Commands****

1. **tee**

* Purpose: Read from standard input and write to both standard output and files simultaneously.
* Example:

bash

CopyEdit

# Save the output of a command to a file and display it on the terminal

echo "Hello World" | tee output.txt

# Append output to a file rather than overwriting it

echo "Additional text" | tee -a output.txt

# Pipe the output to another command in parallel

echo "Hello" | tee >(grep 'H')

1. **diff**

* Purpose: Compare files line by line.
* Example:

bash

CopyEdit

# Compare two files and show differences

diff file1.txt file2.txt

# Show differences in unified format

diff -u file1.txt file2.txt

# Recursively compare two directories

diff -r dir1/ dir2/

# Show only whether files differ or not

diff -q file1.txt file2.txt

1. **cut with Delimiters**

* Purpose: Extract specific sections of text or files.
* Example:

bash

CopyEdit

# Extract the first and second fields from a comma-separated file

cut -d, -f1,2 file.csv

# Extract the first and third fields, using space as delimiter

cut -f1,3 -d" " file.txt

1. **paste**

* Purpose: Merge lines of files.
* Example:

bash

CopyEdit

# Merge lines from two files side by side, separated by tabs

paste file1.txt file2.txt

# Merge files side by side, using a comma as a delimiter

paste -d, file1.txt file2.txt

1. **sort with Advanced Options**

* Purpose: Sort text or files with advanced options.
* Example:

bash

CopyEdit

# Sort a comma-separated file by the second column

sort -t, -k2,2 file.csv

# Sort the second column numerically

sort -k2 -n file.txt

# Sort and remove duplicate lines

sort -u file.txt

1. **column**

* Purpose: Format text into neatly aligned columns.
* Example:

bash

CopyEdit

# Format the file content into neatly aligned columns

column -t file.txt

# Convert space-separated text into columns

echo -e "a b c\nd e f" | column -t

1. **sed with Regular Expressions**

* Purpose: Perform complex text transformations using regex.
* Example:

bash

CopyEdit

# Capitalize the first letter of each word

sed 's/\b[a-z]/\U&/g' file.txt

# Add square brackets around digits

sed 's/\([0-9]\)/[\1]/g' file.txt

# Add a prefix to each line of the file

sed 's/\(.\*\)/prefix\_\1/' file.txt

1. **xargs**

* Purpose: Build and execute commands from input.
* Example:

bash

CopyEdit

# Delete all .log files found by find

find . -name "\*.log" | xargs rm

# Copy multiple files to a directory

echo "file1 file2 file3" | xargs cp -t /destination/

# Download a list of files from URLs specified in a file

cat file\_list.txt | xargs -n 1 curl -O

### ****Log Analysis and File Search****

1. **grep with Regular Expressions**

* Purpose: Search for complex patterns in files using regular expressions.
* Example:

bash

CopyEdit

# Search for lines that match a specific date format (e.g., YYYY-MM-DD)

grep -P '^\d{4}-\d{2}-\d{2}' file.txt

# Search for lines that match either "start" or "end" at the beginning or end of a line

grep -E '^start|end$' file.txt

These examples cover a wide variety of tasks that can be performed using the Linux text processing tools, ranging from simple file manipulations to complex search and transformation operations.

4o

##### You said:

22. split Purpose: Split a file into pieces based on lines or size. • split <file> – Split a file into multiple smaller files (default splits by 1000 lines). • split -l <number> <file> – Split the file into chunks of <number> lines (e.g., split -l 500 file.txt). • split -b <size> <file> – Split the file into chunks of a specified size (e.g., split -b 10M file.txt). • split -d -l <number> <file> – Use numeric suffixes instead of alphabetical ones. • split -a <length> <file> – Set the length of the suffix (default is 2 characters). 23. fmt Purpose: Simple text formatter that formats text to a specific width. • fmt <file> – Format the text to a default width of 75 characters. • fmt -w <width> <file> – Set a specific line width (e.g., fmt -w 50 file.txt). • fmt -c <file> – Display the file with the original spacing and paragraph structure. 24. tr (Advanced) Purpose: Perform advanced transformations on characters. • echo "aabbcc" | tr -d "a" – Delete all occurrences of the character "a". • echo "hello" | tr 'a-z' 'A-Z' – Convert lowercase letters to uppercase. • echo "hello 123" | tr -cd '[:alnum:]' – Delete everything except alphanumeric characters (letters and numbers). • echo "hello world" | tr ' ' '\n' – Replace spaces with newlines. 25. nl Purpose: Number the lines of a file. • nl <file> – Number the lines of a file. • nl -ba <file> – Number all lines, including empty lines. • nl -s ' - ' <file> – Customize the separator between the line number and text (e.g., using ' - '). • nl -n ln <file> – Number lines with leading zeros. 26. comm Purpose: Compare two sorted files line by line. • comm <file1> <file2> – Compare two files line by line and output three columns: lines only in file1, lines only in file2, and common lines. • comm -1 <file1> <file2> – Suppress lines that appear only in the first file. • comm -2 <file1> <file2> – Suppress lines that appear only in the second file. • comm -3 <file1> <file2> – Suppress lines that appear in both files. 27. tee with Multiple Output Files Purpose: Write to multiple files simultaneously while also displaying output on the terminal. • command | tee file1 file2 – Write output to both file1 and file2 and also display it on the terminal. • command | tee -a file1 – Append the output of a command to file1. 28. paste with Custom Delimiters Purpose: Merge lines of files side by side with custom delimiters. • paste file1 file2 – Merge two files side by side with a tab delimiter. • paste -d, file1 file2 – Use a comma as a delimiter. • paste -d'|' file1 file2 – Use a pipe (|) as a delimiter. • paste -s file1 file2 – Paste lines of the files serially (as opposed to side by side). 29. rev Purpose: Reverse the lines of a file. • rev <file> – Reverse the characters of each line in the file. • echo "hello" | rev – Reverse the string "hello" to "olleh". 30. xargs (Advanced) Purpose: Construct argument lists and execute commands using input from standard input or files. • find . -name "\*.txt" | xargs wc -l – Count the lines in all .txt files in the current directory and subdirectories. • echo "file1 file2" | xargs -n 1 cp – Copy each file one by one. • echo "file1 file2 file3" | xargs -I {} mv {} /destination/ – Move multiple files to a specific directory using placeholders. • find /path -type f -print0 | xargs -0 rm – Delete files safely with handling special characters (using -0 and -print0). 31. sort with Complex Sorting Purpose: Sort files based on different criteria (e.g., numeric, by column). • sort <file> – Sort a file alphabetically. • sort -n <file> – Sort the file numerically (useful for numbers instead of alphabetical sorting). • sort -r <file> – Sort in reverse order. • sort -k2 <file> – Sort by the second column. • sort -t, -k2,2 <file> – Sort a CSV file by the second column (using comma as the delimiter). 32. grep with Extended Regular Expressions Purpose: Use advanced regular expressions for more complex pattern matching. • grep -E 'foo|bar' <file> – Search for "foo" or "bar" (extended regular expressions). • grep -P '\d{3}-\d{2}-\d{4}' <file> – Use Perl-compatible regex to search for a social security number pattern. • grep -o 'foo.\*' <file> – Output only the matched portion of the lines that contain "foo" and everything after it. 33. sed for In-place Editing Purpose: Edit files directly and apply transformations. • sed -i 's/old/new/g' <file> – In-place replace all occurrences of old with new in the file. • sed -i '1s/^/Hello /' <file> – Add "Hello " at the beginning of the first line in the file. • sed -i 's/^[^ ]\*/&\t/' <file> – Insert a tab after the first word in each line. • sed -i '/pattern/d' <file> – Delete all lines containing a specific pattern. 34. awk (Advanced Usage) Purpose: Use awk for advanced text processing with pattern matching and text transformation. • awk '{print $1, $2}' <file> – Print the first two fields (columns) of a file. • awk '$1 > 100 {print $1, $2}' <file> – Print the first and second columns where the first column is greater than 100. • awk -F, '{print $1, $3}' <file> – Use comma as a field separator and print the first and third columns. • awk 'BEGIN {FS=":"} {print $1}' <file> – Set the field separator to colon and print the first column. • awk '{sum += $2} END {print sum}' <file> – Sum the values in the second column and print the result at the end. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Working with JSON and XML \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 35. jq Purpose: Process JSON data (a powerful tool for working with structured data). • cat file.json | jq . – Pretty-print the contents of a JSON file. • cat file.json | jq '.key' – Extract the value of a specific key. • cat file.json | jq '.items[] | .name' – Extract the "name" field from all items in an array. • cat file.json | jq '.user | {name: .name, age: .age}' – Extract specific keys and output them in a new structure. • cat file.json | jq '.users | length' – Get the number of elements in a JSON array. 36. xmlstarlet Purpose: Process and transform XML data. • xmlstarlet fo <file.xml> – Format an XML file (prettify). • xmlstarlet sel -t -v "//name" <file.xml> – Extract all values of the <name> tag. • xmlstarlet ed -d "//book[price>100]" <file.xml> – Delete all <book> elements where the price is greater than 100. • xmlstarlet fo file.xml > file\_pretty.xml – Format an XML file and save it. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Log Processing \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 37. logrotate Purpose: Rotate logs and manage log files. • /etc/logrotate.conf – Configuration file for log rotation settings. • logrotate -d /etc/logrotate.conf – Debug and simulate the log rotation process. • logrotate -f /etc/logrotate.conf – Force a log rotation based on the configuration file. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Miscellaneous Advanced Text Processing Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 38. fmt for Text Wrapping Purpose: Automatically wrap text at a specific line length. • fmt -w <width> <file> – Wrap text to a specified width (e.g., fmt -w 50 to wrap lines at 50 characters). • fmt -c <file> – Format a file while preserving the original structure.

###### ChatGPT said:

Here are detailed explanations and examples for the next set of text processing and file manipulation commands, starting from split to working with JSON and XML:

### ****Text Processing Commands****

#### 22. split

* **Purpose**: Split a file into smaller parts based on lines or size.
* **Examples**:

bash

CopyEdit

# Split the file into 1000-line chunks (default behavior)

split file.txt

# Split the file into chunks of 500 lines each

split -l 500 file.txt

# Split the file into chunks of 10 MB each

split -b 10M file.txt

# Use numeric suffixes instead of alphabetical ones

split -d -l 500 file.txt

# Set the length of the suffix to 3 characters

split -a 3 -l 500 file.txt

#### 23. fmt

* **Purpose**: Simple text formatter that formats text to a specific width.
* **Examples**:

bash

CopyEdit

# Format text to a default width of 75 characters

fmt file.txt

# Set a custom line width of 50 characters

fmt -w 50 file.txt

# Display file with the original spacing and paragraph structure

fmt -c file.txt

#### 24. tr **(Advanced)**

* **Purpose**: Perform advanced transformations on characters.
* **Examples**:

bash

CopyEdit

# Delete all occurrences of the character "a"

echo "aabbcc" | tr -d "a"

# Convert lowercase letters to uppercase

echo "hello" | tr 'a-z' 'A-Z'

# Delete everything except alphanumeric characters

echo "hello 123" | tr -cd '[:alnum:]'

# Replace spaces with newlines

echo "hello world" | tr ' ' '\n'

#### 25. nl

* **Purpose**: Number the lines of a file.
* **Examples**:

bash

CopyEdit

# Number the lines of a file

nl file.txt

# Number all lines, including empty lines

nl -ba file.txt

# Customize the separator between line numbers and text

nl -s ' - ' file.txt

# Number lines with leading zeros

nl -n ln file.txt

#### 26. comm

* **Purpose**: Compare two sorted files line by line.
* **Examples**:

bash

CopyEdit

# Compare two files line by line and output three columns: unique to file1, unique to file2, and common lines

comm file1.txt file2.txt

# Suppress lines that appear only in the first file

comm -1 file1.txt file2.txt

# Suppress lines that appear only in the second file

comm -2 file1.txt file2.txt

# Suppress lines that appear in both files

comm -3 file1.txt file2.txt

#### 27. tee **with Multiple Output Files**

* **Purpose**: Write to multiple files simultaneously while also displaying output on the terminal.
* **Examples**:

bash

CopyEdit

# Write output to both file1 and file2 and also display it on the terminal

command | tee file1.txt file2.txt

# Append output to file1

command | tee -a file1.txt

#### 28. paste **with Custom Delimiters**

* **Purpose**: Merge lines of files side by side with custom delimiters.
* **Examples**:

bash

CopyEdit

# Merge two files side by side with a tab delimiter

paste file1.txt file2.txt

# Merge files side by side, using a comma as a delimiter

paste -d, file1.txt file2.txt

# Merge files side by side, using a pipe (|) as a delimiter

paste -d'|' file1.txt file2.txt

# Merge files serially (instead of side by side)

paste -s file1.txt file2.txt

#### 29. rev

* **Purpose**: Reverse the lines of a file.
* **Examples**:

bash

CopyEdit

# Reverse the characters of each line in the file

rev file.txt

# Reverse the string "hello" to "olleh"

echo "hello" | rev

#### 30. xargs **(Advanced)**

* **Purpose**: Construct argument lists and execute commands using input from standard input or files.
* **Examples**:

bash

CopyEdit

# Count the lines in all .txt files in the current directory and subdirectories

find . -name "\*.txt" | xargs wc -l

# Copy each file one by one

echo "file1 file2" | xargs -n 1 cp

# Move multiple files to a specific directory using placeholders

echo "file1 file2 file3" | xargs -I {} mv {} /destination/

# Delete files safely with handling special characters

find /path -type f -print0 | xargs -0 rm

#### 31. sort **with Complex Sorting**

* **Purpose**: Sort files based on different criteria (e.g., numeric, by column).
* **Examples**:

bash

CopyEdit

# Sort a file alphabetically

sort file.txt

# Sort the file numerically

sort -n numbers.txt

# Sort in reverse order

sort -r file.txt

# Sort by the second column

sort -k2 file.txt

# Sort a CSV file by the second column (using a comma as the delimiter)

sort -t, -k2,2 file.csv

#### 32. grep **with Extended Regular Expressions**

* **Purpose**: Use advanced regular expressions for more complex pattern matching.
* **Examples**:

bash

CopyEdit

# Search for "foo" or "bar" (extended regular expressions)

grep -E 'foo|bar' file.txt

# Use Perl-compatible regex to search for a social security number pattern

grep -P '\d{3}-\d{2}-\d{4}' file.txt

# Output only the matched portion of the lines containing "foo" and everything after it

grep -o 'foo.\*' file.txt

#### 33. sed **for In-place Editing**

* **Purpose**: Edit files directly and apply transformations.
* **Examples**:

bash

CopyEdit

# In-place replace all occurrences of 'old' with 'new'

sed -i 's/old/new/g' file.txt

# Add "Hello " at the beginning of the first line in the file

sed -i '1s/^/Hello /' file.txt

# Insert a tab after the first word in each line

sed -i 's/^[^ ]\*/&\t/' file.txt

# Delete all lines containing a specific pattern

sed -i '/pattern/d' file.txt

#### 34. awk **(Advanced Usage)**

* **Purpose**: Use awk for advanced text processing with pattern matching and text transformation.
* **Examples**:

bash

CopyEdit

# Print the first two fields (columns) of a file

awk '{print $1, $2}' file.txt

# Print the first and second columns where the first column is greater than 100

awk '$1 > 100 {print $1, $2}' file.txt

# Use comma as the field separator and print the first and third columns

awk -F, '{print $1, $3}' file.csv

# Set the field separator to colon and print the first column

awk 'BEGIN {FS=":"} {print $1}' file.txt

# Sum the values in the second column and print the result at the end

awk '{sum += $2} END {print sum}' file.txt

### ****Working with JSON and XML****

#### 35. jq

* **Purpose**: Process JSON data.
* **Examples**:

bash

CopyEdit

# Pretty-print the contents of a JSON file

cat file.json | jq .

# Extract the value of a specific key

cat file.json | jq '.key'

# Extract the "name" field from all items in an array

cat file.json | jq '.items[] | .name'

# Extract specific keys and output them in a new structure

cat file.json | jq '.user | {name: .name, age: .age}'

# Get the number of elements in a JSON array

cat file.json | jq '.users | length'

#### 36. xmlstarlet

* **Purpose**: Process and transform XML data.
* **Examples**:

bash

CopyEdit

# Format an XML file (prettify)

xmlstarlet fo <file.xml>

# Extract all values of the <name> tag

xmlstarlet sel -t -v "//name" <file.xml>

# Delete all <book> elements where the price is greater than 100

xmlstarlet ed -d "//book[price>100]" <file.xml>

# Format an XML file and save it

xmlstarlet fo file.xml > file\_pretty.xml

### ****Log Processing****

#### 37. logrotate

* **Purpose**: Rotate logs and manage log files.
* **Examples**:

bash

CopyEdit

# Configuration file for log rotation settings

cat /etc/logrotate.conf

# Debug and simulate the log rotation process

logrotate -d /etc/logrotate.conf

# Force a log rotation based on the configuration file

logrotate -f /etc/logrotate.conf

### ****Miscellaneous Advanced Text Processing Commands****

#### 38. fmt **for Text Wrapping**

* **Purpose**: Automatically wrap text at a specific line length.
* **Examples**:

bash

CopyEdit

# Wrap text to a specified width (e.g., fmt -w 50 to wrap lines at 50 characters)

fmt -w 50 file.txt

# Format a file while preserving the original structure

fmt -c file.txt

##### You said:

Compression and Archiving Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 1. tar (Tape Archive) Purpose: Create archives and extract files. • tar -cf archive.tar <files> – Create an uncompressed archive from files or directories. • tar -czf archive.tar.gz <files> – Create a .tar.gz compressed archive (gzip). • tar -cjf archive.tar.bz2 <files> – Create a .tar.bz2 compressed archive (bzip2). • tar -cJf archive.tar.xz <files> – Create a .tar.xz compressed archive (xz). • tar -xf archive.tar – Extract files from a .tar archive. • tar -xzf archive.tar.gz – Extract files from a .tar.gz archive. • tar -xjf archive.tar.bz2 – Extract files from a .tar.bz2 archive. • tar -xJf archive.tar.xz – Extract files from a .tar.xz archive. • tar -tvf archive.tar – List the contents of a .tar archive without extracting. • tar -xzvf archive.tar.gz – Extract a .tar.gz archive with verbose output. • tar -cf archive.tar --exclude='\*.log' <directory> – Create an archive excluding .log files. 2. gzip Purpose: Compress files using the gzip algorithm. • gzip <file> – Compress a file to .gz format (e.g., file.txt becomes file.txt.gz). • gzip -d <file.gz> – Decompress a .gz file. • gzip -c <file> – Compress a file and write to stdout, leaving the original file intact. • gzip -k <file> – Keep the original file while compressing. • gzip -r <directory> – Compress all files in a directory recursively. • gzip -l <file.gz> – Display information about a compressed file. 3. gunzip Purpose: Decompress files that were compressed with gzip. • gunzip <file.gz> – Decompress a .gz file. • gunzip -c <file.gz> – Decompress a .gz file and write the output to stdout. • gunzip -k <file.gz> – Decompress a .gz file and keep the original file. 4. bzip2 Purpose: Compress files using the bzip2 algorithm. • bzip2 <file> – Compress a file to .bz2 format. • bzip2 -d <file.bz2> – Decompress a .bz2 file. • bzip2 -c <file> – Compress a file and write the output to stdout. • bzip2 -k <file> – Keep the original file while compressing. • bzip2 -z <file> – Compress a file using bzip2. • bzip2 -l <file.bz2> – Display compression statistics of a .bz2 file. 5. bunzip2 Purpose: Decompress files that were compressed with bzip2. • bunzip2 <file.bz2> – Decompress a .bz2 file. • bunzip2 -k <file.bz2> – Decompress and keep the original .bz2 file. 6. xz Purpose: Compress files using the xz algorithm, which provides higher compression ratios. • xz <file> – Compress a file to .xz format. • xz -d <file.xz> – Decompress a .xz file. • xz -c <file> – Compress a file and write to stdout, leaving the original file intact. • xz -k <file> – Keep the original file while compressing. • xz -r <directory> – Compress all files in a directory recursively. • xz -l <file.xz> – Display information about a .xz file. • xz -z <file> – Compress a file with xz. 7. 7z (7-Zip) Purpose: High compression file archiver, supports multiple formats. • 7z a archive.7z <files> – Create a .7z archive from files. • 7z x archive.7z – Extract files from a .7z archive. • 7z t archive.7z – Test the integrity of a .7z archive. • 7z l archive.7z – List the contents of a .7z archive. • 7z e archive.7z – Extract files from a .7z archive (without directory structure). • 7z a archive.7z -p<password> <files> – Create an encrypted archive with a password. 8. zip Purpose: Create and extract .zip archives. • zip archive.zip <files> – Create a .zip archive. • zip -r archive.zip <directory> – Recursively zip all files in a directory. • zip -e archive.zip <files> – Create an encrypted .zip archive. • zip -d archive.zip <file> – Delete a file from a .zip archive. • zip -u archive.zip <file> – Update an existing .zip archive by adding new files. • unzip archive.zip – Extract files from a .zip archive. • unzip -l archive.zip – List the contents of a .zip archive. • unzip -d <directory> archive.zip – Extract the .zip archive into a specific directory. 9. unzip Purpose: Extract files from .zip archives. • unzip <file.zip> – Extract a .zip archive. • unzip -d <directory> <file.zip> – Extract to a specific directory. • unzip -l <file.zip> – List the contents of a .zip archive without extracting. • unzip -o <file.zip> – Overwrite existing files when extracting. 10. rar and unrar Purpose: Create and extract .rar archives. • rar a archive.rar <files> – Create a .rar archive. • unrar x archive.rar – Extract the contents of a .rar archive. • unrar l archive.rar – List the contents of a .rar archive. • unrar e archive.rar – Extract files to the current directory. 11. zstd (Zstandard) Purpose: Fast compression algorithm, often used in modern applications. • zstd <file> – Compress a file using zstd (creates a .zst file). • zstd -d <file.zst> – Decompress a .zst file. • zstd -c <file> – Compress a file to stdout, keeping the original. • zstd -k <file> – Compress a file while keeping the original file. 12. ar Purpose: Create, modify, and extract from archive files (commonly used for creating .deb packages). • ar -cvf archive.a <files> – Create an archive .a file. • ar -x archive.a – Extract files from an archive .a file. • ar -t archive.a – List the contents of an archive .a file. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Compression/Archiving with File Management Tools \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 13. rsync Purpose: Efficient file synchronization and transfer, including compression. • rsync -av <source> <destination> – Copy files from source to destination while preserving attributes. • rsync -avz <source> <destination> – Compress files during transfer using gzip compression. • rsync -avz --delete <source> <destination> – Synchronize directories and delete files in the destination that no longer exist in the source. • rsync -z – Compress file data during the transfer. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Using Compression/Archiving for Backup and Automation \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 14. tar for Backup Purpose: Create a backup of a directory or entire system. • tar -czf backup.tar.gz <directory> – Create a compressed backup of a directory. • tar -czf backup.tar.gz --exclude='\*.log' <directory> – Exclude specific file types (e.g., .log files) from the backup. • tar -czf backup\_$(date +%F).tar.gz <directory> – Create a date-stamped backup archive. • tar -xzf backup.tar.gz -C /path/to/restore/ – Extract backup to a specific directory. 15. logrotate Purpose: Automatically rotate, compress, and remove old log files. • /etc/logrotate.conf – The main configuration file for logrotate. • logrotate -d /etc/logrotate.conf – Debug and simulate log rotation. • logrotate -f /etc/logrotate.conf – Force the rotation of logs according to the configuration file. 16. sort with Complex Options Purpose: Sort lines in text files or output streams based on various criteria. • sort -t',' -k2,2 <file> – Sort a CSV file by the second column (using a comma as the delimiter). • sort -k1,1n -k2,2r <file> – Sort first by the first column (numerically), then by the second column (in reverse order). • sort -u <file> – Sort and remove duplicate lines. • sort -o sorted.txt <file> – Sort the file and output the result into a new file. • sort -n -k3 <file> – Sort a file numerically based on the third column. 17. uniq with Advanced Flags Purpose: Filter out duplicate lines in a file or stream of text. • uniq <file> – Remove consecutive duplicate lines from a file. • uniq -c <file> – Prefix each line with the count of occurrences. • uniq -d <file> – Display only lines that are duplicated. • uniq -u <file> – Display only unique lines (lines that are not repeated). • sort <file> | uniq -c – Sort a file first and then count duplicates. 18. grep with Extended and Perl-Compatible Regex Purpose: Perform complex pattern matching using regular expressions. • grep -P 'pattern' <file> – Use Perl-compatible regular expressions (PCRE) to search. • grep -E 'pattern' <file> – Use extended regular expressions (ERE). • grep -o 'pattern' <file> – Print only the matched part of the lines, rather than the entire line. • grep -A <num> <pattern> <file> – Show <num> lines after the matching line. • grep -B <num> <pattern> <file> – Show <num> lines before the matching line. • grep -C <num> <pattern> <file> – Show <num> lines before and after the matching line. 19. sed (Stream Editor) with Advanced Patterns Purpose: Use sed for complex text replacements, substitutions, and transformations. • sed 's/\b[[:alpha:]]\*/\U&/g' <file> – Capitalize every word in a file. • sed 's/^\([0-9]\)/[\1]/g' <file> – Add square brackets around digits at the start of lines. • sed -n 's/hello/world/p' <file> – Replace 'hello' with 'world' and print only lines that match. • sed -i 's/foo/bar/g' <file> – Replace all occurrences of foo with bar in place (modifying the file). • sed '2d' <file> – Delete the second line of a file. • sed '/pattern/d' <file> – Delete lines that match a pattern. 20. awk (Advanced Usage) Purpose: Powerful programming language for pattern scanning and processing. • awk '{print $1}' <file> – Print the first column of each line. • awk '{sum += $2} END {print sum}' <file> – Calculate the sum of the second column. • awk 'BEGIN {FS=":"} {print $1}' <file> – Set the field separator to colon (:) and print the first column. • awk '{if ($3 > 50) print $1, $3}' <file> – Print the first and third columns where the third column is greater than 50. • awk '{print NR, $0}' <file> – Print line number along with the entire line. • awk 'BEGIN {OFS="\t"} {print $1, $2}' <file> – Set the output field separator (OFS) to a tab and print the first and second columns. 21. find with exec for Advanced Text Processing Purpose: Combine find with other commands for powerful search and text processing. • find . -name "\*.log" -exec cat {} \; – Find all .log files and print their contents. • find . -name "\*.txt" -exec sed -i 's/foo/bar/g' {} \; – Replace "foo" with "bar" in all .txt files. • find . -name "\*.log" -exec grep -H "error" {} \; – Search for "error" in all .log files and show the filename. • find . -type f -exec awk '{print $1}' {} \; – Print the first column from all files found. 22. xargs with Multiple Arguments Purpose: Build and execute commands from standard input, handle multiple arguments. • find . -name "\*.log" | xargs wc -l – Count lines in all .log files found by find. • echo "file1 file2 file3" | xargs -n 1 cp – Copy each file one by one to a destination. • echo "file1 file2 file3" | xargs -I {} mv {} /destination/ – Move multiple files to a specific directory. • echo "file1 file2 file3" | xargs -n 2 rm – Delete files in pairs (run rm twice for every two files). 23. tee with Filters Purpose: Read from standard input, write to both standard output and files. • echo "output" | tee file.txt – Write the output to file.txt and also display it on the terminal. • echo "output" | tee >(grep "pattern") – Send the output to a filter and display the results. • echo "output" | tee file1 file2 – Write the output to multiple files. 24. tr for Multi-character Replacement Purpose: Transform or delete characters. • echo "hello world" | tr 'a-z' 'A-Z' – Convert all lowercase letters to uppercase. • echo "foo 123 bar" | tr -d '[:digit:]' – Remove all digits from the input. • echo "apple banana cherry" | tr ' ' '\n' – Replace spaces with newlines to separate words. • echo "a,b,c" | tr ',' ';' – Replace commas with semicolons. 25. nl (Number Lines) Purpose: Number the lines in a file. • nl <file> – Number the lines in the file. • nl -ba <file> – Number all lines, including empty ones. • nl -s ' ' <file> – Set the separator between the line number and text to a space. • nl -n ln <file> – Use leading zeros for line numbers. 26. split with Custom File Sizes Purpose: Split a file into smaller files based on size or number of lines. • split -l <lines> <file> – Split the file into smaller files, each containing <lines> lines. • split -b <size> <file> – Split the file into smaller files, each of size <size> (e.g., split -b 5M file.txt to split into 5MB chunks). • split -d -l <lines> <file> – Use numeric suffixes instead of the default alphabetic ones. 27. column for Formatted Output Purpose: Format text into neat columns. • column -t <file> – Format the contents of a file into neatly aligned columns. • echo -e "a b c\nd e f" | column -t – Convert space-separated text into aligned columns. • cat file.txt | column -s ',' -t – Format comma-separated values into columns. 28. logrotate for Automated Log File Management Purpose: Automatically rotate, compress, and manage log files. • logrotate /etc/logrotate.conf – Apply log rotation rules specified in the configuration file. • logrotate -d /etc/logrotate.conf – Dry-run to simulate log rotation. • logrotate -f /etc/logrotate.conf – Force log rotation regardless of the time interval. • /etc/logrotate.d/<service> – Service-specific configuration files for rotating logs (e.g., for Apache, MySQL). 29. jq for JSON Processing Purpose: Parse and manipulate JSON data. • cat file.json | jq . – Pretty-print the JSON content of a file. • cat file.json | jq '.key' – Extract the value of a specific key. • cat file.json | jq '.users[] | .name' – Extract the "name" field from each element of an array. • cat file.json | jq '.user | {name: .name, age: .age}' – Extract specific keys and output them in a custom structure. - - create exampleas from scratch and apply all thsese comamnds

###### ChatGPT said:

Sure! Below is a simple example that demonstrates how to use all of the listed commands, from creating and managing archives to using advanced text processing, all the way to handling logs, backups, and JSON processing.

### 1. ****Create and Extract Archives with**** tar

Create a folder with some files, then archive and compress it using tar.

bash

CopyEdit

mkdir my\_folder

echo "Hello, world!" > my\_folder/hello.txt

echo "Backup Data" > my\_folder/data.txt

# Create an uncompressed .tar archive

tar -cf my\_folder.tar my\_folder/

# Create a compressed .tar.gz archive

tar -czf my\_folder.tar.gz my\_folder/

# Extract files from the .tar archive

tar -xf my\_folder.tar

# Extract files from the .tar.gz archive

tar -xzf my\_folder.tar.gz

### 2. ****Compress Files with**** gzip

bash

CopyEdit

# Compress a single file using gzip

gzip hello.txt

# Compress a file without deleting the original

gzip -k data.txt

# Decompress the file

gunzip hello.txt.gz

# List gzip file info

gzip -l hello.txt.gz

### 3. ****Compress Files with**** bzip2

bash

CopyEdit

# Compress a file with bzip2

bzip2 hello.txt

# Decompress the file

bunzip2 hello.txt.bz2

# List compression stats

bzip2 -l hello.txt.bz2

### 4. ****Compress Files with**** xz

bash

CopyEdit

# Compress a file with xz

xz hello.txt

# Decompress the file

xz -d hello.txt.xz

# List xz file info

xz -l hello.txt.xz

### 5. ****Create and Extract Archives with**** 7z

bash

CopyEdit

# Create a .7z archive

7z a my\_archive.7z my\_folder/

# Extract files from the .7z archive

7z x my\_archive.7z

# Test the integrity of a .7z archive

7z t my\_archive.7z

# List contents of the .7z archive

7z l my\_archive.7z

### 6. ****Create and Extract**** .zip ****Archives****

bash

CopyEdit

# Create a .zip archive

zip my\_folder.zip my\_folder/

# Recursively zip a directory

zip -r my\_folder.zip my\_folder/

# Extract files from the .zip archive

unzip my\_folder.zip

# List contents of the .zip archive

unzip -l my\_folder.zip

### 7. ****Create and Extract**** .rar ****Archives****

bash

CopyEdit

# Create a .rar archive

rar a my\_folder.rar my\_folder/

# Extract files from the .rar archive

unrar x my\_folder.rar

# List the contents of a .rar archive

unrar l my\_folder.rar

### 8. ****Compression with**** zstd

bash

CopyEdit

# Compress a file with zstd

zstd hello.txt

# Decompress the file

zstd -d hello.txt.zst

# List zstd file info

zstd -l hello.txt.zst

### 9. ****Backup with**** tar

bash

CopyEdit

# Backup a directory with a timestamp

tar -czf backup\_$(date +%F).tar.gz my\_folder/

### 10. ****File Synchronization with**** rsync

bash

CopyEdit

# Synchronize files and compress during transfer

rsync -avz my\_folder/ remote\_server:/backup/

# Synchronize and delete files that no longer exist

rsync -avz --delete my\_folder/ remote\_server:/backup/

### 11. ****Use**** logrotate ****for Log Management****

bash

CopyEdit

# Rotate logs based on configuration

logrotate /etc/logrotate.conf

# Force log rotation

logrotate -f /etc/logrotate.conf

### 12. ****Sort Files with**** sort

bash

CopyEdit

# Sort a CSV file by the second column

sort -t',' -k2,2 my\_file.csv

# Sort and remove duplicate lines

sort -u my\_file.txt

### 13. ****Use**** uniq ****to Remove Duplicates****

bash

CopyEdit

# Filter out consecutive duplicate lines

uniq my\_file.txt

# Show only unique lines

uniq -u my\_file.txt

### 14. ****Search with**** grep ****Using Extended Regex****

bash

CopyEdit

# Search using Perl-compatible regex

grep -P 'pattern' my\_file.txt

# Show lines before and after the match

grep -C 2 'pattern' my\_file.txt

### 15. ****Text Transformation with**** sed

bash

CopyEdit

# Replace 'hello' with 'world' in place

sed -i 's/hello/world/g' my\_file.txt

# Delete the second line

sed '2d' my\_file.txt

### 16. ****Process Data with**** awk

bash

CopyEdit

# Sum the second column

awk '{sum += $2} END {print sum}' my\_file.txt

# Print the first column

awk '{print $1}' my\_file.txt

### 17. ****Use**** find ****with**** exec ****for Processing****

bash

CopyEdit

# Find all .txt files and count lines

find . -name "\*.txt" -exec wc -l {} \;

### 18. ****Process Files with**** xargs

bash

CopyEdit

# Move files listed in a text file

echo "file1 file2 file3" | xargs -I {} mv {} /destination/

### 19. ****Use**** tee ****to Write and Display Output****

bash

CopyEdit

# Write output to a file and display it

echo "Hello World" | tee output.txt

# Pipe output through grep and display the result

echo "Hello World" | tee >(grep "Hello")

### 20. ****Replace Characters with**** tr

bash

CopyEdit

# Convert lowercase to uppercase

echo "hello" | tr 'a-z' 'A-Z'

# Remove digits

echo "foo 123 bar" | tr -d '[:digit:]'

### 21. ****Number Lines with**** nl

bash

CopyEdit

# Number lines in a file

nl my\_file.txt

### 22. ****Split Files with**** split

bash

CopyEdit

# Split a file into smaller files with 100 lines each

split -l 100 large\_file.txt

# Split a file into 5MB chunks

split -b 5M large\_file.txt

### 23. ****Format Output with**** column

bash

CopyEdit

# Format space-separated values into columns

echo -e "a b c\nd e f" | column -t

# Format comma-separated values into columns

cat file.csv | column -s ',' -t

### 24. ****JSON Processing with**** jq

bash

CopyEdit

# Pretty-print JSON data

cat file.json | jq .

# Extract specific key from JSON

cat file.json | jq '.key'

# Extract an array of names

cat file.json | jq '.users[] | .name'

### Summary

This comprehensive set of examples covers creating and extracting archives, compressing files, managing logs, performing file synchronization, sorting, searching, text processing, and working with JSON data. By running these commands, you'll get hands-on experience with managing files, backups, and log rotation, while also learning how to process text and data effectively in a shell environment.

4o

##### You said:

please for all commands

###### ChatGPT said:

Below is a full example that demonstrates how to apply all the commands you requested. This will showcase each command in use, from creating archives, compressing files, and managing backups to text processing and JSON handling.

### 1. ****Create and Extract Archives with**** tar

bash

CopyEdit

# Create directory and files for testing

mkdir my\_folder

echo "Hello, world!" > my\_folder/hello.txt

echo "Backup Data" > my\_folder/data.txt

# Create an uncompressed .tar archive

tar -cf my\_folder.tar my\_folder/

# Create a compressed .tar.gz archive

tar -czf my\_folder.tar.gz my\_folder/

# Create a compressed .tar.bz2 archive

tar -cjf my\_folder.tar.bz2 my\_folder/

# Create a compressed .tar.xz archive

tar -cJf my\_folder.tar.xz my\_folder/

# Extract files from the .tar archive

tar -xf my\_folder.tar

# Extract files from the .tar.gz archive

tar -xzf my\_folder.tar.gz

# Extract files from the .tar.bz2 archive

tar -xjf my\_folder.tar.bz2

# Extract files from the .tar.xz archive

tar -xJf my\_folder.tar.xz

# List contents of the .tar archive

tar -tvf my\_folder.tar

# Exclude .log files from the archive

tar -cf my\_folder.tar --exclude='\*.log' my\_folder/

### 2. ****Compress Files with**** gzip

bash

CopyEdit

# Compress a single file using gzip

gzip hello.txt

# Compress a file without deleting the original

gzip -k data.txt

# Decompress the file

gunzip hello.txt.gz

# List gzip file info

gzip -l hello.txt.gz

### 3. ****Compress Files with**** bzip2

bash

CopyEdit

# Compress a file with bzip2

bzip2 hello.txt

# Decompress the file

bunzip2 hello.txt.bz2

# List compression stats

bzip2 -l hello.txt.bz2

### 4. ****Compress Files with**** xz

bash

CopyEdit

# Compress a file with xz

xz hello.txt

# Decompress the file

xz -d hello.txt.xz

# List xz file info

xz -l hello.txt.xz

### 5. ****Create and Extract Archives with**** 7z

bash

CopyEdit

# Create a .7z archive

7z a my\_archive.7z my\_folder/

# Extract files from the .7z archive

7z x my\_archive.7z

# Test the integrity of a .7z archive

7z t my\_archive.7z

# List contents of the .7z archive

7z l my\_archive.7z

### 6. ****Create and Extract**** .zip ****Archives****

bash

CopyEdit

# Create a .zip archive

zip my\_folder.zip my\_folder/

# Recursively zip a directory

zip -r my\_folder.zip my\_folder/

# Extract files from the .zip archive

unzip my\_folder.zip

# List contents of the .zip archive

unzip -l my\_folder.zip

### 7. ****Create and Extract**** .rar ****Archives****

bash

CopyEdit

# Create a .rar archive

rar a my\_folder.rar my\_folder/

# Extract files from the .rar archive

unrar x my\_folder.rar

# List the contents of a .rar archive

unrar l my\_folder.rar

### 8. ****Compression with**** zstd

bash

CopyEdit

# Compress a file with zstd

zstd hello.txt

# Decompress the file

zstd -d hello.txt.zst

# List zstd file info

zstd -l hello.txt.zst

### 9. ****Backup with**** tar

bash

CopyEdit

# Backup a directory with a timestamp

tar -czf backup\_$(date +%F).tar.gz my\_folder/

### 10. ****File Synchronization with**** rsync

bash

CopyEdit

# Synchronize files and compress during transfer

rsync -avz my\_folder/ remote\_server:/backup/

# Synchronize and delete files that no longer exist

rsync -avz --delete my\_folder/ remote\_server:/backup/

### 11. ****Use**** logrotate ****for Log Management****

bash

CopyEdit

# Rotate logs based on configuration

logrotate /etc/logrotate.conf

# Force log rotation

logrotate -f /etc/logrotate.conf

### 12. ****Sort Files with**** sort

bash

CopyEdit

# Sort a CSV file by the second column

sort -t',' -k2,2 my\_file.csv

# Sort and remove duplicate lines

sort -u my\_file.txt

### 13. ****Use**** uniq ****to Remove Duplicates****

bash

CopyEdit

# Filter out consecutive duplicate lines

uniq my\_file.txt

# Show only unique lines

uniq -u my\_file.txt

### 14. ****Search with**** grep ****Using Extended Regex****

bash

CopyEdit

# Search using Perl-compatible regex

grep -P 'pattern' my\_file.txt

# Show lines before and after the match

grep -C 2 'pattern' my\_file.txt

### 15. ****Text Transformation with**** sed

bash

CopyEdit

# Replace 'hello' with 'world' in place

sed -i 's/hello/world/g' my\_file.txt

# Delete the second line

sed '2d' my\_file.txt

### 16. ****Process Data with**** awk

bash

CopyEdit

# Sum the second column

awk '{sum += $2} END {print sum}' my\_file.txt

# Print the first column

awk '{print $1}' my\_file.txt

### 17. ****Use**** find ****with**** exec ****for Processing****

bash

CopyEdit

# Find all .txt files and count lines

find . -name "\*.txt" -exec wc -l {} \;

# Replace "foo" with "bar" in all .txt files

find . -name "\*.txt" -exec sed -i 's/foo/bar/g' {} \;

### 18. ****Process Files with**** xargs

bash

CopyEdit

# Move files listed in a text file

echo "file1 file2 file3" | xargs -I {} mv {} /destination/

### 19. ****Use**** tee ****to Write and Display Output****

bash

CopyEdit

# Write output to a file and display it

echo "Hello World" | tee output.txt

# Pipe output through grep and display the result

echo "Hello World" | tee >(grep "Hello")

### 20. ****Replace Characters with**** tr

bash

CopyEdit

# Convert lowercase to uppercase

echo "hello" | tr 'a-z' 'A-Z'

# Remove digits

echo "foo 123 bar" | tr -d '[:digit:]'

### 21. ****Number Lines with**** nl

bash

CopyEdit

# Number lines in a file

nl my\_file.txt

### 22. ****Split Files with**** split

bash

CopyEdit

# Split a file into smaller files with 100 lines each

split -l 100 large\_file.txt

# Split a file into 5MB chunks

split -b 5M large\_file.txt

### 23. ****Format Output with**** column

bash

CopyEdit

# Format space-separated values into columns

echo -e "a b c\nd e f" | column -t

# Format comma-separated values into columns

cat file.csv | column -s ',' -t

### 24. ****JSON Processing with**** jq

bash

CopyEdit

# Pretty-print JSON data

cat file.json | jq .

# Extract specific key from JSON

cat file.json | jq '.key'

# Extract an array of names

cat file.json | jq '.users[] | .name'

### Summary

1. top Purpose: Displays real-time system summary and process information. • top – Display system information (CPU, memory, processes). • top -u <user> – Show processes for a specific user. • top -p <pid> – Show specific process by PID. • top -d <seconds> – Set the update delay interval (in seconds). • top -n <count> – Show the output for a specified number of iterations. • top -c – Show full command lines of processes. 2. htop Purpose: Interactive process viewer (more user-friendly version of top). • htop – Start the interactive viewer. • htop -u <user> – Show processes for a specific user. • htop -s <field> – Sort processes by specific fields (e.g., CPU, memory). • htop -d <delay> – Adjust the delay between updates. 3. ps Purpose: Display information about active processes. • ps aux – Show all processes running on the system. • ps -ef – Show all processes with full command lines. • ps -u <user> – Show processes for a specific user. • ps -p <pid> – Show a specific process by PID. • ps -eo pid,comm,%cpu,%mem – Show specific fields (PID, command, CPU, memory). 4. vmstat Purpose: Report virtual memory statistics. • vmstat 1 – Report every second. • vmstat -s – Show memory statistics in a human-readable format. • vmstat -d – Report disk statistics. 5. free Purpose: Display memory usage. • free – Show memory usage (RAM and swap). • free -h – Show human-readable memory values (e.g., MB, GB). • free -m – Display memory in megabytes. • free -g – Display memory in gigabytes. 6. iostat Purpose: Display CPU and input/output statistics for devices. • iostat – Show CPU and I/O statistics. • iostat -d – Show device statistics. • iostat -c – Show CPU statistics. • iostat -x – Show extended device statistics. 7. sar Purpose: Collect and report system activity information. • sar -u 1 3 – Show CPU usage every 1 second, 3 times. • sar -r – Show memory usage. • sar -d – Show disk activity. • sar -n DEV – Show network activity. 8. dstat Purpose: Versatile resource statistics tool. • dstat – Show CPU, disk, network, and I/O stats. • dstat -c – Show only CPU stats. • dstat -d – Show only disk stats. • dstat -n – Show only network stats. • dstat -t – Show time stats. 9. netstat Purpose: Show network connections, routing tables, interface statistics. • netstat -tuln – Show active listening ports. • netstat -tulpn – Show active connections with the associated program. • netstat -i – Show network interface statistics. • netstat -rn – Show routing tables. • netstat -s – Show network statistics. 10. ss Purpose: Utility to investigate sockets. • ss – Show all sockets. • ss -tuln – Show listening sockets. • ss -p – Show process using the socket. • ss -s – Show summary of socket statistics. 11. uptime Purpose: Show how long the system has been running. • uptime – Displays the uptime, number of users, and load averages. 12. lsof Purpose: List open files and processes using them. • lsof – Show all open files. • lsof -i – Show all network connections. • lsof -u <user> – Show files opened by a specific user. • lsof +D <directory> – Show files opened in a specific directory. 13. df Purpose: Report file system disk space usage. • df – Show disk space usage for mounted filesystems. • df -h – Human-readable format (e.g., MB, GB). • df -T – Show file system types. • df -i – Show inode usage. 14. du Purpose: Estimate file space usage. • du – Show the disk usage of files and directories. • du -sh <directory> – Show total disk usage of a directory in human-readable format. • du -ah – Show disk usage of all files and directories. 15. watch Purpose: Execute a program periodically and show output. • watch <command> – Run a command every 2 seconds by default. • watch -n <seconds> <command> – Run the command at custom intervals. • watch -d <command> – Highlight changes in the output. 16. lscpu Purpose: Display information about the CPU architecture. • lscpu – Show detailed CPU architecture information. 17. lsblk Purpose: List information about block devices (disks). • lsblk – List all block devices. • lsblk -f – Show file system information. • lsblk -o NAME,SIZE,TYPE,MOUNTPOINT – Custom output columns. 18. systemctl Purpose: Control the systemd system and service manager. • systemctl status – Show system and service manager status. • systemctl status <service> – Show specific service status. • systemctl list-units --type=service – List all active services. • systemctl start <service> – Start a service. • systemctl stop <service> – Stop a service. 19. journalctl Purpose: Query and display logs from systemd journal. • journalctl – Show all system logs. • journalctl -u <service> – Show logs for a specific service. • journalctl -f – Follow the log output. • journalctl --since "2025-04-05" – Show logs since a specific date. 20. atop Purpose: Advanced system and process monitoring tool. • atop – Start the tool to monitor system and process activities. • atop -r <file> – Read a log file recorded by atop. • atop -d <seconds> – Update every x seconds. 21. sar (Sysstat) Purpose: Collect and report system activity information. • sar -u 1 3 – Report CPU utilization every second, three times. • sar -r – Show memory statistics. • sar -d – Show disk statistics. 22. tcpdump Purpose: Network traffic capture. • tcpdump – Capture network traffic. • tcpdump -i <interface> – Capture traffic from a specific network interface. • tcpdump -n – Don't resolve hostnames. • tcpdump -v – Increase verbosity. 23. iftop Purpose: Display bandwidth usage on an interface. • iftop – Show bandwidth usage on each connection. • iftop -ni <interface> – Show bandwidth on a specific interface. 24. uptime Purpose: Show system uptime and load averages. • uptime – Displays the system uptime and load averages. 25. mpstat Purpose: Report CPU statistics. • mpstat – Show CPU stats for each processor. • mpstat -P ALL – Show stats for all processors. 26. strace Purpose: Trace system calls and signals. • strace -p <pid> – Attach to a running process by its PID and trace system calls. • strace -e trace=<syscall> – Trace a specific system call (e.g., read, write). • strace -o <file> – Save the output to a file. • strace -f – Trace child processes (forks). • strace -c – Count time, calls, and errors for each system call. 27. dmesg Purpose: Print kernel ring buffer messages, typically for debugging hardware issues. • dmesg – Display kernel messages. • dmesg | grep <pattern> – Filter messages based on a pattern. • dmesg -T – Show human-readable timestamps. • dmesg --clear – Clear the kernel ring buffer. 28. sysctl Purpose: Interface to change kernel parameters at runtime. • sysctl -a – Display all available kernel parameters. • sysctl -w <parameter>=<value> – Set a specific kernel parameter. • sysctl net.ipv4.ip\_forward=1 – Enable IP forwarding (e.g., for routing). • sysctl -p – Load settings from /etc/sysctl.conf or a configuration file. 29. pidstat Purpose: Collect and display statistics by process. • pidstat – Show CPU, memory, and other statistics for processes. • pidstat -u – Display CPU usage by process. • pidstat -r – Show memory usage by process. • pidstat -d – Show I/O statistics for processes. • pidstat -p <pid> – Display stats for a specific process. 30. netcat (nc) Purpose: Network utility for reading from and writing to network connections. • nc -l <port> – Start listening on a specific port. • nc <host> <port> – Connect to a remote host and port. • nc -zv <host> <port-range> – Scan a range of ports on a remote host. • nc -w 5 <host> <port> – Set a timeout of 5 seconds for a connection. 31. curl Purpose: Transfer data from or to a server. • curl -I <URL> – Show only the HTTP header information. • curl -O <URL> – Download a file. • curl -L <URL> – Follow redirects. • curl -u <user>:<password> <URL> – Make an authenticated request. • curl -X POST <URL> -d "<data>" – Make a POST request with data. • curl --silent --show-error <URL> – Suppress output except for errors. 32. ncdu Purpose: Disk usage analyzer with a user-friendly interface. • ncdu – Display disk usage and allow interactive navigation. • ncdu <path> – Analyze disk usage for a specific path or directory. 33. tshark Purpose: Network protocol analyzer (similar to tcpdump, but with more features). • tshark – Start capturing packets. • tshark -i <interface> – Capture packets from a specific interface. • tshark -f "<filter>" – Capture only specific packets based on a filter (e.g., tcp port 80). • tshark -w <file> – Save captured packets to a file. 34. tcpdump Purpose: Capture network traffic and troubleshoot networking issues. • tcpdump -i <interface> – Capture packets on a specified network interface. • tcpdump -s 0 -A – Capture and display full packet contents (useful for HTTP traffic). • tcpdump -n – Don't resolve hostnames (show IP addresses). • tcpdump -v – Verbose output (more detailed packet info). • tcpdump -X – Show packet contents in hexadecimal and ASCII. • tcpdump 'port 80' – Capture HTTP traffic. 35. bpfcc / bpftrace Purpose: Tools for tracing and monitoring Linux kernel events with eBPF (Extended Berkeley Packet Filter). • bpfcc – Use BPF tools to trace and debug system performance. • bpftrace – Dynamic tracing tool that allows custom tracing scripts. • bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { @[comm] = count(); }' – Trace the open() system call. • bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }' – Trace file reads. 36. inotifywait Purpose: Monitor file system events using inotify. • inotifywait -m <directory> – Monitor events in a directory. • inotifywait -r – Monitor directories recursively. • inotifywait -e <event> <file> – Monitor a specific event (e.g., create, delete). 37. monitorix Purpose: A system and network monitoring tool designed to collect and display various system parameters. • monitorix – Launch the web-based monitoring interface. • It provides detailed visual reports on system metrics, such as CPU, memory, disk, network, and many other metrics. - create exampleas from scratch and apply all thsese comamnds show examples for all these commands

###### ChatGPT said:

Here’s a comprehensive set of examples for each of the commands you mentioned, demonstrating their usage in a Linux environment. These examples will cover system monitoring, file management, network utilities, and performance tools.

### 1. top

bash

CopyEdit

# Display system information

top

# Show processes for a specific user

top -u user1

# Show specific process by PID

top -p 1234

# Set the update delay interval to 5 seconds

top -d 5

# Show output for a specified number of iterations (5 iterations)

top -n 5

# Show full command lines of processes

top -c

### 2. htop

bash

CopyEdit

# Start the interactive viewer

htop

# Show processes for a specific user

htop -u user1

# Sort processes by CPU usage

htop -s PERCENT\_CPU

# Adjust delay between updates to 1 second

htop -d 1

### 3. ps

bash

CopyEdit

# Show all processes running on the system

ps aux

# Show all processes with full command lines

ps -ef

# Show processes for a specific user

ps -u user1

# Show specific process by PID

ps -p 1234

# Show PID, command, CPU, and memory usage

ps -eo pid,comm,%cpu,%mem

### 4. vmstat

bash

CopyEdit

# Report every second

vmstat 1

# Show memory statistics in a human-readable format

vmstat -s

# Report disk statistics

vmstat -d

### 5. free

bash

CopyEdit

# Show memory usage

free

# Show memory usage in human-readable format (e.g., MB, GB)

free -h

# Display memory in megabytes

free -m

# Display memory in gigabytes

free -g

### 6. iostat

bash

CopyEdit

# Show CPU and I/O statistics

iostat

# Show device statistics

iostat -d

# Show CPU statistics

iostat -c

# Show extended device statistics

iostat -x

### 7. sar

bash

CopyEdit

# Show CPU usage every 1 second, 3 times

sar -u 1 3

# Show memory usage

sar -r

# Show disk activity

sar -d

# Show network activity

sar -n DEV

### 8. dstat

bash

CopyEdit

# Show CPU, disk, network, and I/O stats

dstat

# Show only CPU stats

dstat -c

# Show only disk stats

dstat -d

# Show only network stats

dstat -n

# Show time stats

dstat -t

### 9. netstat

bash

CopyEdit

# Show active listening ports

netstat -tuln

# Show active connections with the associated program

netstat -tulpn

# Show network interface statistics

netstat -i

# Show routing tables

netstat -rn

# Show network statistics

netstat -s

### 10. ss

bash

CopyEdit

# Show all sockets

ss

# Show listening sockets

ss -tuln

# Show process using the socket

ss -p

# Show summary of socket statistics

ss -s

### 11. uptime

bash

CopyEdit

# Display system uptime, number of users, and load averages

uptime

### 12. lsof

bash

CopyEdit

# Show all open files

lsof

# Show all network connections

lsof -i

# Show files opened by a specific user

lsof -u user1

# Show files opened in a specific directory

lsof +D /home/user1/

### 13. df

bash

CopyEdit

# Show disk space usage for mounted filesystems

df

# Show disk space in human-readable format

df -h

# Show file system types

df -T

# Show inode usage

df -i

### 14. du

bash

CopyEdit

# Show the disk usage of files and directories

du

# Show total disk usage of a directory in human-readable format

du -sh /home/user1/

# Show disk usage of all files and directories

du -ah

### 15. watch

bash

CopyEdit

# Run a command every 2 seconds by default

watch df

# Run the command at custom intervals (5 seconds)

watch -n 5 df

# Highlight changes in the output

watch -d df

### 16. lscpu

bash

CopyEdit

# Show detailed CPU architecture information

lscpu

### 17. lsblk

bash

CopyEdit

# List all block devices

lsblk

# Show file system information

lsblk -f

# Custom output columns: NAME, SIZE, TYPE, MOUNTPOINT

lsblk -o NAME,SIZE,TYPE,MOUNTPOINT

### 18. systemctl

bash

CopyEdit

# Show system and service manager status

systemctl status

# Show specific service status

systemctl status apache2

# List all active services

systemctl list-units --type=service

# Start a service

systemctl start apache2

# Stop a service

systemctl stop apache2

### 19. journalctl

bash

CopyEdit

# Show all system logs

journalctl

# Show logs for a specific service

journalctl -u apache2

# Follow the log output

journalctl -f

# Show logs since a specific date

journalctl --since "2025-04-05"

### 20. atop

bash

CopyEdit

# Start the tool to monitor system and process activities

atop

# Read a log file recorded by atop

atop -r /var/log/atop/atop\_2025-04-05\_00:00:00

# Update every 5 seconds

atop -d 5

### 21. sar ****(Sysstat)****

bash

CopyEdit

# Report CPU utilization every second, three times

sar -u 1 3

# Show memory statistics

sar -r

# Show disk statistics

sar -d

### 22. tcpdump

bash

CopyEdit

# Capture network traffic

tcpdump

# Capture traffic from a specific network interface

tcpdump -i eth0

# Don't resolve hostnames

tcpdump -n

# Increase verbosity

tcpdump -v

# Show packet contents in hexadecimal and ASCII

tcpdump -X

# Capture HTTP traffic

tcpdump 'port 80'

### 23. iftop

bash

CopyEdit

# Show bandwidth usage on each connection

iftop

# Show bandwidth on a specific interface (e.g., eth0)

iftop -ni eth0

### 24. uptime (Repeated command)

bash

CopyEdit

# Show system uptime and load averages

uptime

### 25. mpstat

bash

CopyEdit

# Show CPU stats for each processor

mpstat

# Show stats for all processors

mpstat -P ALL

### 26. strace

bash

CopyEdit

# Attach to a running process by PID and trace system calls

strace -p 1234

# Trace a specific system call (e.g., read)

strace -e trace=read -p 1234

# Save the output to a file

strace -o output.txt -p 1234

# Trace child processes (forks)

strace -f -p 1234

# Count time, calls, and errors for each system call

strace -c -p 1234

### 27. dmesg

bash

CopyEdit

# Display kernel messages

dmesg

# Filter messages based on a pattern

dmesg | grep error

# Show human-readable timestamps

dmesg -T

# Clear the kernel ring buffer

dmesg --clear

### 28. sysctl

bash

CopyEdit

# Display all available kernel parameters

sysctl -a

# Set a specific kernel parameter

sysctl -w net.ipv4.ip\_forward=1

# Load settings from /etc/sysctl.conf or a configuration file

sysctl -p

### 29. pidstat

bash

CopyEdit

# Show CPU, memory, and other statistics for processes

pidstat

# Display CPU usage by process

pidstat -u

# Show memory usage by process

pidstat -r

# Show I/O statistics for processes

pidstat -d

# Display stats for a specific process (PID 1234)

pidstat -p 1234

### 30. netcat (nc)

bash

CopyEdit

# Start listening on a specific port

nc -l 8080

# Connect to a remote host and port

nc example.com 80

# Scan a range of ports on a remote host

nc -zv example.com 80-90

# Set a timeout of 5 seconds for a connection

nc -w 5 example.com 80

### 31. curl

bash

CopyEdit

# Show only the HTTP header information

curl -I http://example.com

# Download a file

curl -O http://example.com/file.txt

# Follow redirects

curl -L http://example.com

# Make an authenticated request

curl -u user:password http://example.com

# Make a POST request with data

curl -X POST http://example.com -d "param1=value1&param2=value2"

# Suppress output except for errors

curl --silent --show-error http://example.com

### 32. ncdu

bash

CopyEdit

# Display disk usage and allow interactive navigation

ncdu

# Analyze disk usage for a specific path or directory

ncdu /home/user1/

### 33. tshark

bash

CopyEdit

# Start capturing packets

tshark

# Capture packets from a specific interface

tshark -i eth0

# Capture only specific packets based on a filter (e.g., TCP port 80)

tshark -f "tcp port 80"

# Save captured packets to a file

tshark -w capture.pcap

### 34. tcpdump (Repeated command)

bash

CopyEdit

# Capture packets on a specified network interface

tcpdump -i eth0

# Capture and display full packet contents (useful for HTTP traffic)

tcpdump -s 0 -A

# Show packet contents in hexadecimal and ASCII

tcpdump -X

# Capture HTTP traffic

tcpdump 'port 80'

### 35. bpfcc / bpftrace

bash

CopyEdit

# Use BPF tools to trace and debug system performance

bpfcc

# Trace the open() system call

bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { @[comm] = count(); }'

# Trace file reads

bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }'

### 36. inotifywait

bash

CopyEdit

# Monitor events in a directory

inotifywait -m /path/to/directory

# Monitor directories recursively

inotifywait -r /path/to/directory

# Monitor a specific event (e.g., file creation)

inotifywait -e create /path/to/file

### 37. monitorix

bash

CopyEdit

# Launch the web-based monitoring interface

monitorix

1. ifconfig Purpose: Display and configure network interfaces. • ifconfig – Show all network interfaces and their IP addresses. • ifconfig <interface> – Show specific interface (e.g., eth0, wlan0). • ifconfig <interface> up – Bring up a network interface. • ifconfig <interface> down – Bring down a network interface. • ifconfig <interface> <IP> – Assign an IP address to an interface. • ifconfig <interface> netmask <netmask> – Set the netmask for the interface. 2. ip Purpose: A more modern and powerful alternative to ifconfig for network management. • ip a – Show all network interfaces and their IP addresses. • ip link show – Display link information (e.g., interface status). • ip addr show – Show IP addresses assigned to all interfaces. • ip link set <interface> up – Bring the interface up. • ip link set <interface> down – Bring the interface down. • ip addr add <IP>/24 dev <interface> – Assign a new IP to an interface. • ip route show – Display the routing table. • ip route add <destination> via <gateway> – Add a new route. 3. ping Purpose: Send ICMP echo requests to check network connectivity. • ping <hostname or IP> – Ping a host by its IP address or hostname. • ping -c <count> <hostname> – Send a specific number of packets. • ping -i <interval> <hostname> – Set the interval between packets. • ping -t <ttl> <hostname> – Set the TTL (Time To Live) for packets. • ping -s <size> <hostname> – Send packets of a specific size. • ping -4 <hostname> – Force the use of IPv4. • ping -6 <hostname> – Force the use of IPv6. 4. traceroute Purpose: Trace the route packets take to a destination. • traceroute <hostname> – Trace the route to a remote host. • traceroute -m <max\_hops> <hostname> – Set the maximum number of hops. • traceroute -T – Use TCP packets for tracing. • traceroute -I – Use ICMP packets for tracing. 5. nslookup Purpose: Query DNS to obtain domain name or IP address information. • nslookup <hostname> – Query the IP address of a given hostname. • nslookup <IP> – Query the domain name for a given IP. • nslookup -type=MX <domain> – Query MX (mail) records for a domain. • nslookup -type=NS <domain> – Query the name servers for a domain. 6. dig Purpose: Perform DNS lookups and query DNS servers. • dig <hostname> – Perform a basic DNS lookup for a domain. • dig @<DNS server> <hostname> – Query a specific DNS server. • dig <hostname> A – Query A (address) record for a domain. • dig <hostname> MX – Query MX (mail) record for a domain. • dig +short <hostname> – Shorten output to show only the result. 7. route Purpose: View and configure network routing tables. • route – Show the current routing table. • route -n – Show the routing table with numeric addresses (no DNS resolution). • route add <destination> gw <gateway> – Add a new route to a destination. • route del <destination> – Delete a specific route. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Advanced Networking Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 8. netstat Purpose: Display network connections, routing tables, interface statistics, etc. • netstat – Show active connections and listening ports. • netstat -tuln – Display TCP and UDP ports that are open and listening. • netstat -an – Display all network connections (IP and port). • netstat -s – Show network statistics. • netstat -i – Show network interfaces and packet statistics. • netstat -r – Show the routing table. 9. ss Purpose: Utility to investigate sockets. • ss – Display all sockets. • ss -tuln – Display listening TCP and UDP sockets. • ss -a – Show all sockets (listening and non-listening). • ss -p – Display processes using the sockets. • ss -r – Display routing information. 10. netcat (nc) Purpose: A versatile tool for network debugging and banner grabbing. • nc -zv <hostname> <port> – Scan a specific port or range of ports. • nc -l <port> – Listen on a specific port for incoming connections. • nc <hostname> <port> – Connect to a host on a specific port. • nc -u <hostname> <port> – Use UDP instead of TCP. • nc -v <hostname> <port> – Verbose output for connection details. 11. tcpdump Purpose: Capture network traffic and display the packets. • tcpdump – Capture packets on the default interface. • tcpdump -i <interface> – Capture on a specific network interface. • tcpdump -n – Do not resolve hostnames or port names. • tcpdump -v – Show more verbose packet information. • tcpdump -c <count> – Capture a specific number of packets. • tcpdump -w <file> – Save the captured packets to a file. • tcpdump 'port 80' – Capture HTTP traffic. • tcpdump -A – Display packet contents in ASCII format. • tcpdump -X – Display packet contents in both hex and ASCII. 12. iftop Purpose: Display bandwidth usage on a network interface in real-time. • iftop – Display real-time bandwidth usage. • iftop -i <interface> – Specify the network interface to monitor. • iftop -n – Show numerical addresses (do not resolve hostnames). • iftop -P – Show port numbers along with IP addresses. 13. nmap Purpose: Network exploration and security auditing tool. • nmap <hostname> – Perform a simple scan of a host. • nmap -sP <IP-range> – Perform a ping scan to identify live hosts. • nmap -p <ports> <hostname> – Scan specific ports (e.g., -p 80,443). • nmap -sS <hostname> – Perform a SYN scan (stealth scan). • nmap -O <hostname> – OS detection. • nmap -A <hostname> – Enable OS detection, version detection, script scanning, and traceroute. 14. mtr Purpose: A network diagnostic tool combining traceroute and ping. • mtr <hostname> – Continuous traceroute and ping to a destination. • mtr -r <hostname> – Generate a report instead of continuous output. • mtr -c <count> <hostname> – Set the number of pings to send. 15. arp Purpose: View and manipulate the system's ARP (Address Resolution Protocol) cache. • arp -a – Display the current ARP table. • arp -d <IP> – Delete an entry from the ARP table. • arp -s <IP> <MAC> – Manually add a static entry to the ARP table. 16. ethtool Purpose: Query and control network device settings. • ethtool <interface> – Display information about a network interface. • ethtool -s <interface> speed <speed> duplex <duplex> – Change the speed and duplex settings of a network interface. • ethtool -p <interface> – Blink the LED on a network interface to help locate it. 17. iwconfig Purpose: Configure wireless network interfaces. • iwconfig – Display wireless network information. • iwconfig <interface> essid <SSID> – Connect to a specific Wi-Fi network. • iwconfig <interface> mode <mode> – Set the mode of the wireless interface (e.g., managed, ad-hoc). 18. ip link Purpose: Display or modify network device attributes. • ip link show – Display detailed information about network interfaces. • ip link set <interface> up/down – Bring the interface up or down. • ip link set <interface> mtu <mtu\_size> – Change the MTU size for an interface. 19. sshd (SSH Daemon) Purpose: Manage SSH access and configurations. • sshd -T – Show current configuration options. • sshd -f <config\_file> – Use a specific SSH configuration file. • sshd -D – Run in the foreground (useful for debugging). \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Cloud/Container Specific Networking Commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 20. docker network Purpose: Manage Docker network settings. • docker network ls – List all Docker networks. • docker network inspect <network\_name> – Inspect a specific Docker network. • docker network create <network\_name> – Create a new Docker network. • docker network connect <network\_name> <container\_name> – Connect a container to a network. 21. kubectl get svc Purpose: Retrieve the details of Kubernetes services. • kubectl get svc – List all services in the current namespace. • kubectl get svc <service\_name> – Get details of a specific service. • kubectl describe svc <service\_name> – Detailed information on the service, including networking details. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ These commands, along with their options and flags, are crucial for diagnosing, configuring, and optimizing networking in any system or infrastructure. DevOps Engineers often use them to ensure reliable network performance, security, and troubleshooting in both on-premise and cloud-based environments. You said: please continue with more advance linux networking cmds used by devops engineer ChatGPT said: Here’s a continuation of advanced Linux networking commands that are typically used by DevOps Engineers for deep network diagnostics, configuration, performance monitoring, and troubleshooting. These commands are important when working with complex network setups, performance tuning, and cloud or containerized environments. \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 22. ss (Socket Stat) Purpose: A more modern and faster alternative to netstat for analyzing socket connections. • ss -tuln – Display all listening TCP and UDP sockets. • ss -an – Show all network connections in numeric form (no DNS resolution). • ss -p – Show the processes associated with each socket. • ss -s – Display socket statistics (summary). • ss -o state established – Show only established connections. • ss -r – Display routing information along with socket statistics. • ss -A – Show application layer statistics. 23. nft (Netfilter) Purpose: Interface for configuring packet filtering and network address translation (NAT). • nft list ruleset – Display all the rules in the firewall. • nft add rule inet filter input tcp dport 22 accept – Add a rule to allow incoming SSH connections. • nft flush ruleset – Flush all rules from the ruleset. • nft delete rule inet filter input handle 4 – Delete a specific rule by its handle number. • nft list table inet filter – List the rules in the filter table. • nft add chain inet filter input { type filter hook input priority 0 \; } – Add a chain to filter incoming packets. 24. iptables Purpose: Configure packet filtering rules in the Linux kernel firewall. • iptables -L – List all active firewall rules. • iptables -A INPUT -p tcp --dport 80 -j ACCEPT – Allow incoming HTTP traffic. • iptables -A INPUT -p tcp --dport 443 -j ACCEPT – Allow incoming HTTPS traffic. • iptables -A INPUT -p tcp --dport 22 -s <IP> -j ACCEPT – Allow SSH traffic from a specific IP. • iptables -D INPUT -p tcp --dport 80 -j ACCEPT – Remove the rule allowing HTTP traffic. • iptables -F – Flush all rules in the filter table. • iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.2:8080 – Redirect HTTP traffic to a different destination. 25. firewalld Purpose: Manage firewall rules dynamically with firewalld. • firewall-cmd --list-all – Show the current firewall settings and rules. • firewall-cmd --zone=public --add-port=8080/tcp --permanent – Open port 8080 permanently. • firewall-cmd --reload – Reload the firewall to apply new rules. • firewall-cmd --zone=public --remove-port=8080/tcp --permanent – Close port 8080. • firewall-cmd --list-ports – Show open ports. 26. bpftrace Purpose: Dynamic tracing tool for observing and troubleshooting kernel-level events using BPF (Berkeley Packet Filter). • bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { printf("%s opened\n", str(args->filename)); }' – Trace all open system calls. • bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }' – Trace file reads. • bpftrace -e 'tracepoint:net:net\_dev\_xmit { @[args->dev\_name] = count(); }' – Count network transmissions for each device. 27. tcpdump with Advanced Filtering Purpose: Advanced packet capture with specific filters. • tcpdump -i eth0 port 443 – Capture packets on port 443 (HTTPS). • tcpdump -i eth0 'tcp[tcpflags] & tcp-push != 0' – Capture TCP packets with the PUSH flag set. • tcpdump -i eth0 'src host <IP>' – Capture packets from a specific source IP. • tcpdump -i eth0 'dst host <IP>' – Capture packets destined to a specific IP. • tcpdump -i eth0 -w capture.pcap – Write captured packets to a file. • tcpdump -r capture.pcap – Read packets from a capture file. • tcpdump -i eth0 -v – Increase verbosity of the output. 28. ip rule Purpose: Used to configure policy-based routing. • ip rule show – Display the current routing rules. • ip rule add from <IP> table <table\_id> – Add a rule to route traffic from a specific IP through a custom routing table. • ip rule delete from <IP> – Delete the rule for traffic from a specific IP. • ip rule add iif <interface> table <table\_id> – Route traffic coming from a specific interface through a particular table. 29. tc (Traffic Control) Purpose: Control network traffic on interfaces. • tc qdisc show – Show the current queuing disciplines (QDisc) for interfaces. • tc qdisc add dev eth0 root netem delay 100ms – Add network delay to interface eth0. • tc qdisc change dev eth0 root netem loss 5% – Simulate 5% packet loss on eth0. • tc class add dev eth0 parent 1:1 classid 1:10 htb rate 1mbit – Add a traffic class with a rate limit. • tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 80 0xffff flowid 1:10 – Filter HTTP traffic and assign it to a specific traffic class. 30. ethtool (Advanced) Purpose: Query and modify network interface driver and hardware settings. • ethtool eth0 – Display settings for the eth0 network interface. • ethtool -s eth0 speed 1000 duplex full – Set the speed and duplex mode for the interface. • ethtool -p eth0 – Blink the LED to locate the physical interface. • ethtool -i eth0 – Display driver information for the network interface. • ethtool -S eth0 – Show statistics about the network interface. 31. nmcli Purpose: Command-line interface for NetworkManager (useful for managing network connections). • nmcli connection show – List all active network connections. • nmcli device status – Show the status of all network devices. • nmcli connection add type ethernet con-name eth0 ifname eth0 – Add a new Ethernet connection. • nmcli device disconnect eth0 – Disconnect a specific network interface. • nmcli device wifi list – List available Wi-Fi networks. 32. conntrack Purpose: Used to view and manage the connection tracking subsystem in the Linux kernel. • conntrack -L – List all tracked connections. • conntrack -F – Flush all tracked connections. • conntrack -D – Delete a specific connection from the connection table. • conntrack -L -p tcp – List all tracked TCP connections. • conntrack -E – Display connection events in real-time. 33. wireshark Purpose: A GUI-based network protocol analyzer (often used in tandem with tcpdump for deep packet inspection). • Launch Wireshark and capture packets from a specific interface. • wireshark – Start Wireshark GUI. • wireshark -i <interface> – Start Wireshark and capture packets from a specific interface. • Wireshark supports advanced filtering using display filters (e.g., http, ip.addr==192.168.1.1). 34. bridge Purpose: Configure and manage network bridges. • bridge link – Show the status of all network bridges. • bridge fdb show – Show the Forwarding Database (FDB) for a bridge. • bridge vlan – Show or configure VLANs on a bridge. • bridge addbr br0 – Create a new network bridge. • bridge addif br0 eth0 – Add an interface to the bridge. 35. bmon Purpose: Bandwidth monitor and rate estimator for Linux. • bmon – Display bandwidth usage in real-time. • bmon -p – Display statistics for all interfaces. • bmon -c – Show detailed statistics in a continuous mode. - create exampleas from scratch and apply all thsese comamnds show examples for all these commands \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

###### ChatGPT said:

Sure! Below is a set of example commands for each of the networking tools you’ve mentioned, illustrating their usage for managing and troubleshooting networking configurations in a Linux environment.

### 1. ifconfig

bash

CopyEdit

# Show all network interfaces and their IP addresses

ifconfig

# Show specific interface (e.g., eth0)

ifconfig eth0

# Bring up a network interface

ifconfig eth0 up

# Bring down a network interface

ifconfig eth0 down

# Assign an IP address to an interface

ifconfig eth0 192.168.1.100

# Set the netmask for the interface

ifconfig eth0 netmask 255.255.255.0

### 2. ip

bash

CopyEdit

# Show all network interfaces and their IP addresses

ip a

# Display link information (e.g., interface status)

ip link show

# Show IP addresses assigned to all interfaces

ip addr show

# Bring the interface up

ip link set eth0 up

# Bring the interface down

ip link set eth0 down

# Assign a new IP address to an interface

ip addr add 192.168.1.100/24 dev eth0

# Display the routing table

ip route show

# Add a new route

ip route add 192.168.1.0/24 via 192.168.1.1

### 3. ping

bash

CopyEdit

# Ping a host by its IP address or hostname

ping google.com

# Send a specific number of packets (e.g., 4 packets)

ping -c 4 google.com

# Set the interval between packets to 0.5 seconds

ping -i 0.5 google.com

# Set the TTL (Time To Live) for packets

ping -t 64 google.com

# Send packets of a specific size (e.g., 1000 bytes)

ping -s 1000 google.com

# Force the use of IPv4

ping -4 google.com

# Force the use of IPv6

ping -6 google.com

### 4. traceroute

bash

CopyEdit

# Trace the route to a remote host

traceroute google.com

# Set the maximum number of hops

traceroute -m 15 google.com

# Use TCP packets for tracing

traceroute -T google.com

# Use ICMP packets for tracing

traceroute -I google.com

### 5. nslookup

bash

CopyEdit

# Query the IP address of a given hostname

nslookup google.com

# Query the domain name for a given IP address

nslookup 8.8.8.8

# Query MX (mail) records for a domain

nslookup -type=MX google.com

# Query the name servers for a domain

nslookup -type=NS google.com

### 6. dig

bash

CopyEdit

# Perform a basic DNS lookup for a domain

dig google.com

# Query a specific DNS server

dig @8.8.8.8 google.com

# Query A (address) record for a domain

dig google.com A

# Query MX (mail) record for a domain

dig google.com MX

# Shorten the output to show only the result

dig +short google.com

### 7. route

bash

CopyEdit

# Show the current routing table

route

# Show the routing table with numeric addresses

route -n

# Add a new route to a destination

route add 192.168.1.0 gw 192.168.1.1

# Delete a specific route

route del 192.168.1.0

### 8. netstat

bash

CopyEdit

# Show active connections and listening ports

netstat

# Display TCP and UDP ports that are open and listening

netstat -tuln

# Display all network connections (IP and port)

netstat -an

# Show network statistics

netstat -s

# Show network interfaces and packet statistics

netstat -i

# Show the routing table

netstat -r

### 9. ss

bash

CopyEdit

# Display all sockets

ss

# Display listening TCP and UDP sockets

ss -tuln

# Show all sockets (listening and non-listening)

ss -a

# Display processes using the sockets

ss -p

# Display routing information along with socket statistics

ss -r

### 10. netcat (nc)

bash

CopyEdit

# Scan a specific port or range of ports

nc -zv google.com 80-90

# Listen on a specific port for incoming connections

nc -l 8080

# Connect to a host on a specific port

nc google.com 80

# Use UDP instead of TCP

nc -u google.com 53

# Verbose output for connection details

nc -v google.com 80

### 11. tcpdump

bash

CopyEdit

# Capture packets on the default interface

tcpdump

# Capture packets on a specific network interface

tcpdump -i eth0

# Do not resolve hostnames or port names

tcpdump -n

# Show more verbose packet information

tcpdump -v

# Capture a specific number of packets

tcpdump -c 10

# Save captured packets to a file

tcpdump -w capture.pcap

# Capture HTTP traffic

tcpdump 'port 80'

# Display packet contents in ASCII format

tcpdump -A

# Display packet contents in both hex and ASCII

tcpdump -X

### 12. iftop

bash

CopyEdit

# Display real-time bandwidth usage

iftop

# Specify the network interface to monitor

iftop -i eth0

# Show numerical addresses (do not resolve hostnames)

iftop -n

# Show port numbers along with IP addresses

iftop -P

### 13. nmap

bash

CopyEdit

# Perform a simple scan of a host

nmap google.com

# Perform a ping scan to identify live hosts in a range

nmap -sP 192.168.1.0/24

# Scan specific ports (e.g., HTTP and HTTPS)

nmap -p 80,443 google.com

# Perform a SYN scan (stealth scan)

nmap -sS google.com

# Enable OS detection

nmap -O google.com

# Enable OS detection, version detection, script scanning, and traceroute

nmap -A google.com

### 14. mtr

bash

CopyEdit

# Continuous traceroute and ping to a destination

mtr google.com

# Generate a report instead of continuous output

mtr -r google.com

# Set the number of pings to send

mtr -c 10 google.com

### 15. arp

bash

CopyEdit

# Display the current ARP table

arp -a

# Delete an entry from the ARP table

arp -d 192.168.1.100

# Manually add a static entry to the ARP table

arp -s 192.168.1.100 00:11:22:33:44:55

### 16. ethtool

bash

CopyEdit

# Display settings for the eth0 network interface

ethtool eth0

# Set the speed and duplex mode for the interface

ethtool -s eth0 speed 1000 duplex full

# Blink the LED to locate the physical interface

ethtool -p eth0

# Display driver information for the network interface

ethtool -i eth0

# Show statistics about the network interface

ethtool -S eth0

### 17. iwconfig

bash

CopyEdit

# Display wireless network information

iwconfig

# Connect to a specific Wi-Fi network

iwconfig wlan0 essid "MySSID"

# Set the mode of the wireless interface (e.g., managed, ad-hoc)

iwconfig wlan0 mode managed

### 18. ip link

bash

CopyEdit

# Display detailed information about network interfaces

ip link show

# Bring the interface up

ip link set eth0 up

# Bring the interface down

ip link set eth0 down

# Change the MTU size for an interface

ip link set eth0 mtu 1500

### 19. sshd ****(SSH Daemon)****

bash

CopyEdit

# Show current SSH configuration options

sshd -T

# Use a specific SSH configuration file

sshd -f /etc/ssh/sshd\_config

# Run SSH daemon in the foreground for debugging

sshd -D

### 20. docker network

bash

CopyEdit

# List all Docker networks

docker network ls

# Inspect a specific Docker network

docker network inspect bridge

# Create a new Docker network

docker network create my\_network

# Connect a container to a network

docker network connect my\_network my\_container

### 21. kubectl get svc

bash

CopyEdit

# List all services in the current namespace

kubectl get svc

# Get details of a specific service

kubectl get svc my\_service

# Detailed information on the service, including networking details

kubectl describe svc my\_service

### 22. ss ****(Socket Stat)****

bash

CopyEdit

# Display all listening TCP and UDP sockets

ss -tuln

# Show all network connections in numeric form

ss -an

# Display the processes associated with each socket

ss -p

# Display socket statistics (summary)

ss -s

### 23. nft ****(Netfilter)****

bash

CopyEdit

# Display all the rules in the firewall

nft list ruleset

# Add a rule to allow incoming SSH connections

nft add rule inet filter input tcp dport 22 accept

# Flush all rules from the ruleset

nft flush ruleset

# Add a chain to filter incoming packets

nft add chain inet filter input { type filter hook input priority 0 \; }

### 24. iptables

bash

CopyEdit

# List all active firewall rules

iptables -L

# Allow incoming HTTP traffic

iptables -A INPUT -p tcp --dport 80 -j ACCEPT

# Allow incoming HTTPS traffic

iptables -A INPUT -p tcp --dport 443 -j ACCEPT

# Allow SSH traffic from a specific IP

iptables -A INPUT -p tcp --dport 22 -s 192.168.1.100 -j ACCEPT

# Remove the rule allowing HTTP traffic

iptables -D INPUT -p tcp --dport 80 -j ACCEPT

# Flush all rules in the filter table

iptables -F

# Redirect HTTP traffic to a different destination

iptables -t nat -A PREROUTING -p tcp --dport 80 -j DNAT --to-destination 192.168.1.2:8080

### 25. firewalld

bash

CopyEdit

# Show the current firewall settings and rules

firewall-cmd --list-all

# Open port 8080 permanently

firewall-cmd --zone=public --add-port=8080/tcp --permanent

# Reload the firewall to apply new rules

firewall-cmd --reload

# Close port 8080

firewall-cmd --zone=public --remove-port=8080/tcp --permanent

# Show open ports

firewall-cmd --list-ports

### 26. bpftrace

bash

CopyEdit

# Trace all open system calls

bpftrace -e 'tracepoint:syscalls:sys\_enter\_open { printf("%s opened\n", str(args->filename)); }'

# Trace file reads

bpftrace -e 'kprobe:vfs\_read { printf("PID %d read %d bytes\n", pid, args->count); }'

# Count network transmissions for each device

bpftrace -e 'tracepoint:net:net\_dev\_xmit { @[args->dev\_name] = count(); }'

### 27. tcpdump ****with Advanced Filtering****

bash

CopyEdit

# Capture packets on port 443 (HTTPS)

tcpdump -i eth0 port 443

# Capture TCP packets with the PUSH flag set

tcpdump -i eth0 'tcp[tcpflags] & tcp-push != 0'

# Capture packets from a specific source IP

tcpdump -i eth0 'src host 192.168.1.100'

# Capture packets destined to a specific IP

tcpdump -i eth0 'dst host 192.168.1.100'

# Write captured packets to a file

tcpdump -i eth0 -w capture.pcap

# Read packets from a capture file

tcpdump -r capture.pcap

### 28. ip rule

bash

CopyEdit

# Display the current routing rules

ip rule show

# Add a rule to route traffic from a specific IP through a custom routing table

ip rule add from 192.168.1.100 table 100

# Delete the rule for traffic from a specific IP

ip rule delete from 192.168.1.100

# Route traffic coming from a specific interface through a particular table

ip rule add iif eth0 table 200

### 29. tc ****(Traffic Control)****

bash

CopyEdit

# Show the current queuing disciplines (QDisc) for interfaces

tc qdisc show

# Add network delay to interface eth0

tc qdisc add dev eth0 root netem delay 100ms

# Simulate 5% packet loss on eth0

tc qdisc change dev eth0 root netem loss 5%

# Add a traffic class with a rate limit

tc class add dev eth0 parent 1:1 classid 1:10 htb rate 1mbit

# Filter HTTP traffic and assign it to a specific traffic class

tc filter add dev eth0 protocol ip parent 1:0 prio 1 u32 match ip dport 80 0xffff flowid 1:10

### 30. ethtool ****(Advanced)****

bash

CopyEdit

# Display settings for the eth0 network interface

ethtool eth0

# Set the speed and duplex mode for the interface

ethtool -s eth0 speed 1000 duplex full

# Blink the LED to locate the physical interface

ethtool -p eth0

# Display driver information for the network interface

ethtool -i eth0

# Show statistics about the network interface

ethtool -S eth0

### 31. nmcli

bash

CopyEdit

# List all active network connections

nmcli connection show

# Show the status of all network devices

nmcli device status

# Add a new Ethernet connection

nmcli connection add type ethernet con-name eth0 ifname eth0

# Disconnect a specific network interface

nmcli device disconnect eth0

# List available Wi-Fi networks

nmcli device wifi list

### 32. conntrack

bash

CopyEdit

# List all tracked connections

conntrack -L

# Flush all tracked connections

conntrack -F

# Delete a specific connection from the connection table

conntrack -D

# List all tracked TCP connections

conntrack -L -p tcp

# Display connection events in real-time

conntrack -E

### 33. wireshark

bash

CopyEdit

# Start Wireshark GUI

wireshark

# Start Wireshark and capture packets from a specific interface

wireshark -i eth0

# Use display filters to analyze traffic (e.g., filter HTTP traffic)

wireshark -i eth0 "http"

### 34. bridge

bash

CopyEdit

# Show the status of all network bridges

bridge link

# Show the Forwarding Database (FDB) for a bridge

bridge fdb show

# Show or configure VLANs on a bridge

bridge vlan

# Create a new network bridge

bridge addbr br0

# Add an interface to the bridge

bridge addif br0 eth0

### 35. bmon

bash

CopyEdit

# Display bandwidth usage in real-time

bmon

# Display statistics for all interfaces

bmon -p

# Show detailed statistics in a continuous mode

bmon -c

This collection of commands covers network management, troubleshooting, and performance monitoring, which are essential for DevOps engineers to ensure network reliability, security, and efficiency.