# 1. df (Disk Free) Command in Linux

The df command is used to check disk space usage for file systems in Linux. It provides information on the total space, used space, available space, and usage percentage.

## Syntax

sh

df [OPTIONS] [FILESYSTEM]

- If no file system is specified, it shows the disk space usage of all mounted file systems.

## 1.1. Options and Examples

### 1. Display Disk Space in Human-Readable Format

sh

df -h

**Output:**

bash

```
Filesystem     Size  Used Avail Use% Mounted on
/dev/sda1      50G   15G   35G   30% /
tmpfs          2G    4M    2G    1% /dev/shm
/dev/sdb1      100G  45G   55G   45% /data
```

**Explanation:**

- -h: Displays sizes in human-readable format (GB, MB, etc.).

## 2. Show Disk Space in Bytes, Kilobytes, Megabytes, or Gigabytes

sh

```
df -B 1M
```

**Output:**

bash

```
Filesystem     1M-blocks  Used Available Use% Mounted on
/dev/sda1      51200     15360 35840     30% /
tmpfs          2048      4     2044      1% /dev/shm
/dev/sdb1      102400    45000  55000     45% /data
```

**Explanation:**

- -B 1M: Displays output in megabytes.
- You can replace 1M with 1G for gigabytes, 1K for kilobytes, etc.

## 3. Display File System Type

sh

```
df -T
```

**Output:**

bash

```
Filesystem     Type     Size  Used Avail Use% Mounted on
/dev/sda1      ext4     50G   15G   35G   30% /
tmpfs          tmpfs    2G    4M    2G    1% /dev/shm
```

/dev/sdb1  xfs  100G  45G  55G  45% /data

**Explanation:**

- -T: Displays the type of file system (ext4, xfs, tmpfs, etc.).

## 4. Exclude Specific File Systems

sh

df -x tmpfs

**Output:**

bash

Filesystem  Size  Used Avail Use% Mounted on
/dev/sda1  50G  15G  35G  30% /
/dev/sdb1  100G  45G  55G  45% /data

**Explanation:**

- -x tmpfs: Excludes tmpfs file systems from the output.

## 5. Show Disk Usage for a Specific File or Directory

sh

df -h /home/user

**Output:**

bash

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       50G   15G   35G   30% /
```

**Explanation:**

- Shows the disk usage of the partition where /home/user is located.

## 1.2. Common Interview Questions on df

1. What does the df command do in Linux?
2. How can you display disk usage in a human-readable format?
3. What option is used to display the file system type?
4. How do you show disk space in megabytes instead of default units?
5. How can you exclude a specific file system type from the df output?
6. What's the difference between df -h and df -B 1M?
7. How can you check the available disk space of a specific directory?
8. How would you check disk space usage on a remote server?
9. What are the differences between df and du commands?
10. How would you use df to troubleshoot disk space issues?

## 2. free (Memory Usage) Command in Linux

The free command is used to check system memory usage, including total, used, and available RAM and swap space.

## 2.1 Syntax

sh

free [OPTIONS]

If no option is provided, it shows memory usage in kilobytes.

## 2.2 Options and Examples

### 1. Display Memory Usage in Human-Readable Format

sh

```
free -h
```

**Output:**

vbnet

```
              total        used        free      shared  buff/cache   available
Mem:           16G        4G        10G       512M        2G        11G
Swap:          2G        1G        1G
```

**Explanation:**

- -h: Displays memory sizes in a human-readable format (GB, MB, etc.).
- Mem: row shows physical RAM usage.
- Swap: row shows swap space usage.
- available: Approximate amount of memory available for new processes.

### 2. Show Memory Usage in Megabytes

sh

```
free -m
```

**Output:**

yaml

```
              total        used        free      shared  buff/cache   available
Mem:          16000       4000       10000       500        2000       11000
```

Swap:  2000  1000  1000

**Explanation:**

- -m: Displays output in megabytes.

## 3. Show Memory Usage in Gigabytes

sh

free -g

**Output:**

vbnet

```
        total  used  free  shared  buff/cache  available
Mem:    16     4     10    0.5     2           11
Swap:   2      1     1
```

**Explanation:**

- -g: Displays output in gigabytes.

## 4. Show Memory Usage with Continuous Updates

sh

free -s 5

**Output (refreshes every 5 seconds):**

vbnet

```
          total       used       free     shared  buff/cache   available
Mem:          16G        4G       10G      512M        2G        11G
Swap:          2G        1G        1G
```

```
          total       used       free     shared  buff/cache   available
Mem:          16G        4G       10G      512M        2G        11G
Swap:          2G        1G        1G
```

**Explanation:**

- -s 5: Refreshes memory usage every 5 seconds.

### 5. Show Memory Usage Without Buffers and Cache

sh

free -w

**Output:**

vbnet

```
          total       used       free     shared    buffers     cache   available
Mem:          16G        4G       10G      512M        1G        1G        11G
Swap:          2G        1G        1G
```

**Explanation:**

- -w: Shows additional buffer and cache details.

## 2.3 Common Interview Questions on free

1. What is the purpose of the free command?
2. How do you display memory usage in a human-readable format?

3. What is the difference between used and available memory in free?
4. How can you continuously monitor memory usage using free?
5. What does the buffers/cache column indicate?
6. How do you check memory usage in gigabytes instead of kilobytes?
7. How do you display memory usage excluding cache and buffers?
8. How would you check swap usage using free?
9. What command would you use to track memory usage over time?
10. How does free differ from top when checking memory?

# 3. nproc (Number of Processing Units) Command in Linux

The nproc command is used to display the number of CPU cores available to the current process.

## 3.1 Syntax

sh

nproc [OPTIONS]

By default, it returns the number of available CPU cores.

## 3.2 Options and Examples

### 1. Display the Number of Available CPU Cores

sh

nproc

**Output:**

8

**Explanation:**

- The system has 8 processing cores available.

## 2. Display the Total Number of CPU Cores, Including Offline Ones

sh

nproc --all

**Output:**

16

**Explanation:**

- --all: Displays all CPU cores, even if some are restricted or offline.

## 3. Limit the Number of Cores for a Process

If you want to restrict a process to use fewer cores, use taskset:

sh

taskset -c 0-3 some_command

**Explanation:**

- Runs some_command using only CPUs 0, 1, 2, and 3.

## 3.3 Common Interview Questions on nproc

1. What does the nproc command do?
2. How do you check the total number of CPU cores available?
3. What is the difference between nproc and lscpu?
4. How can you display all CPU cores, including restricted ones?
5. How do you limit a process to use only a certain number of CPU cores?
6. How does nproc behave inside a Docker container?
7. What command would you use to find the number of physical CPU cores?
8. How does nproc help in parallel computing tasks?
9. Can nproc be used in shell scripting for CPU-intensive tasks?
10. How do you assign a specific number of CPU cores to a process?

# 4. top (Task Manager) Command in Linux

The top command provides real-time information about system performance, including CPU and memory usage, running processes, and load averages.

## 4.1 Syntax

sh

top [OPTIONS]

Without any options, top displays a dynamic, real-time view of system processes.

## 4.2 Options and Examples

### 1. Display Real-Time System Processes

sh

top

**Output:**

sql

```
top - 14:35:17 up 1 day,  3:47,  1 user,  load average: 0.25, 0.30, 0.28
Tasks: 221 total,   2 running, 219 sleeping,   0 stopped,   0 zombie
%Cpu(s):  2.0 us,  1.0 sy,  0.0 ni, 96.5 id,  0.3 wa,  0.1 hi,  0.1 si,  0.0 st
KiB Mem :  16318264 total,  10203440 free,   4096044 used,   2001880 buff/cache
KiB Swap:   2097148 total,   2097148 free,        0 used.  12204152 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 1500 root      20   0  391984  32984  11824 R   3.3  0.2   0:00.23 top
 1000 user      20   0  156568   9440   4124 S   1.0  0.1   0:10.45 gnome-shell
```

**Explanation:**

- Tasks: Shows the total number of processes.
- %Cpu(s): Displays CPU usage breakdown.
- PID: Process ID.
- USER: Process owner.
- %CPU: CPU usage percentage.
- %MEM: Memory usage percentage.
- COMMAND: Name of the running command.

## 2. Sort Processes by Memory Usage

sh

```
top -o %MEM
```

**Output:**

sql

```
(PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND)
 2000 root      20   0  1.5G  512M   123M S   1.5  10.2   1:12.34 firefox
```

```
 1500 user     20   0  1.2G  256M   64M S   1.2   8.1   0:43.22 chrome
```

**Explanation:**

- -o %MEM: Sorts the output by memory usage.

## 3. Show Only Processes of a Specific User

sh

```
top -u user
```

**Output:**

sql

```
 PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
 2500 user     20   0  391984  32984  11824 S   3.0  0.2   0:00.53 bash
```

**Explanation:**

- -u user: Displays processes for the specified user.

## 4. Show CPU Usage for Each Core

Press 1 inside top to display per-core CPU usage.

**Output:**

shell

```
%Cpu0  :  2.3 us,  1.1 sy,  0.0 ni, 95.5 id,  0.2 wa,  0.2 hi,  0.7 si,  0.0 st
%Cpu1  :  1.8 us,  0.9 sy,  0.0 ni, 96.8 id,  0.3 wa,  0.1 hi,  0.1 si,  0.0 st
```

**Explanation:**

- Displays CPU usage for each core.

### 5. Kill a Process from top

Press k, enter the PID, and specify the signal (9 for force kill).

Example:

yaml

Kill PID: 2500
Send signal [15]: 9

- This kills process 2500 using SIGKILL.

## 4.3 Common Interview Questions on top

1. What does the top command do in Linux?
2. How can you display only processes of a specific user?
3. How do you sort processes by memory usage?
4. What key do you press inside top to show per-core CPU usage?
5. How do you kill a process using top?
6. How does top differ from ps?
7. How can you make top update less frequently?
8. What's the difference between load average and %CPU?
9. How can you monitor real-time network activity using top?
10. How do you save top output to a file?

## 5. ps (Process Status) Command in Linux

The ps command provides a snapshot of the currently running processes, their process IDs (PIDs), CPU and memory usage, and other details.

## 5.1 Syntax

sh

ps [OPTIONS]

Without options, ps displays only processes from the current shell session.

## 5.2 Options and Examples

### 1. Display Currently Running Processes in the Shell

sh

ps

**Output:**

yaml

```
 PID TTY          TIME CMD
3201 pts/0    00:00:00 bash
4002 pts/0    00:00:00 ps
```

**Explanation:**

- PID: Process ID.
- TTY: Terminal controlling the process.
- TIME: Total CPU time used by the process.
- CMD: Command that started the process.

## 2. Show All Running Processes

sh

```
ps -e
```

**Output:**

yaml

```
 PID TTY          TIME CMD
   1 ?        00:01:24 systemd
 342 ?        00:00:02 sshd
 983 ?        00:00:14 apache2
2567 ?        00:00:05 mysql
3999 pts/0    00:00:00 bash
4005 pts/0    00:00:00 ps
```

**Explanation:**

- -e: Displays all system processes.

## 3. Show Detailed Process Information

sh

```
ps -ef
```

**Output:**

yaml

```
UID        PID  PPID  C STIME TTY          TIME CMD
root         1     0  0 10:00 ?        00:01:24 systemd
root       342     1  0 10:01 ?        00:00:02 sshd
mysql     2567     1  0 10:10 ?        00:00:05 mysqld
```

```
user     4005  3999  0 14:35 pts/0    00:00:00 ps -ef
```

**Explanation:**

- -f: Full format.
- UID: User ID of the process owner.
- PPID: Parent process ID.
- C: CPU utilization.
- STIME: Start time of the process.

## 4. Find a Specific Process by Name

sh

```
ps -ef | grep apache2
```

**Output:**

kotlin

```
root      983    1  0 10:05 ?       00:00:14 apache2
www-data  1002   983  0 10:06 ?       00:00:02 apache2
```

**Explanation:**

- grep apache2: Filters processes containing "apache2".

## 5. Show Processes in a Tree Format

sh

```
ps -e --forest
```

**Output:**

yaml

```
  1 ?      00:01:24 systemd
  ├─ 342 ?       00:00:02 sshd
  ├─ 983 ?       00:00:14 apache2
  │   ├─ 1002 ?       00:00:02 apache2
  │   ├─ 1003 ?       00:00:02 apache2
  ├─ 2567 ?       00:00:05 mysqld
```

**Explanation:**

- --forest: Shows parent-child process relationships.

## 6. Display a Specific User's Processes

sh

```
ps -u root
```

**Output:**

css

```
 PID TTY        TIME CMD
   1 ?       00:01:24 systemd
 342 ?       00:00:02 sshd
 983 ?       00:00:14 apache2
2567 ?       00:00:05 mysqld
```

**Explanation:**

- -u root: Shows processes owned by root.

### 7. Show Processes Sorted by CPU Usage

sh

```
ps -eo pid,ppid,cmd,%mem,%cpu --sort=-%cpu
```

**Output:**

perl

```
 PID  PPID CMD                  %MEM %CPU
 3201  1   firefox              5.2  20.0
 2567  1   mysqld                 2.3  15.6
 983   1   apache2                1.8  12.1
```

**Explanation:**

- -eo: Specifies custom columns.
- %cpu: CPU usage.
- %mem: Memory usage.
- --sort=-%cpu: Sorts by CPU usage (descending).

### 8. Kill a Process by PID

sh

```
kill 3201
```

- This kills process 3201 (Firefox in the example above).

## 5.3 Common Interview Questions on ps

1. What is the purpose of the ps command?
2. How do you display all running processes?
3. What is the difference between ps -e and ps -ef?

4. How can you find a process by name?
5. What command shows processes in a hierarchical tree format?
6. How do you list processes for a specific user?
7. How do you sort processes by CPU or memory usage?
8. How do you kill a process using ps?
9. What's the difference between ps and top?
10. How can you monitor real-time process execution using ps?

# 6. grep (Global Regular Expression Print) Command in Linux

The grep command searches for a specific pattern in files or output and prints matching lines.

## 6.1 Syntax

sh

grep [OPTIONS] PATTERN [FILE...]

- PATTERN: The text or regex pattern to search for.
- FILE: The file(s) to search in.

## 6.2 Options and Examples

### 1. Search for a Word in a File

sh

grep "error" log.txt

**Output:**

css

[ERROR] Failed to connect to database

**Explanation:**

- Searches for error in log.txt and prints matching lines.

## 2. Perform a Case-Insensitive Search

sh

grep -i "error" log.txt

**Output:**

css

[ERROR] Failed to connect to database
[error] File not found

**Explanation:**

- -i: Ignores case.

## 3. Display Line Numbers with Matches

sh

grep -n "failed" log.txt

**Output:**

vbnet

5:[ERROR] Failed to connect to database
9:[WARNING] Login failed for user

**Explanation:**

- -n: Shows line numbers.

## 4. Search Recursively in All Files

sh

grep -r "timeout" /var/log/

**Output:**

bash

/var/log/syslog: Network timeout occurred
/var/log/auth.log: Authentication timeout

**Explanation:**

- -r: Searches in subdirectories.

## 5. Show Only Matching Words

sh

grep -o "failed" log.txt

**Output:**

failed
failed

**Explanation:**

- -o: Prints only the matched text.

## 6. Count the Number of Matches

sh

grep -c "error" log.txt

**Output:**

3

**Explanation:**

- -c: Displays count instead of matching lines.

## 7. Show Lines That Do Not Match

sh

grep -v "success" log.txt

**Output:**

css

[ERROR] Failed to connect to database
[WARNING] Disk space low

**Explanation:**

- -v: Prints lines that do not contain success.

### 8. Use Extended Regular Expressions

sh

grep -E "error|failed|warning" log.txt

**Output:**

css

[ERROR] Failed to connect to database
[WARNING] Disk space low

**Explanation:**

- -E: Enables extended regex.

## 6.3 Common Interview Questions on grep

1. What is the grep command used for?
2. How do you perform a case-insensitive search?
3. How do you count occurrences of a pattern?
4. How do you exclude lines that match a pattern?
5. How can you search recursively in all files?
6. What is the difference between grep and grep -E?
7. How do you display only matching words instead of entire lines?
8. How do you use grep with multiple patterns?

9. How can you use grep in a pipeline?
10. How do you search for an exact word match?

# 7. awk (Pattern Scanning and Processing) Command in Linux

The awk command is a powerful text-processing tool that processes and analyzes text files, extracts data, and performs transformations.

## 7.1 Syntax

sh

awk [OPTIONS] 'pattern { action }' file

- pattern: Specifies when the action should be applied.
- action: Defines what to do with the matching lines.
- file: The input file.

## 7.2 Options and Examples

### 1. Print All Lines in a File

sh

awk '{ print }' data.txt

**Output (data.txt):**

Alice 25 Developer
Bob 30 Manager
Charlie 28 Designer

**Explanation:**

- { print }: Prints each line.

## 2. Print a Specific Column

sh

awk '{ print $1 }' data.txt

**Output:**

Alice
Bob
Charlie

**Explanation:**

- $1: First column.

## 3. Print Specific Columns with Formatting

sh

awk '{ print "Name:", $1, "Age:", $2 }' data.txt

**Output:**

makefile

Name: Alice Age: 25
Name: Bob Age: 30
Name: Charlie Age: 28

**Explanation:**

- Formats output with labels.

## 4. Filter Rows Based on a Condition

sh

```
awk '$2 > 25 { print $1, "is older than 25" }' data.txt
```

**Output:**

csharp

```
Bob is older than 25
Charlie is older than 25
```

**Explanation:**

- $2 > 25: Selects rows where the second column is greater than 25.

## 5. Search for a Specific Pattern

sh

```
awk '/Manager/ { print $1, "is a Manager" }' data.txt
```

**Output:**

csharp

```
Bob is a Manager
```

**Explanation:**

- Matches lines containing "Manager."

## 6. Calculate Sum of a Column

sh

```
awk '{ sum += $2 } END { print "Total Age:", sum }' data.txt
```

**Output:**

mathematica

```
Total Age: 83
```

**Explanation:**

- sum += $2: Adds up column 2.
- END { print sum }: Prints after processing.

## 7. Print Line Numbers

sh

```
awk '{ print NR, $0 }' data.txt
```

**Output:**

```
1 Alice  25  Developer
2 Bob    30  Manager
3 Charlie 28  Designer
```

**Explanation:**

- NR: Line number.

### 8. Find and Replace Text

sh

```
awk '{ gsub(/Developer/, "Engineer"); print }' data.txt
```

**Output:**

```
Alice  25  Engineer
Bob    30  Manager
Charlie 28  Designer
```

**Explanation:**

- gsub(/pattern/, "replacement"): Replaces text.

## 7.3 Common Interview Questions on awk

1. What is awk used for?
2. How do you print only a specific column?
3. How do you filter lines based on a condition?
4. What does $1, $2, etc., represent?
5. How do you sum a column of numbers?
6. How do you use awk to replace text?
7. What is the difference between NR and NF?
8. How do you print only lines containing a specific word?
9. How do you print lines where a column matches a pattern?
10. How do you use awk in combination with grep or sed?

# 8. sed (Stream Editor) Command in Linux

The sed command is used for text manipulation, including searching, replacing, inserting, and deleting lines.

## 8.1 Syntax

sh

sed [OPTIONS] 'COMMAND' file

- COMMAND: Specifies what transformation to apply.
- file: The input file.

## 8.2 Options and Examples

### 1. Replace a Word in a File

sh

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

**Before (file.txt):**

sql

Hello old world.

**After Output:**

arduino

Hello new world.

**Explanation:**

- s/old/new/: Substitutes old with new in the first occurrence per line.

## 2. Replace All Occurrences in a Line

sh

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

**Before:**

csharp

old is replaced with new. old should be gone.

**After:**

csharp

new is replaced with new. new should be gone.

**Explanation:**

- g: Global replacement (all occurrences per line).

## 3. Replace in a Specific Line

sh

sed '2s/old/new/' file.txt

**Explanation:**

- Only replaces old with new on line 2.

## 4. Delete a Specific Line

sh

```
sed '3d' file.txt
```

**Explanation:**

- 3d: Deletes line 3.

## 5. Delete All Empty Lines

sh

```
sed '/^$/d' file.txt
```

**Explanation:**

- /^$/d: Deletes lines that are empty.

## 6. Print Only Certain Lines

sh

```
sed -n '2,4p' file.txt
```

**Explanation:**

- -n: Suppresses default output.
- 2,4p: Prints lines 2 to 4.

### 7. Insert a Line Before Another Line

sh

sed '2i\This is a new line' file.txt

**Explanation:**

- 2i\: Inserts text before line 2.

### 8. Append a Line After Another Line

sh

sed '3a\This is an added line' file.txt

**Explanation:**

- 3a\: Adds text after line 3.

## 8.3 Common Interview Questions on sed

1. What is the sed command used for?
2. How do you replace all occurrences of a word in a file?
3. How do you delete a specific line using sed?
4. How do you insert a new line before a specific line?
5. How do you delete blank lines in a file?
6. What is the difference between sed and awk?
7. How do you print only a range of lines using sed?
8. How do you perform an in-place file update with sed?
9. How do you replace text only in a specific line?
10. How do you use sed with regex patterns?

# 9. cut Command in Linux

The cut command is used for cutting out sections of each line from a file or input stream. It's useful for extracting columns or specific characters from text.

## 9.1 Syntax

sh

cut [OPTIONS] [FILE]

- -f FIELD: Selects specific fields (columns).
- -d DELIM: Specifies the delimiter to use for splitting fields.
- -c RANGE: Selects specific characters from each line.

## 9.2 Options and Examples

### 1. Cut Based on Delimiters (Fields)

sh

cut -d "," -f 1,3 data.csv

**Before (data.csv):**

Alice,25,Developer
Bob,30,Manager
Charlie,28,Designer

**After Output:**

Alice,Developer
Bob,Manager

Charlie,Designer

**Explanation:**

- -d ",": Specifies the comma as the delimiter.
- -f 1,3: Selects fields 1 and 3.

## 2. Cut Based on Character Position

sh

cut -c 1-5 file.txt

**Before (file.txt):**

arduino

Hello World
Test String
Cut this line

**After Output:**

Hello
Test
Cut t

**Explanation:**

- -c 1-5: Selects characters from position 1 to 5.

## 3. Cut and Exclude a Specific Field

sh

```
cut -d ":" -f 1,3 /etc/passwd
```

**Explanation:**

- -d ":": Uses colon as delimiter.
- -f 1,3: Selects fields 1 and 3 from /etc/passwd.

## 4. Cut Using Multiple Delimiters

sh

```
cut -d ":" -f 2 /etc/passwd | cut -d "," -f 1
```

**Explanation:**

- Extracts the second field from /etc/passwd using colon, then extracts the first field using comma.

## 5. Print Multiple Ranges of Characters

sh

```
cut -c 1-4,7-10 file.txt
```

**Before (file.txt):**

```
abcdefg
1234567
hello123
```

**After Output:**

yaml

abcd
1234
hello

**Explanation:**

- -c 1-4,7-10: Selects characters from position 1 to 4 and from 7 to 10.

## 9.3 Common Interview Questions on cut

1. What is the cut command used for?
2. How do you extract specific fields from a file using cut?
3. How do you specify a custom delimiter in cut?
4. What is the difference between cut and awk?
5. How can you select specific characters from a line using cut?
6. How do you handle multiple delimiters with cut?
7. Can you use cut to extract specific characters and fields simultaneously?
8. How do you use cut with pipes to filter output?
9. How would you select and print columns in a CSV file using cut?
10. How do you remove a specific column from output using cut?

# 10. tr (Translate) Command in Linux

The tr command is used to translate or delete characters from a stream of text. It's commonly used for replacing or removing specific characters, converting case, and squeezing repeating characters.

## 10.1 Syntax

sh

tr [OPTION]... SET1 [SET2]

- SET1: The set of characters to be replaced or deleted.
- SET2: The set of characters to replace characters in SET1.

## 10.2 Options and Examples

### 1. Convert Lowercase to Uppercase

sh

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

**Output:**

```
HELLO WORLD
```

**Explanation:**

- Translates all lowercase characters (a-z) to uppercase (A-Z).

### 2. Remove Specific Characters

sh

```
echo "Hello, World!" | tr -d '!,'
```

**Output:**

```
Hello World
```

**Explanation:**

- -d: Deletes characters (removes ! and ,).

## 3. Replace a Character with Another

sh

```
echo "hello world" | tr ' ' '_'
```

**Output:**

```
hello_world
```

**Explanation:**

- Replaces spaces with underscores.

## 4. Squeeze Repeated Characters

sh

```
echo "aaabbbccc" | tr -s 'a' 'b'
```

**Output:**

```
bbb
```

**Explanation:**

- -s: Squeezes (reduces) repeated occurrences of a into a single b.

### 5. Delete All Newlines

sh

```
echo -e "hello\nworld" | tr -d '\n'
```

**Output:**

```
helloworld
```

**Explanation:**

- Deletes newline characters.

### 6. Replace Multiple Characters Simultaneously

sh

```
echo "apple banana" | tr 'ae' 'uo'
```

**Output:**

```
upplo bunono
```

**Explanation:**

- Replaces a with u and e with o in the string.

## 10.3 Common Interview Questions on tr

1. What is the tr command used for?
2. How do you convert lowercase letters to uppercase using tr?

3. How do you delete specific characters using tr?
4. How do you replace one character with another using tr?
5. What does the -s option in tr do?
6. How do you replace multiple characters at once in tr?
7. Can you use tr to remove spaces from a string?
8. How do you delete newlines using tr?
9. What is the difference between tr and sed?
10. How can you use tr to clean up input data by removing unwanted characters?

# 11. bc (Basic Calculator) Command in Linux

The bc command is a command-line calculator that supports mathematical operations, including basic arithmetic, variables, and functions. It supports both integer and floating-point calculations.

## 11.1 Syntax

sh

```
echo "EXPRESSION" | bc
```

- EXPRESSION: Mathematical expression to evaluate.

## 11.2 Options and Examples

### 1. Perform Basic Arithmetic

sh

```
echo "5 + 3" | bc
```

**Output:**

8

**Explanation:**

- Performs addition of 5 and 3.

## 2. Perform Division with Decimal Precision

sh

```
echo "10 / 3" | bc
```

**Output:**

3

**Explanation:**

- By default, bc truncates division results to integer values.

## 3. Set Decimal Precision

sh

```
echo "scale=2; 10 / 3" | bc
```

**Output:**

3.33

**Explanation:**

- scale=2: Sets the decimal precision to 2 places.

## 4. Use Variables

sh

```
echo "a=10; b=5; a + b" | bc
```

**Output:**

15

**Explanation:**

- a and b are variables. This performs addition of a and b.

## 5. Calculate Square Root

sh

```
echo "scale=3; sqrt(16)" | bc
```

**Output:**

4.000

**Explanation:**

- sqrt(16): Calculates the square root of 16 with 3 decimal places of precision.

## 6. Perform Complex Calculations

sh

```
echo "scale=3; (5 + 3) * (10 / 2)" | bc
```

**Output:**

```
40.000
```

**Explanation:**

- Performs arithmetic operations with parentheses to determine the order of operations.

## 7. Use Built-in Functions

sh

```
echo "scale=4; s(1)" | bc -l
```

**Output:**

```
0.8415
```

**Explanation:**

- s(1): Calculates the sine of 1 (in radians). The -l option loads the standard math library.

## 8. Perform Exponentiation

sh

```
echo "2^3" | bc
```

**Output:**

```
8
```

**Explanation:**

- ^: Exponentiation operator, calculates 2^3.

## 11.3 Common Interview Questions on bc

1. What is the bc command used for?
2. How do you perform floating-point division in bc?
3. How do you set the decimal precision for calculations in bc?
4. How do you calculate the square root of a number using bc?
5. Can you perform exponentiation in bc? If so, how?
6. How do you use variables in bc?
7. What is the purpose of the -l option in bc?
8. How can you perform complex mathematical expressions using bc?
9. How do you perform trigonometric operations in bc?
10. What happens if you perform division without setting the scale in bc?

# 12. curl Command in Linux

The curl command is a versatile tool used for transferring data from or to a server. It supports various protocols, including HTTP, HTTPS, FTP, and more. It is often used for downloading or uploading files, testing APIs, and fetching web pages.

## 12.1 Syntax

```
sh
```

curl [OPTIONS] [URL]

- URL: The URL or the location from which to fetch or send data.
- OPTIONS: Additional flags to customize the behavior of the command.

## 12.2 Options and Examples

### 1. Fetch a Web Page

sh

curl https://www.example.com

**Explanation:**

- Downloads the HTML content of the specified URL and prints it to the terminal.

### 2. Save the Output to a File

sh

curl https://www.example.com -o page.html

**Explanation:**

- -o page.html: Saves the output to a file (page.html).

### 3. Download a File

sh

curl -O https://www.example.com/sample.txt

**Explanation:**

- -O: Downloads the file with its original name.

## 4. Follow Redirects

sh

curl -L https://bit.ly/xyz

**Explanation:**

- -L: Follows redirects automatically if the URL points to a different location.

## 5. Send a GET Request (Default Method)

sh

curl https://jsonplaceholder.typicode.com/posts

**Explanation:**

- By default, curl sends a GET request to fetch data from the URL.

## 6. Send a POST Request with Data

sh

curl -X POST -d "name=John&age=25" https://jsonplaceholder.typicode.com/posts

**Explanation:**

- -X POST: Specifies the HTTP method (POST).
- -d: Sends data with the POST request (form data in this case).

## 7. Include Headers in the Output

sh

curl -i https://www.example.com

**Explanation:**

- -i: Includes the HTTP headers in the output.

## 8. Set a Custom Header

sh

curl -H "Content-Type: application/json" https://jsonplaceholder.typicode.com/posts

**Explanation:**

- -H: Adds a custom header to the request.

## 9. Limit the Download Speed

sh

curl --limit-rate 100k https://www.example.com/largefile.zip

**Explanation:**

- --limit-rate 100k: Limits the download speed to 100 KB/s.

### 10. Upload a File

sh

curl -T file.txt ftp://ftp.example.com/

**Explanation:**

- -T: Uploads a file to the specified FTP server.

## 12.3 Common Interview Questions on curl

1. What is the curl command used for?
2. How do you download a file with curl?
3. How do you save the output of a curl command to a file?
4. How do you send data using a POST request with curl?
5. How can you follow redirects with curl?
6. How do you include HTTP headers in the output with curl?
7. What is the difference between -O and -o options in curl?
8. How do you limit the download speed with curl?
9. How can you upload a file using curl?
10. How do you set custom headers when sending a curl request?

## 13. wget Command in Linux

The wget command is used for non-interactive downloading of files from the web. It supports HTTP, HTTPS, and FTP protocols and can retrieve content from remote servers while offering options for recursion, mirroring, and automatic handling of files.

## 13.1 Syntax

sh

wget [OPTIONS] [URL]

- URL: The location to fetch the file from.
- OPTIONS: Additional flags to control the download behavior.

## 13.2 Options and Examples

### 1. Download a Single File

sh

wget https://www.example.com/sample.txt

**Explanation:**

- Downloads the file sample.txt from the specified URL to the current directory.

### 2. Download a File with a Custom Filename

sh

wget -O newfile.txt https://www.example.com/sample.txt

**Explanation:**

- -O newfile.txt: Specifies the custom filename (newfile.txt) for the downloaded file.

### 3. Download Files Recursively

sh

wget -r https://www.example.com/files/

**Explanation:**

- -r: Downloads files recursively from the specified directory.

## 4. Limit Download Speed

sh

wget --limit-rate=200k https://www.example.com/largefile.zip

**Explanation:**

- --limit-rate=200k: Limits the download speed to 200 KB/s.

## 5. Resume a Download

sh

wget -c https://www.example.com/largefile.zip

**Explanation:**

- -c: Resumes an interrupted download.

## 6. Download a File in the Background

sh

wget -b https://www.example.com/largefile.zip

**Explanation:**

- -b: Downloads the file in the background.

## 7. Download a Website (Mirror a Site)

sh

wget -m https://www.example.com/

**Explanation:**

- -m: Mirrors the entire website, downloading all linked content and files.

## 8. Download Only Specific File Types

sh

wget -r -A .pdf https://www.example.com/

**Explanation:**

- -r: Downloads recursively.
- -A .pdf: Downloads only .pdf files.

## 9. Specify Maximum Number of Retries

sh

wget --tries=3 https://www.example.com/sample.txt

**Explanation:**

- --tries=3: Limits the number of retry attempts to 3 in case of download failure.

## 10. Download from FTP with Authentication

sh

wget --ftp-user=username --ftp-password=password ftp://ftp.example.com/sample.txt

**Explanation:**

- --ftp-user and --ftp-password: Specify the FTP credentials for downloading the file.

## 13.3 Common Interview Questions on wget

1. What is the wget command used for?
2. How do you download a file using wget?
3. How can you download a file and save it with a different name using wget?
4. How do you download a website using wget?
5. How can you resume an interrupted download with wget?
6. How do you limit the download speed in wget?
7. What is the purpose of the -r option in wget?
8. How do you download files of specific types (e.g., only .pdf) using wget?
9. How do you run a wget command in the background?
10. How do you handle FTP downloads with authentication using wget?

## 14. find Command in Linux

The find command is used to search for files and directories in a directory hierarchy based on various criteria like name, size, type, permissions, etc. It is an extremely powerful tool for locating files and performing actions on them.

## 14.1 Syntax

sh

find [PATH] [OPTIONS] [EXPRESSION]

- PATH: The directory where the search begins (can be . for the current directory).
- OPTIONS: Flags to modify the behavior of the search.
- EXPRESSION: The search criteria like filename, size, or modification time.

## 14.2 Options and Examples

### 1. Find Files by Name

sh

```
find /home/user -name "file.txt"
```

**Explanation:**

- Searches for a file named file.txt starting from /home/user.

### 2. Find Files by Extension

sh

```
find /home/user -name "*.log"
```

**Explanation:**

- Searches for all files with the .log extension.

### 3. Find Files with Specific Permissions

sh

```
find /home/user -perm 644
```

**Explanation:**

- Searches for files with permissions 644.

## 4. Find Files by Size

sh

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

**Explanation:**

- Finds files larger than 100MB.

## 5. Find Files Modified Within the Last 7 Days

sh

```
find /home/user -mtime -7
```

**Explanation:**

- Finds files modified in the last 7 days.

## 6. Find Files and Execute a Command

sh

```
find /home/user -name "*.log" -exec rm {} \;
```

**Explanation:**

- Searches for .log files and deletes them using rm. {} is replaced with the found files.

## 7. Find Empty Files or Directories

sh

find /home/user -empty

**Explanation:**

- Finds empty files or directories.

## 8. Find Files Based on Access Time

sh

find /home/user -atime -5

**Explanation:**

- Finds files that were accessed in the last 5 days.

## 9. Find Files with Multiple Conditions

sh

find /home/user -name "*.txt" -size +1M -mtime -10

**Explanation:**

- Finds .txt files larger than 1MB and modified in the last 10 days.

## 10. Find Files and Print Detailed Information

sh

find /home/user -name "*.log" -ls

**Explanation:**

- -ls: Prints detailed information about each found file (similar to ls -l).

## 14.3 Common Interview Questions on find

1. What is the find command used for?
2. How do you search for files by name using find?
3. How can you find files of a specific size using find?
4. How do you search for files modified within the last 7 days using find?
5. How do you delete files found by find?
6. How do you find empty files or directories with find?
7. What does the -exec option do in find?
8. How do you find files based on their access time with find?
9. How can you find files with multiple conditions in find?
10. How do you print detailed information about found files using find?

# 15. trap Command in Linux

The trap command is used to catch and handle signals and events in a script. It allows you to specify actions that should be executed when certain signals are received or when specific events (such as errors or script exits) occur.

## 15.1 Syntax

sh

trap [COMMAND] [SIGNAL]

- COMMAND: The command to execute when the signal occurs.
- SIGNAL: The signal or event that triggers the command (e.g., EXIT, SIGINT, SIGTERM).

## 15.2 Options and Examples

### 1. Catch the Exit of a Script

sh

```
trap 'echo "Exiting script..."' EXIT
```

**Explanation:**

- The EXIT signal triggers the command echo "Exiting script..." when the script finishes or exits.

### 2. Handle the Interrupt Signal (Ctrl+C)

sh

```
trap 'echo "Ctrl+C pressed!"' SIGINT
```

**Explanation:**

- SIGINT is triggered when the user presses Ctrl+C, and the message Ctrl+C pressed! is displayed instead of terminating the script.

### 3. Clean Up Temporary Files Before Exiting

sh

```
trap 'rm -f /tmp/tempfile' EXIT
```

**Explanation:**

- Ensures that /tmp/tempfile is deleted when the script exits (whether normally or due to an error).

## 4. Handle the Termination Signal (SIGTERM)

sh

```
trap 'echo "Terminating the script..."' SIGTERM
```

**Explanation:**

- SIGTERM is triggered when the script receives a termination request (e.g., kill command), and the message Terminating the script... is displayed.

## 5. Handle Errors in a Script

sh

```
trap 'echo "An error occurred!"' ERR
```

**Explanation:**

- The ERR signal catches any errors that occur during the script execution, printing the message An error occurred!.

## 6. Ignore a Specific Signal

sh

```
trap '' SIGINT
```

**Explanation:**

- The SIGINT signal is ignored, preventing the script from being interrupted by Ctrl+C.

### 7. Trap Multiple Signals

sh

```
trap 'echo "Received a signal!"' SIGINT SIGTERM
```

**Explanation:**

- Catches both SIGINT and SIGTERM signals, printing the message Received a signal! when either is received.

### 8. Reset the Trap for a Signal

sh

```
trap - SIGINT
```

**Explanation:**

- Resets the trap for SIGINT, so the script will resume its default behavior when Ctrl+C is pressed.

## 15.3 Common Interview Questions on trap

1. What is the trap command used for in Linux?
2. How can you execute a command when the script exits using trap?
3. How do you handle the Ctrl+C interrupt signal in a script?
4. How do you clean up temporary files before exiting a script using trap?
5. What signal is triggered when a script is terminated with the kill command, and how can you handle it?
6. How can you handle errors in a script using the trap command?
7. How do you ignore a specific signal using trap?
8. How can you trap multiple signals simultaneously in a script?
9. What does the trap - command do?
10. Can trap be used to handle signals in both interactive and non-interactive scripts?

# Shell Script Debugging Options

The set command in shell scripting is used to change the behavior of the shell. The -x, -e, and -o pipefail options are primarily used for debugging shell scripts, allowing you to trace script execution, handle errors, and control the behavior of pipelines.

## 1. set -x (Enable Debugging)

### Explanation:

- The set -x command enables debugging in a shell script. When this option is enabled, the shell prints each command and its arguments to the terminal before executing it.
- It's useful for tracing the flow of execution and understanding how the script processes commands.

### Example:

sh

```
#!/bin/bash

set -x

echo "This is a debug test"
x=5
y=10
z=$((x + y))
echo "Result: $z"
```

**Output:**

bash

```
+ echo 'This is a debug test'
This is a debug test
+ x=5
+ y=10
+ z=15
+ echo 'Result: 15'
Result: 15
```

**Explanation:**

- The + symbol before each command shows the command that is being executed. This helps to trace what's happening at each step.

## 2. set -e (Exit on Error)

### Explanation:

- The set -e option causes the script to exit immediately if any command returns a non-zero exit status (indicating an error).
- This helps prevent the script from continuing with incorrect or unexpected results. However, it does not apply if a command is part of a pipeline or if it is explicitly ignored using ||.

### Example:

sh

```
#!/bin/bash

set -e

echo "This is a test"
nonexistent_command  # This will cause the script to exit
echo "This line will not be executed"
```

**Output:**

bash

This is a test
./script.sh: line 4: nonexistent_command: command not found

**Explanation:**

- The script exits when it encounters the error (nonexistent_command), and the line echo "This line will not be executed" is not executed.

## 3. set -o pipefail (Exit on Pipeline Failure)

### Explanation:

- The set -o pipefail option ensures that if any command in a pipeline fails, the entire pipeline will return a non-zero exit status.
- By default, in a pipeline, only the exit status of the last command is returned. With set -o pipefail, the return status of the entire pipeline reflects the status of the command that failed.

### Example:

sh

#!/bin/bash

set -o pipefail

echo "This is a test" | nonexistent_command
echo "This line will not be executed"

**Output:**

bash

./script.sh: line 4: nonexistent_command: command not found

**Explanation:**

- Since set -o pipefail is enabled, the script exits immediately when nonexistent_command fails in the pipeline, even though it's not the last command in the pipeline.

## Common Interview Questions on set Options

1. What is the purpose of set -x in a shell script?
2. How does set -e help in error handling in shell scripts?
3. Can set -e be disabled after it is enabled in a script? How?
4. How does set -o pipefail improve error handling in pipelines?
5. What happens if set -e is enabled but the error occurs in a pipeline? How can you handle it?
6. How do you enable or disable debugging in a script?
7. What is the behavior of set -x when used with conditional statements or loops?
8. Why would you use set -o pipefail in a script with complex pipelines?
9. Can you combine set -e with set -o pipefail for more robust error handling? How does it behave?
10. How does set -e interact with the || operator in shell scripting?