Day-7 SRE Training

Topic - Linux Commands and Basic Shell Scripting

`rmdir -p` → removes **empty** directories along with their parent directories **recursively** if they are also empty.

mkdir -p a/b/c → Creates nested directories `a/b/c`, ensuring parent directories exist.

rmdir -p a/b/c → Removes `c`, then `b` (if empty), then `a` (if empty).

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ mkdir -p a/b/c
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ ls -lrt
total 4
drwxr-xr-x 3 veenaroot veenaroot 4096 Feb 18 15:59 a
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ rmdir -p a/b/c
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ ls -lrt
total 0
```

mkdir -p x/y/z

touch x/y/file.txt # Creates a file inside 'y'

rmdir -p x/y/z # Removes only 'z' because 'y' is not empty

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ mkdir -p x/y/z
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ ls -lrt
total 4
drwxr-xr-x 3 veenaroot veenaroot 4096 Feb 18 16:01 x
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ cd x
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x$ cd y
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x/y$ touch new.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x/y$ cd ../..
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ rmdir -p x/y/z
rmdir: failed to remove directory 'x/y': Directory not empty
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux$ cd x
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x$ cd y
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x/y$ ls -lrt
total 0
-rw-r--r-- 1 veenaroot veenaroot 0 Feb 18 16:01 new.txt
```

`uname -a` → Displays all system information (kernel name, version, architecture, etc.).

`uname -s` → Shows the kernel name (e.g., Linux).

`uname -r` → Displays the kernel release version.

`uname -m` → Shows the system architecture (e.g., x86_64).

`uname` → By default, prints the kernel name (same as `uname -s`).

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x/y$ uname -a
Linux LAPTOP-S0KHU6AM 5.15.167.4-microsoft-standard-WSL2 #1 SMP Tue Nov 5 00:21:55 UTC 2024 x86_64 x86_64 x86_64 GNU/Lin
ux
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x/y$ uname -s
Linux
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x/y$ uname -r
5.15.167.4-microsoft-standard-WSL2
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x/y$ uname -m
x86_64
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice/linux/x/y$ uname
Linux
```

ps aux → Lists all running processes with detailed information.

kill processid → Terminates a process by its Process ID (PID).

kill -9 processid → Forcefully terminates a process.

sort → Sorts lines of text in ascending order.

By default, `sort` is case-sensitive. Uppercase letters come before lowercase letters in the **ASCII** table. To ignore case sensitivity, use the `-f` flag.

sort -r → Sorts lines of text in reverse (descending) order.

sort -n → Sorts lines of text numerically.

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ sort a.txt
Hi
Mahalakshmi
a
b
d
d
e
f
g
r
s
v
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ sort -f a.txt
a
b
d
d
e
f
g
Hi
Mahalakshmi
r
s
```

wc -w filename.txt → Counts the number of words in the file `filename.txt`

wc -l filename.txt → Counts the number of lines in the file `filename.txt`

wc -c filename.txt → Counts the number of bytes (characters) in the file `filename.txt`

wc * → Counts words, lines, and bytes for all files in the current directory

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ wc -w a.txt
13 a.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ wc -l a.txt
12 a.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ wc -c a.txt
41 a.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ wc *
wc: a: Is a directory
      0       0       0 a
     12      13      41 a.txt
      1       1       5 b.txt
wc: linux: Is a directory
      0       0       0 linux
      2       6   10240 new.tar
     15      15      40 num.txt
     30      35   10326 total
```

ncal → Displays the current month's calendar.

ncal 12 2025 → Displays the calendar for December 2025.

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ ncal
    February 2025
Su     2  9 16 23
Mo     3 10 17 24
Tu     4 11 18 25
We     5 12 19 26
Th     6 13 20 27
Fr     7 14 21 28
Sa  1  8 15 22
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ ncal 12 2025
    December 2025
Su     7 14 21 28
Mo  1  8 15 22 29
Tu  2  9 16 23 30
We  3 10 17 24 31
Th  4 11 18 25
Fr  5 12 19 26
Sa  6 13 20 27
```

ncal -3 → Displays the previous, current, and next months' calendars.

ncal 1990 → Displays the calendar for the entire year 1990.

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ ncal -3
     January 2025       February 2025      March 2025
Su     5 12 19 26         2  9 16 23         2  9 16 23 30
Mo     6 13 20 27         3 10 17 24         3 10 17 24 31
Tu     7 14 21 28         4 11 18 25         4 11 18 25
We  1  8 15 22 29         5 12 19 26         5 12 19 26
Th  2  9 16 23 30         6 13 20 27         6 13 20 27
Fr  3 10 17 24 31         7 14 21 28         7 14 21 28
```

find -name "*.txt" → This command finds all files with a `.txt` extension in the current directory and subdirectories.

find . -type d → This command finds all directories (excluding files) starting from the current directory (`.`).

find . -name "*.tmp" -exec rm {} \; → This command finds all files with a `.tmp` extension in the current directory and subdirectories and removes them.

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ find -name "*.txt"
./linux/x/y/new.txt
./b.txt
./a.txt
./num.txt
```

history | grep command → `history` displays the list of all previously used commands in the terminal.

`|` is the pipe operator, which passes the output of the `history` command to the next command.

`grep command` searches through the output of `history` for any command that contains the word `command` (you can replace `command` with any text you want to search for).

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ history | grep sort
  164  sort a.txt
  165  sort -f a.txt
  166  sort -rf a.txt
  169  sort -n num.txt
  183  history | grep sort
```

`tar -cvf archive.tar directory/`

- `tar`: The command for working with tar archives.
- `-c`: Create a new archive.
- `-v`: Verbose mode, shows the progress.
- `-f`: Specifies the name of the archive (in this case `archive.tar`).

`tar -xvf archive.tar`

- `-x`: Extract the contents of the archive.
- `-v`: Verbose mode, shows the extraction process.
- `-f`: Specifies the archive file (`archive.tar`).

`tar -tvf archive.tar`

- `-t`: List the contents of the archive without extracting it.
- `-v`: Verbose mode, lists files in detail.
- `-f`: Specifies the archive file.

`tar -czvf archive.tar.gz directory/`

- `-z`: Compress the archive using gzip.
- `archive.tar.gz`: The name of the compressed archive file.

`tar -xzvf archive.tar.gz`

- `-z`: Decompress the archive using gzip.
- `archive.tar.gz`: The compressed archive to extract.

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ tar -xvf new.tar
a.txt
b.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ ls -lrt
total 20
-rw-r--r-- 1 veenaroot veenaroot     8 Feb 18 09:52 a.txt
-rw-r--r-- 1 veenaroot veenaroot     5 Feb 18 09:52 b.txt
-rw-r--r-- 1 veenaroot veenaroot 10240 Feb 18 09:53 new.tar
```

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ vi a.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ vi b.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ ls -lrt
total 8
-rw-r--r-- 1 veenaroot veenaroot 8 Feb 18 09:52 a.txt
-rw-r--r-- 1 veenaroot veenaroot 5 Feb 18 09:52 b.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ cd ..
veenaroot@LAPTOP-S0KHU6AM:~$ cd LinuxPractice
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ tar -cvf new.tar a.txt b.txt
a.txt
b.txt
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ ls -lrt
total 20
-rw-r--r-- 1 veenaroot veenaroot     8 Feb 18 09:52 a.txt
-rw-r--r-- 1 veenaroot veenaroot     5 Feb 18 09:52 b.txt
-rw-r--r-- 1 veenaroot veenaroot 10240 Feb 18 09:53 new.tar
```

`alias j1="ls -lrt"` → Creates an alias named `j1` for the `ls -lrt` command.

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ alias j1="ls -lrt"
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ j1
total 32
-rw-r--r-- 1 veenaroot veenaroot     5 Feb 18 15:22 b.txt
-rw-r--r-- 1 veenaroot veenaroot 10240 Feb 18 15:33 new.tar
drwxr-xr-x 3 veenaroot veenaroot  4096 Feb 18 15:59 a
drwxr-xr-x 3 veenaroot veenaroot  4096 Feb 18 16:01 linux
-rw-r--r-- 1 veenaroot veenaroot    40 Feb 18 16:15 num.txt
-rw-r--r-- 1 veenaroot veenaroot    41 Feb 18 16:22 a.txt
```

`lsof` → Lists open files and processes that have those files open, useful for identifying which process is using which file.

`ip a` → Displays network interfaces, their IP addresses, and other details about the network configuration.

`ncdu .` → A disk usage analyzer that helps visualize disk usage in a user-friendly format when run in a directory (like `.` for the current directory).

`tmux` → A terminal multiplexer that allows running multiple terminal sessions within one window, helping manage multiple tasks.

`ping www.google.com` → Sends ICMP echo requests to `www.google.com` and waits for a reply, testing network connectivity.

`echo $SHELL` → Prints the value of the `$SHELL` environment variable, showing the current shell being used (e.g., `/bin/bash`, `/bin/zsh`).

`head -2 name.txt` → Displays the first 2 lines of the file `name.txt`.

`tail -2 name.txt` → Displays the last 2 lines of the file `name.txt`.

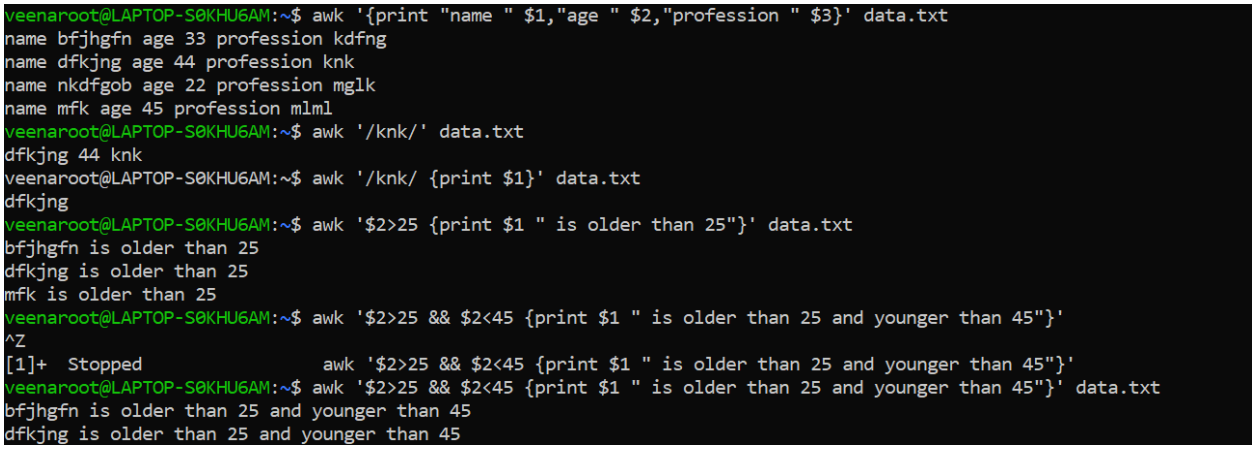
`awk '{print $1}' data.txt` → Prints the first column of each line in the `data.txt` file.

`awk '{print $1 $2 $3}' data.txt` → Prints the first, second, and third columns of each line in the `data.txt` file without any space between them.

`awk '/exp/' data.txt` → Prints lines from `data.txt` that contain the string "exp".

`awk '$2>25 {print $1 " is older than 25"}' data.txt` → Prints the first column followed by " is older than 25" for rows where the second column is greater than 25.

`awk '$2>25 && $2<45 {print $1 " is older than 25 and younger than 45"}'` → Prints the first column followed by " is older than 25 and younger than 45" for rows where the second column is greater than 25 and less than 45.



```
veenaroot@LAPTOP-S0KHU6AM:~$ awk '{print $1}' data.txt
bfjhgfn
dfkjng
nkdfgob
mfk
veenaroot@LAPTOP-S0KHU6AM:~$ awk '{print $1 $2 $3}' data.txt
bfjhgfn33kdfng
dfkjng44knk
nkdfgob22mglk
mfk45mlml
```

Shell Scripting

`echo "$var_1$var_2"` → Prints the concatenation of `var_1` and `var_2` (i.e., "Mahalakshmi").

`unset var_1` → Removes the variable `var_1`, effectively deleting its value.

`echo $var_1` → Since `var_1` was unset, it prints an empty line (as `var_1` no longer exists).

`readonly var_2` → Makes the variable `var_2` read-only, meaning its value cannot be modified after this command.

`read username` → Waits for the user to input a value and assigns it to the variable `username`.

```
var_1="Maha"
var_2="lakshmi"
echo "$var_1$var_2"
unset var_1
echo $var_1
readonly var_2
#var_2="Maha"
echo "write your username:"
read username
echo $username
```

```
veenaroot@LAPTOP-S0KHU6AM:~$ vi code.sh
veenaroot@LAPTOP-S0KHU6AM:~$ ./code.sh
Mahalakshmi

write your username:
root
root
```

`time=$(date +%H)` → Stores the current hour (in 24-hour format) into the variable `time`.

`date +%H` extracts the hour from the current date.

`if [ $time -lt 12 ]; then` → Starts an `if` statement to check if the current hour is less than 12 (i.e., morning). `-lt` is used to compare numbers (less than).

`fi` → Ends the `if-elif-else` block.

```
time=$(date +%H)
echo $time
if [ $time -lt 12 ];then
        message="Good Morning User"
elif [ $time -lt 18 ];then
        message="Good Afternoon user"
else
        message="Good Evening User"
fi
echo $message
```

```
veenaroot@LAPTOP-S0KHU6AM:~$ vi second.sh
veenaroot@LAPTOP-S0KHU6AM:~$ ./second.sh
18
Good Evening User
```

```
var_name="Mahalakshmi"
var_age=24
echo "Name is $var_name and age is $var_age"
var_blood_group="O-"
readonly var_blood_group
echo var_blood_group
echo var_blood_group="B+"
```

`echo var_blood_group="B+"` → This will print the string `var_blood_group=B+` because `echo` prints the literal string given to it. This does not change the value of the `var_blood_group` because it is read-only and cannot be reassigned.

`timedatectl` → Displays or sets system time and date information. It provides the current

`sudo timedatectl set-timezone Asia/Kolkata` → Changes the system's time zone to "Asia/Kolkata" (Indian Standard Time).

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ timedatectl
               Local time: Tue 2025-02-18 10:12:18 UTC
           Universal time: Tue 2025-02-18 10:12:18 UTC
                 RTC time: Tue 2025-02-18 16:43:49
                Time zone: Etc/UTC (UTC, +0000)
System clock synchronized: yes
              NTP service: active
          RTC in local TZ: no
```

```
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ sudo timedatectl set-timezone Asia/Kolkata
[sudo] password for veenaroot:
veenaroot@LAPTOP-S0KHU6AM:~/LinuxPractice$ timedatectl
               Local time: Tue 2025-02-18 15:43:03 IST
           Universal time: Tue 2025-02-18 10:13:03 UTC
                 RTC time: Tue 2025-02-18 16:44:35
                Time zone: Asia/Kolkata (IST, +0530)
System clock synchronized: no
              NTP service: active
          RTC in local TZ: no
```

```
i=1
```

```
while [ $i -lt 5 ]
do
   echo "$i"
   i=`expr $i + 1`
done
```

`while [ $i -lt 5 ]`:

- Starts a `while` loop. The condition `[ $i -lt 5 ]` checks if the value of `i` is less than 5.
- The loop will continue executing as long as the condition is true.

`i=\`expr $i + 1``:

- Increments the value of `i` by 1 using the `expr` command.
- `expr $i + 1` adds 1 to the current value of `i`, and the result is stored back in `i`.

```
for a in {1..9}
do
        if [ $a == 5 ]
        then
                continue
     else
        echo "$a"
      fi
done
```

```
for a in {1..9}
do
        if (( a % 2 == 0 ))
    then echo "$a is even"
    else
            echo "$a is odd"
    fi
done
```

The `[ ]` is used for string comparisons or tests. `(( ))` is used for performing arithmetic operations and comparisons.

```
read X
read Y
# echo `expr $X + $Y`
# echo `expr $X - $Y`
# echo `expr $X \* $Y`
# echo `expr $X / $Y`

echo $((X + Y))
echo $((X - Y))
echo $((X * Y))
echo $((X / Y))
```

In shell scripting, the `*` symbol is a **special character** that is used as a **wildcard** to match multiple files or directories. For example, `*.txt` would match all `.txt` files in the current directory.

The backslash `\` ensures the shell treats `*` as the multiplication operator instead of a wildcard.