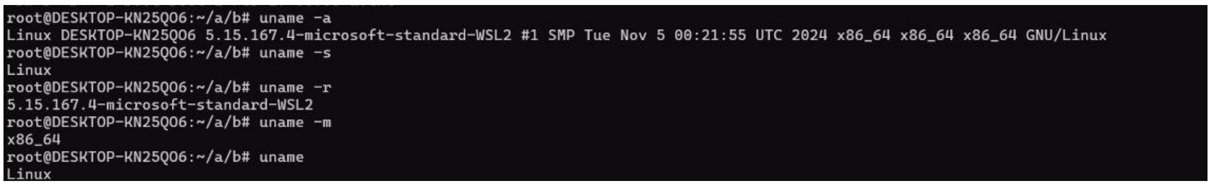
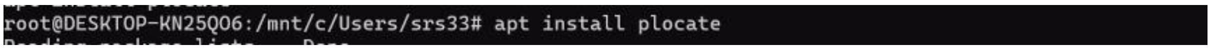
Linux lecture-2 (day-7):

if u want to know username, kernel , release date, hardware name

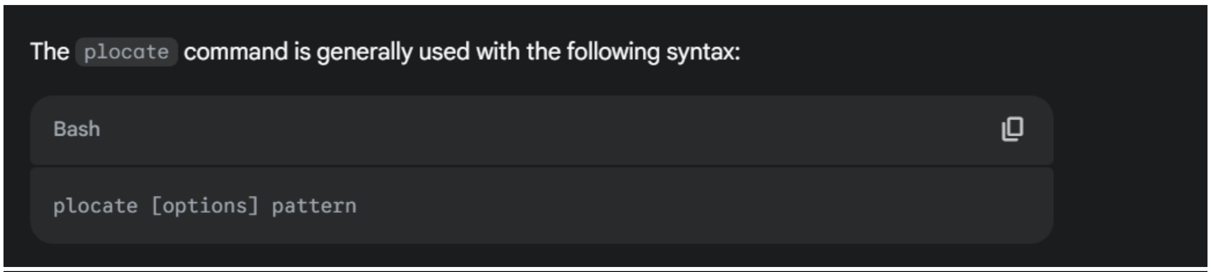


```
root@DESKTOP-KN25Q06:~/a/b# uname -a
Linux DESKTOP-KN25Q06 5.15.167.4-microsoft-standard-WSL2 #1 SMP Tue Nov 5 00:21:55 UTC 2024 x86_64 x86_64 x86_64 GNU/Linux
root@DESKTOP-KN25Q06:~/a/b# uname -s
Linux
root@DESKTOP-KN25Q06:~/a/b# uname -r
5.15.167.4-microsoft-standard-WSL2
root@DESKTOP-KN25Q06:~/a/b# uname -m
x86_64
root@DESKTOP-KN25Q06:~/a/b# uname
Linux
```

To search a file recursively we use locate command and file name, and if not installed then we need to install it:



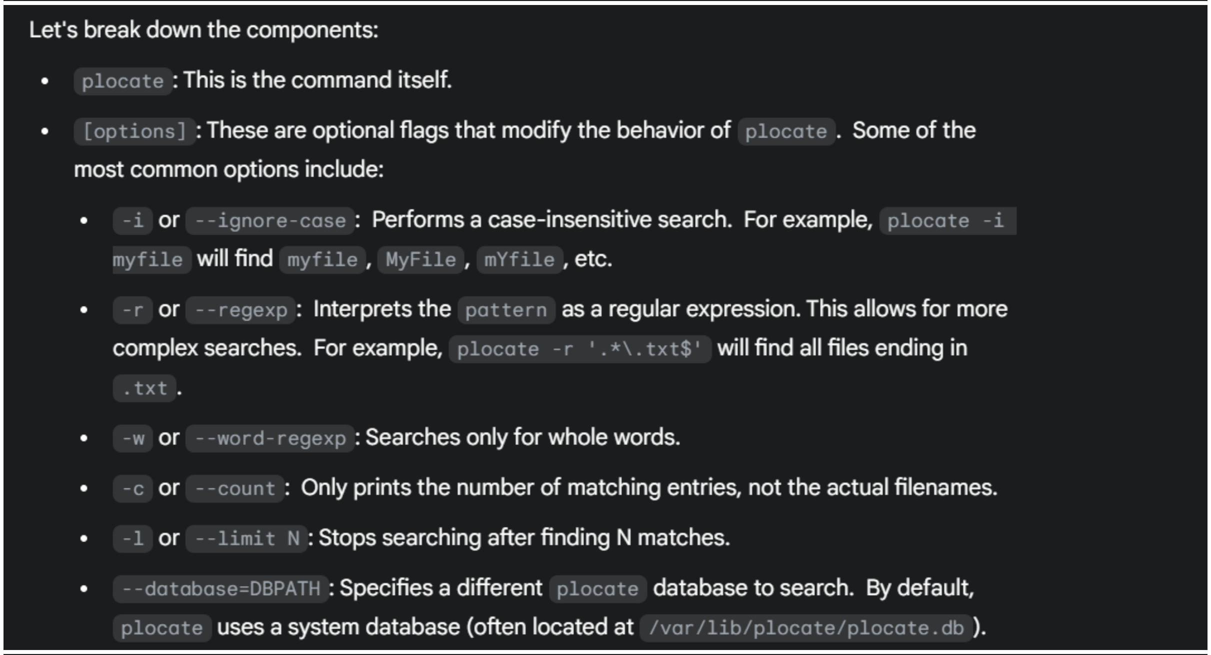
```
root@DESKTOP-KN25Q06:/mnt/c/Users/srs33# apt install plocate
```



The `plocate` command is generally used with the following syntax:

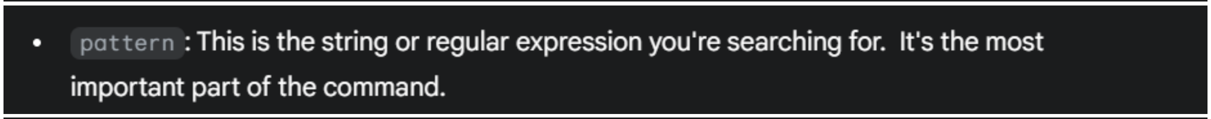
Bash

```
plocate [options] pattern
```

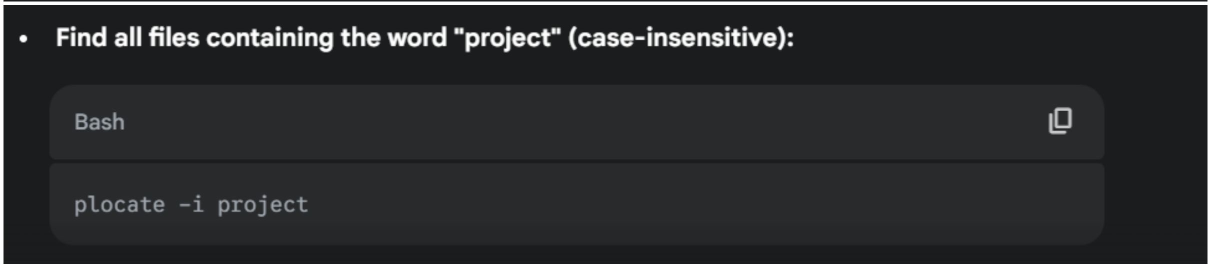


Let's break down the components:

- `plocate`: This is the command itself.
- `[options]`: These are optional flags that modify the behavior of `plocate`. Some of the most common options include:
  - `-i` or `--ignore-case`: Performs a case-insensitive search. For example, `plocate -i myfile` will find `myfile`, `MyFile`, `mYfile`, etc.
  - `-r` or `--regexp`: Interprets the `pattern` as a regular expression. This allows for more complex searches. For example, `plocate -r '.*\.txt$'` will find all files ending in `.txt`.
  - `-w` or `--word-regexp`: Searches only for whole words.
  - `-c` or `--count`: Only prints the number of matching entries, not the actual filenames.
  - `-l` or `--limit N`: Stops searching after finding N matches.
  - `--database=DBPATH`: Specifies a different `plocate` database to search. By default, `plocate` uses a system database (often located at `/var/lib/plocate/plocate.db`).



- `pattern`: This is the string or regular expression you're searching for. It's the most important part of the command.

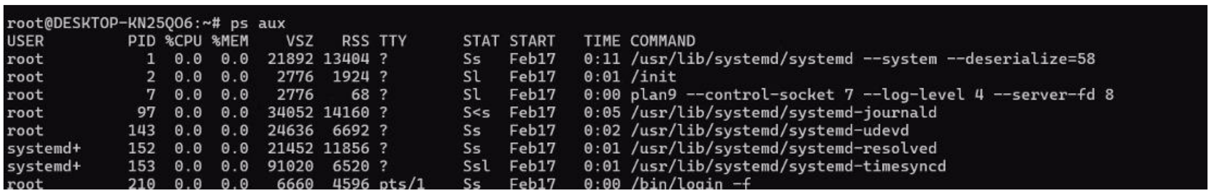


- **Find all files containing the word "project" (case-insensitive):**

Bash

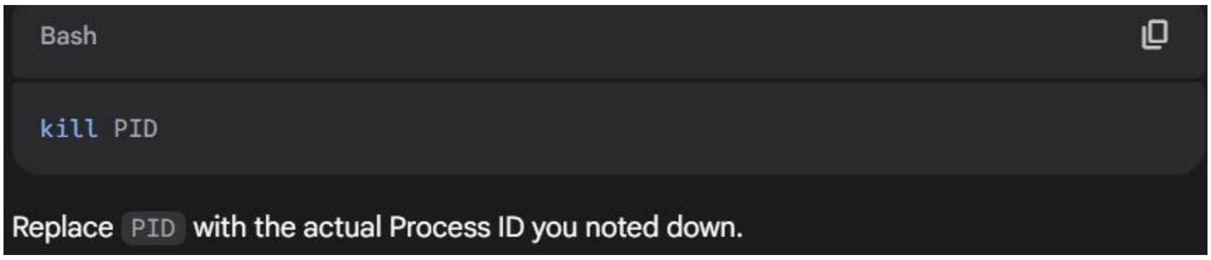
```
plocate -i project
```

To see all the process and the memory taken by them: ps aux



```
root@DESKTOP-KN25Q06:~# ps aux
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0  21892 13404 ?        Ss   Feb17   0:11 /usr/lib/systemd/systemd --system --deserialize=58
root           2  0.0  0.0   2776  1924 ?        Sl   Feb17   0:01 /init
root           7  0.0  0.0   2776    68 ?        Sl   Feb17   0:00 plan9 --control-socket 7 --log-level 4 --server-fd 8
root          97  0.0  0.0  34052 14160 ?        S<s  Feb17   0:05 /usr/lib/systemd/systemd-journald
root         143  0.0  0.0  24636  6692 ?        Ss   Feb17   0:02 /usr/lib/systemd/systemd-udevd
systemd+     152  0.0  0.0  21452 11856 ?        Ss   Feb17   0:01 /usr/lib/systemd/systemd-resolved
systemd+     153  0.0  0.0  91020  6520 ?        Ssl  Feb17   0:01 /usr/lib/systemd/systemd-timesyncd
root         210  0.0  0.0   6660  4596 pts/1    Ss   Feb17   0:00 /bin/login -f
```

And to kill a process we simply write: kill and process_id.

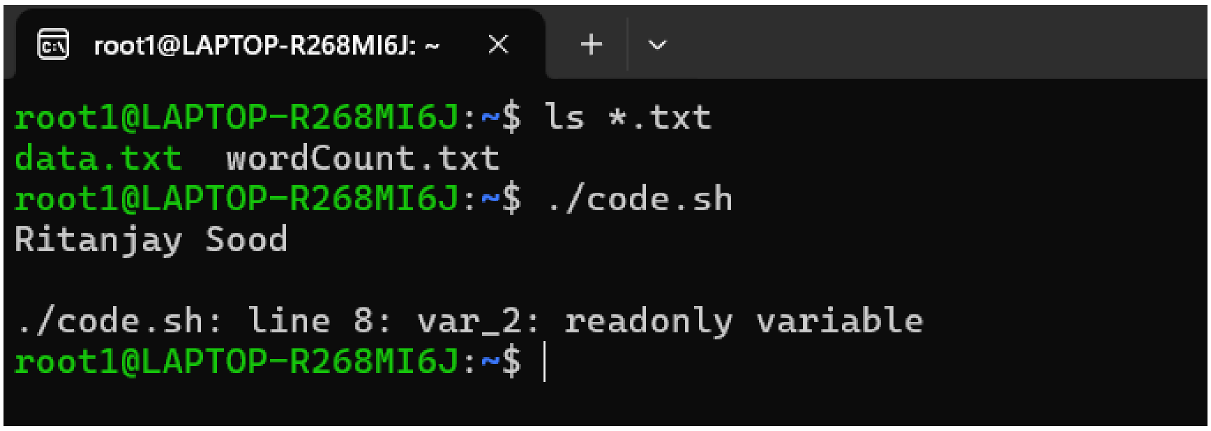


Bash

```bash
kill PID
```

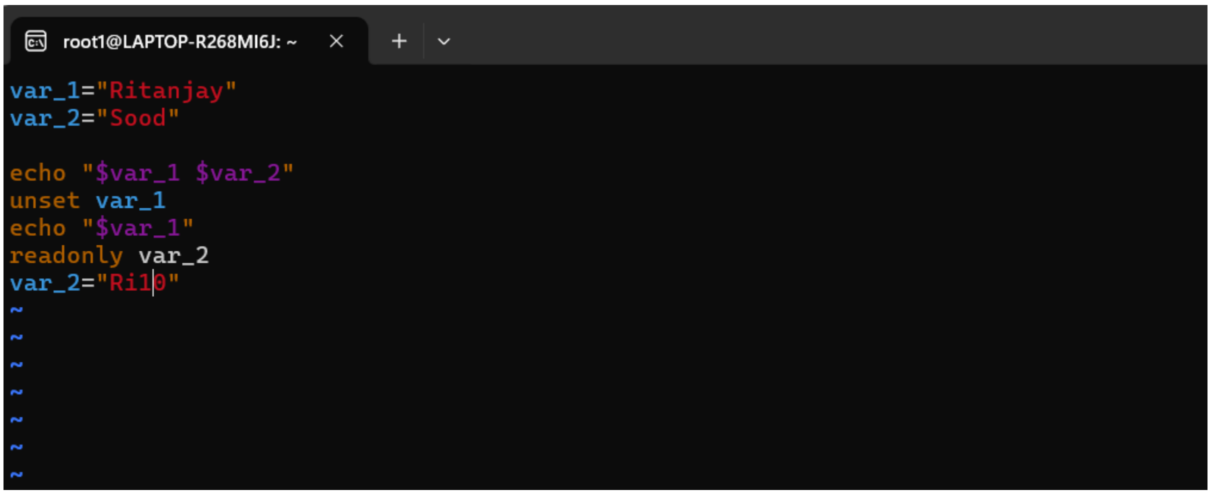
Replace `PID` with the actual Process ID you noted down.

To print a name: In shell scripting we use "echo command".



root1@LAPTOP-R268MI6J: ~

```
root1@LAPTOP-R268MI6J:~$ ls *.txt
data.txt  wordCount.txt
root1@LAPTOP-R268MI6J:~$ ./code.sh
Ritanjay Sood

./code.sh: line 8: var_2: readonly variable
root1@LAPTOP-R268MI6J:~$ 
```

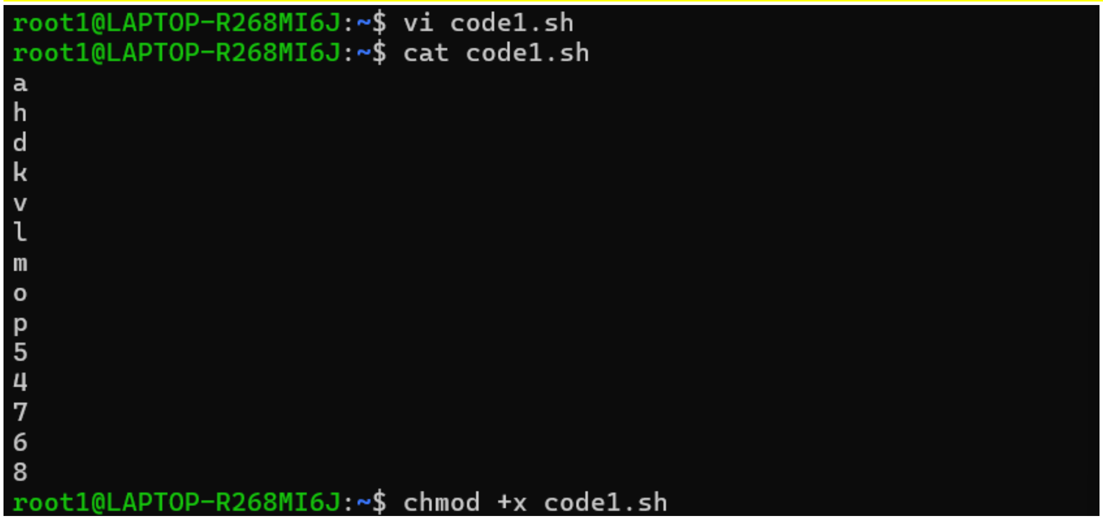


root1@LAPTOP-R268MI6J: ~

```
var_1="Ritanjay"
var_2="Sood"

echo "$var_1 $var_2"
unset var_1
echo "$var_1"
readonly var_2
var_2="Ri10"
~
~
~
~
~
~
~
```

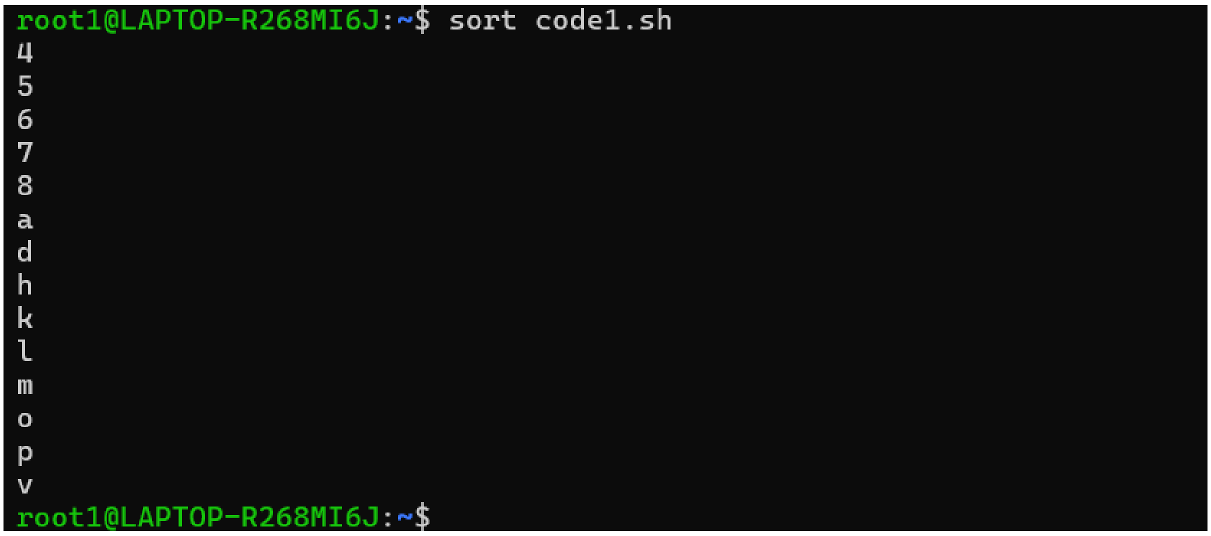
We use vim editor to edit the code.

Now if we want to sort file, then we simply write “sort <filename>”:



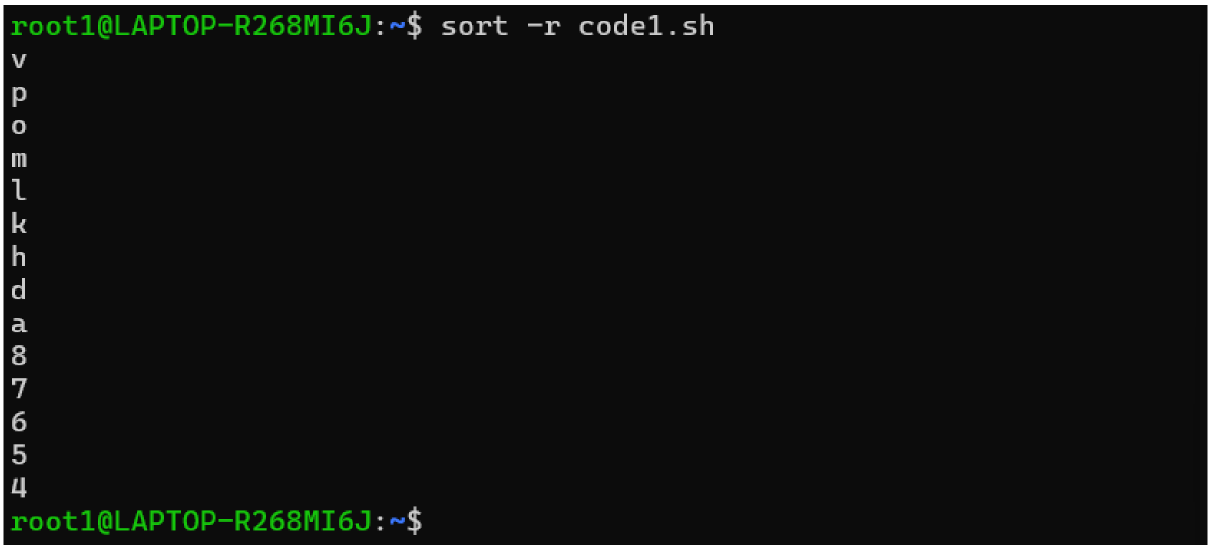
```
root1@LAPTOP-R268MI6J:~$ vi code1.sh
root1@LAPTOP-R268MI6J:~$ cat code1.sh
a
h
d
k
v
l
m
o
p
5
4
7
6
8
root1@LAPTOP-R268MI6J:~$ chmod +x code1.sh
```

Now performing sort operation:



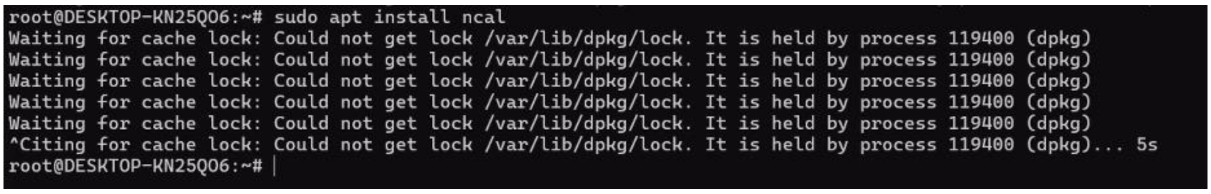
```
root1@LAPTOP-R268MI6J:~$ sort code1.sh
4
5
6
7
8
a
d
h
k
l
m
o
p
v
root1@LAPTOP-R268MI6J:~$
```

Now if you want to sort in reverse order:

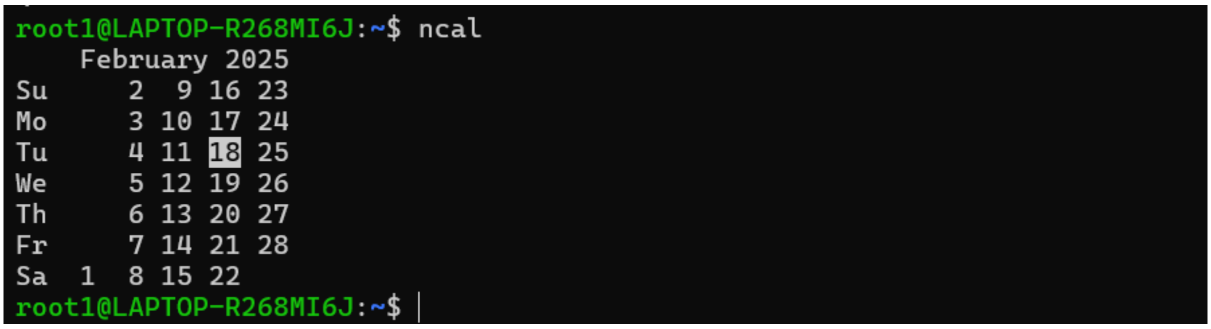


```
root1@LAPTOP-R268MI6J:~$ sort -r code1.sh
v
p
o
m
l
k
h
d
a
8
7
6
5
4
root1@LAPTOP-R268MI6J:~$
```

We can also use calendar: for that we need to install "ncal" or in some linux systems we can directly use "cal" for calendar.

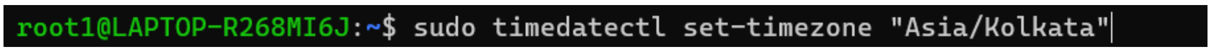


```
root@DESKTOP-KN25Q06:~# sudo apt install ncal
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock. It is held by process 119400 (dpkg)
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock. It is held by process 119400 (dpkg)
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock. It is held by process 119400 (dpkg)
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock. It is held by process 119400 (dpkg)
Waiting for cache lock: Could not get lock /var/lib/dpkg/lock. It is held by process 119400 (dpkg)
^Citing for cache lock: Could not get lock /var/lib/dpkg/lock. It is held by process 119400 (dpkg)... 5s
root@DESKTOP-KN25Q06:~# |
```



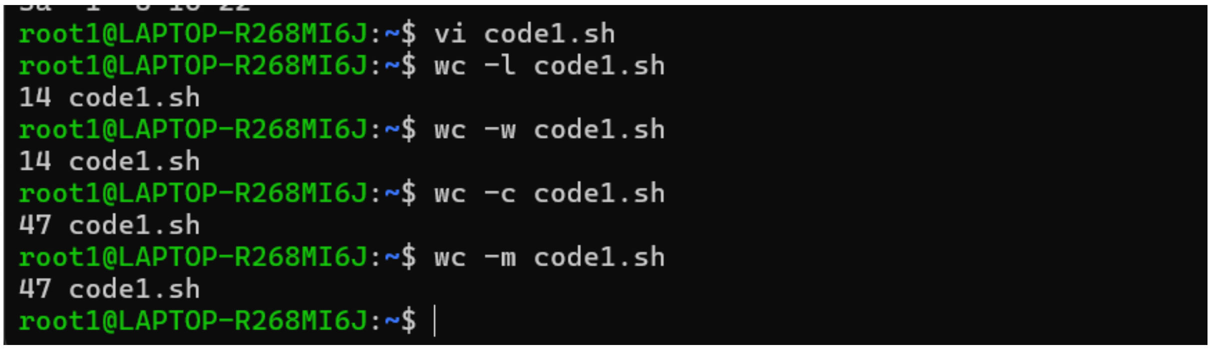
```
root1@LAPTOP-R268MI6J:~$ ncal
    February 2025
Su      2  9 16 23
Mo      3 10 17 24
Tu      4 11 18 25
We      5 12 19 26
Th      6 13 20 27
Fr      7 14 21 28
Sa   1  8 15 22
root1@LAPTOP-R268MI6J:~$ |
```

and we can even set the time/ zone according to our zone using:



```
root1@LAPTOP-R268MI6J:~$ sudo timedatectl set-timezone "Asia/Kolkata"|
```

If we want to count lines or words or letters of a document, then we use word count:



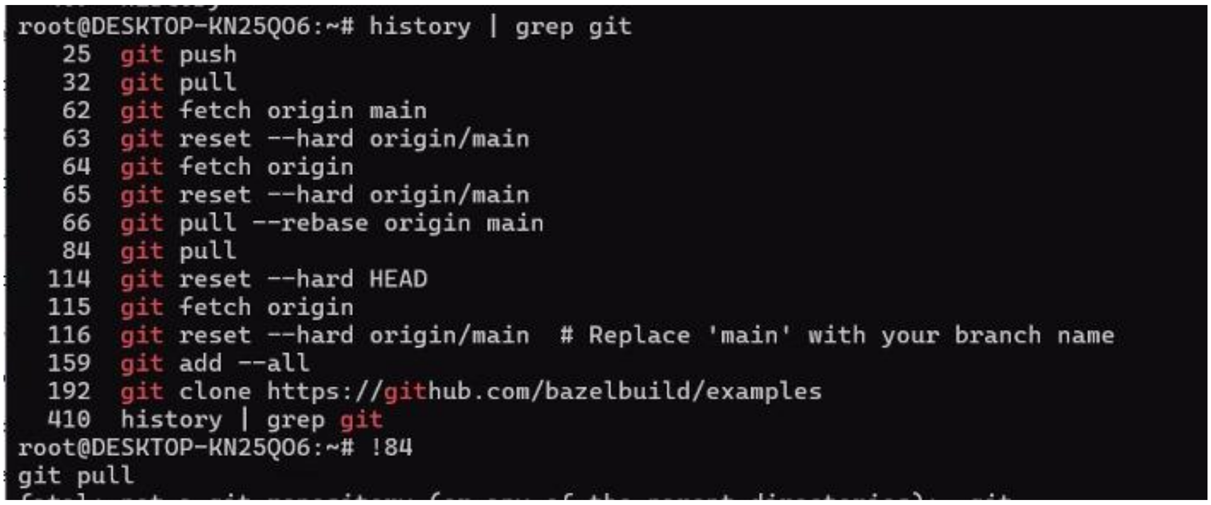
```
root1@LAPTOP-R268MI6J:~$ vi code1.sh
root1@LAPTOP-R268MI6J:~$ wc -l code1.sh
14 code1.sh
root1@LAPTOP-R268MI6J:~$ wc -w code1.sh
14 code1.sh
root1@LAPTOP-R268MI6J:~$ wc -c code1.sh
47 code1.sh
root1@LAPTOP-R268MI6J:~$ wc -m code1.sh
47 code1.sh
root1@LAPTOP-R268MI6J:~$ |
```

Here flag -l is used to count lines, -w for counting words, -c to count characters.

-c is used to count the bytes. And generally in normal English languages , the bytes count is equal to character count. But in some cases it fails. **So to accurately count the character we use -m command.**

- **Use -c when you need to know the number of bytes in a file.**
- **Use -m when you need to know the accurate number of characters, especially if you're working with text files that might contain multi-byte characters (like UTF-8 encoded files)**

Pipe is used when we want to join multiple commands. here output of one command is input for other command.

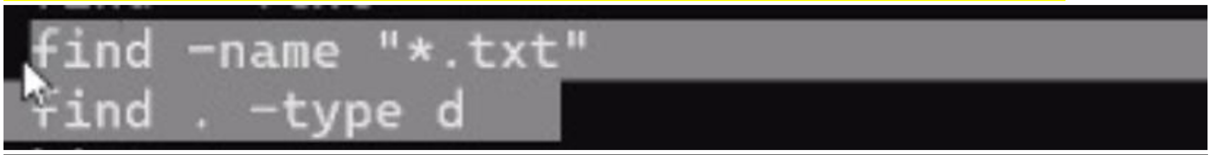


```
root@DESKTOP-KN25Q06:~# history | grep git
   25  git push
   32  git pull
   62  git fetch origin main
   63  git reset --hard origin/main
   64  git fetch origin
   65  git reset --hard origin/main
   66  git pull --rebase origin main
   84  git pull
  114  git reset --hard HEAD
  115  git fetch origin
  116  git reset --hard origin/main  # Replace 'main' with your branch name
  159  git add --all
  192  git clone https://github.com/bazelbuild/examples
  410  history | grep git
root@DESKTOP-KN25Q06:~# !84
git pull
```

Here we merged the history command and grep command, for searching all commands in history with "git" in it.

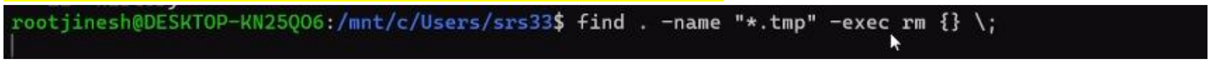
if u want to reuse one command from history, then just simply write ! operator and command number in history

if we want to find recursively, then we use "find command":



```
find -name "*.txt"
find . -type d
```

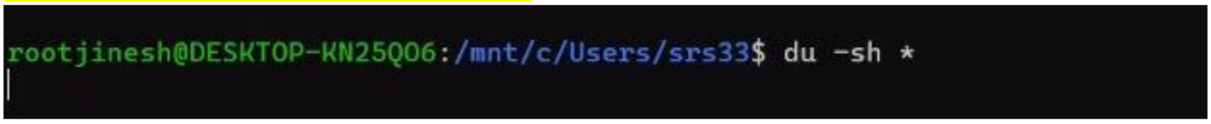
if we want to remove all temporary files:



```
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33$ find . -name "*.tmp" -exec rm {} \;
```

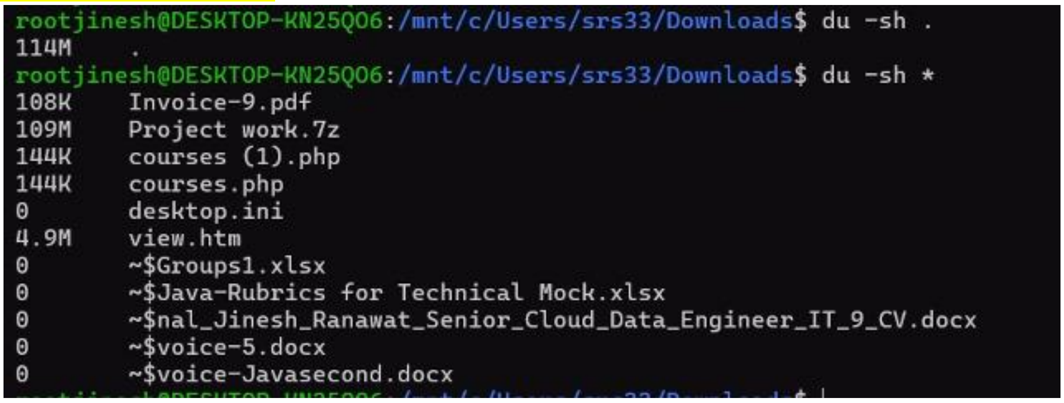
here we are finding all files in home directory with extension of tmp, and then we are using exec to execute to remove file and as there can be multiple files we would write { } displaying list or array. and then to end exec we write /.

To see highest consuming file:



```
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33$ du -sh *
```

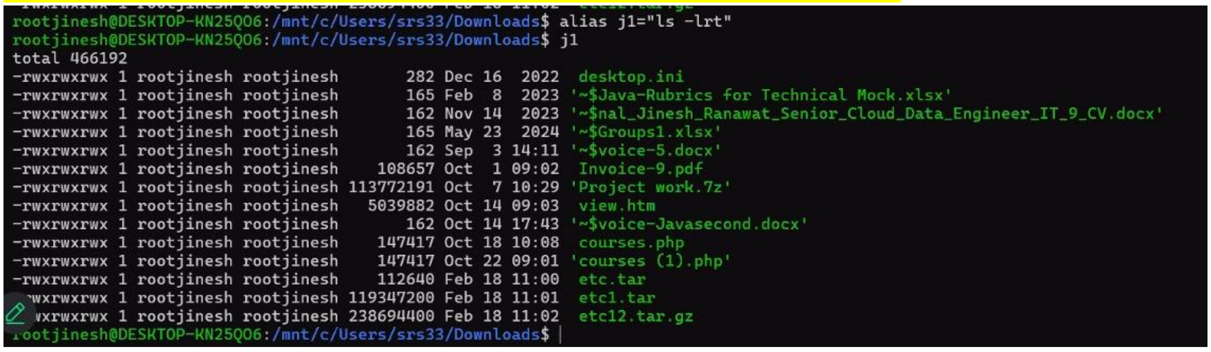
To see all process actively uses our system memory, if we want to see top activities:



```
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33/Downloads$ du -sh .
114M    .
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33/Downloads$ du -sh *
108K    Invoice-9.pdf
109M    Project work.7z
144K    courses (1).php
144K    courses.php
0       desktop.ini
4.9M    view.htm
0       ~$Groups1.xlsx
0       ~$Java-Rubrics for Technical Mock.xlsx
0       ~$nal_Jinesh_Ranawat_Senior_Cloud_Data_Engineer_IT_9_CV.docx
0       ~$voice-5.docx
0       ~$voice-Javasecond.docx
```

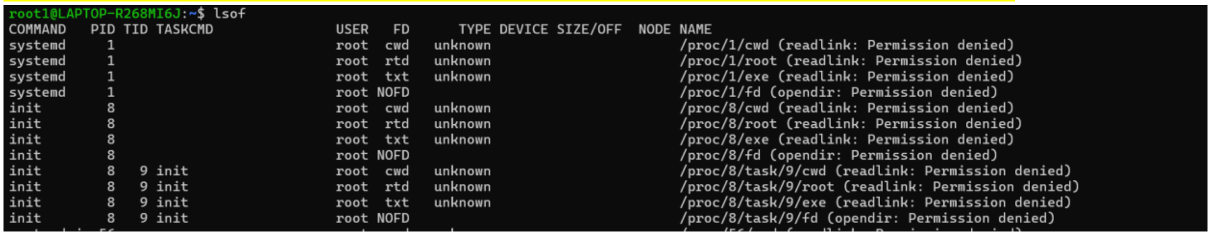
And ps -aux is static whereas top is real time, and if ny process comes with more requirement it would automatically show their.

we can also alias the commands in some variables:



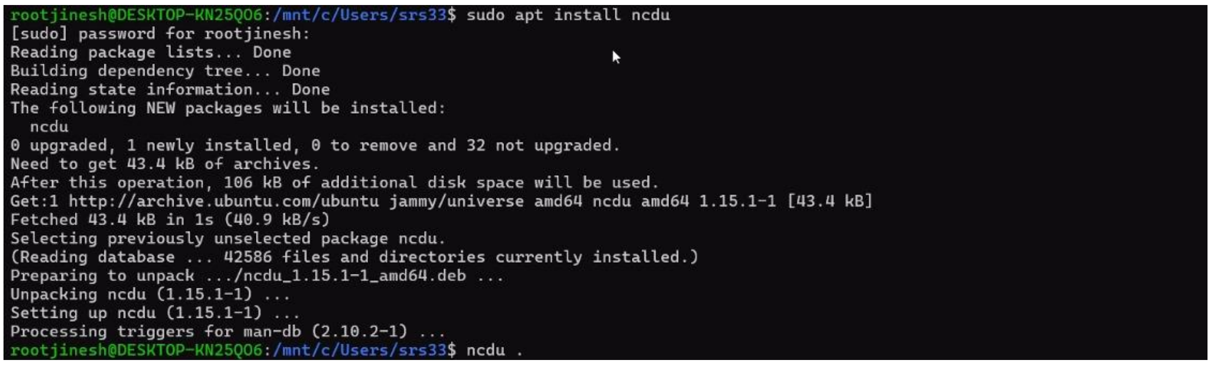
```
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33/Downloads$ alias j1="ls -lrt"
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33/Downloads$ j1
total 466192
-rwxrwxrwx 1 rootjinesh rootjinesh       282 Dec 16  2022  desktop.ini
-rwxrwxrwx 1 rootjinesh rootjinesh       165 Feb  8  2023 '~$Java-Rubrics for Technical Mock.xlsx'
-rwxrwxrwx 1 rootjinesh rootjinesh       162 Nov 14  2023 '~$nal_Jinesh_Ranawat_Senior_Cloud_Data_Engineer_IT_9_CV.docx'
-rwxrwxrwx 1 rootjinesh rootjinesh       165 May 23  2024 '~$Groups1.xlsx'
-rwxrwxrwx 1 rootjinesh rootjinesh       162 Sep  3 14:11 '~$voice-5.docx'
-rwxrwxrwx 1 rootjinesh rootjinesh    108657 Oct  1 09:02  Invoice-9.pdf
-rwxrwxrwx 1 rootjinesh rootjinesh 113772191 Oct  7 10:29 'Project work.7z'
-rwxrwxrwx 1 rootjinesh rootjinesh   5039882 Oct 14 09:03  view.htm
-rwxrwxrwx 1 rootjinesh rootjinesh       162 Oct 14 17:43 '~$voice-Javasecond.docx'
-rwxrwxrwx 1 rootjinesh rootjinesh    147417 Oct 18 10:08  courses.php
-rwxrwxrwx 1 rootjinesh rootjinesh    147417 Oct 22 09:01 'courses (1).php'
-rwxrwxrwx 1 rootjinesh rootjinesh    112640 Feb 18 11:00  etc.tar
 wxrwxrwx 1 rootjinesh rootjinesh 119347200 Feb 18 11:01  etc1.tar
 wxrwxrwx 1 rootjinesh rootjinesh 238694400 Feb 18 11:02  etc12.tar.gz
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33/Downloads$
```

To see all the opened files, we write list open files as lsof:



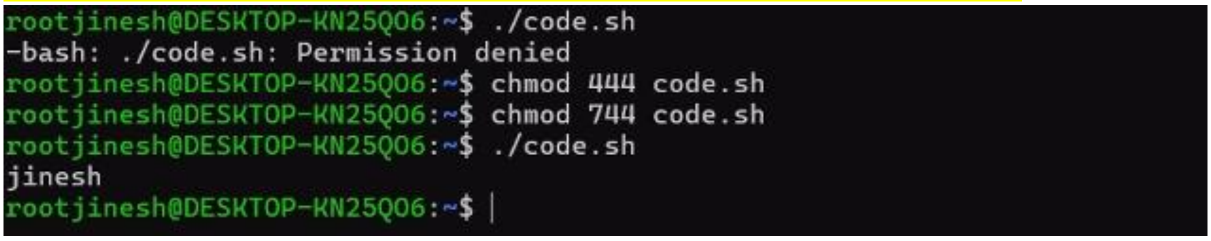
```
root1@LAPTOP-R268MI6J:~$ lsof
COMMAND    PID TID TASKCMD               USER   FD      TYPE DEVICE SIZE/OFF   NODE NAME
systemd      1                           root  cwd   unknown                        /proc/1/cwd (readlink: Permission denied)
systemd      1                           root  rtd   unknown                        /proc/1/root (readlink: Permission denied)
systemd      1                           root  txt   unknown                        /proc/1/exe (readlink: Permission denied)
systemd      1                           root NOFD                                  /proc/1/fd (opendir: Permission denied)
init         8                           root  cwd   unknown                        /proc/8/cwd (readlink: Permission denied)
init         8                           root  rtd   unknown                        /proc/8/root (readlink: Permission denied)
init         8                           root  txt   unknown                        /proc/8/exe (readlink: Permission denied)
init         8                           root NOFD                                  /proc/8/fd (opendir: Permission denied)
init         8   9 init                  root  cwd   unknown                        /proc/8/task/9/cwd (readlink: Permission denied)
init         8   9 init                  root  rtd   unknown                        /proc/8/task/9/root (readlink: Permission denied)
init         8   9 init                  root  txt   unknown                        /proc/8/task/9/exe (readlink: Permission denied)
init         8   9 init                  root NOFD                                  /proc/8/task/9/fd (opendir: Permission denied)
```

To see interactive disk usage: we use ndcu and we might need to install it if not done.



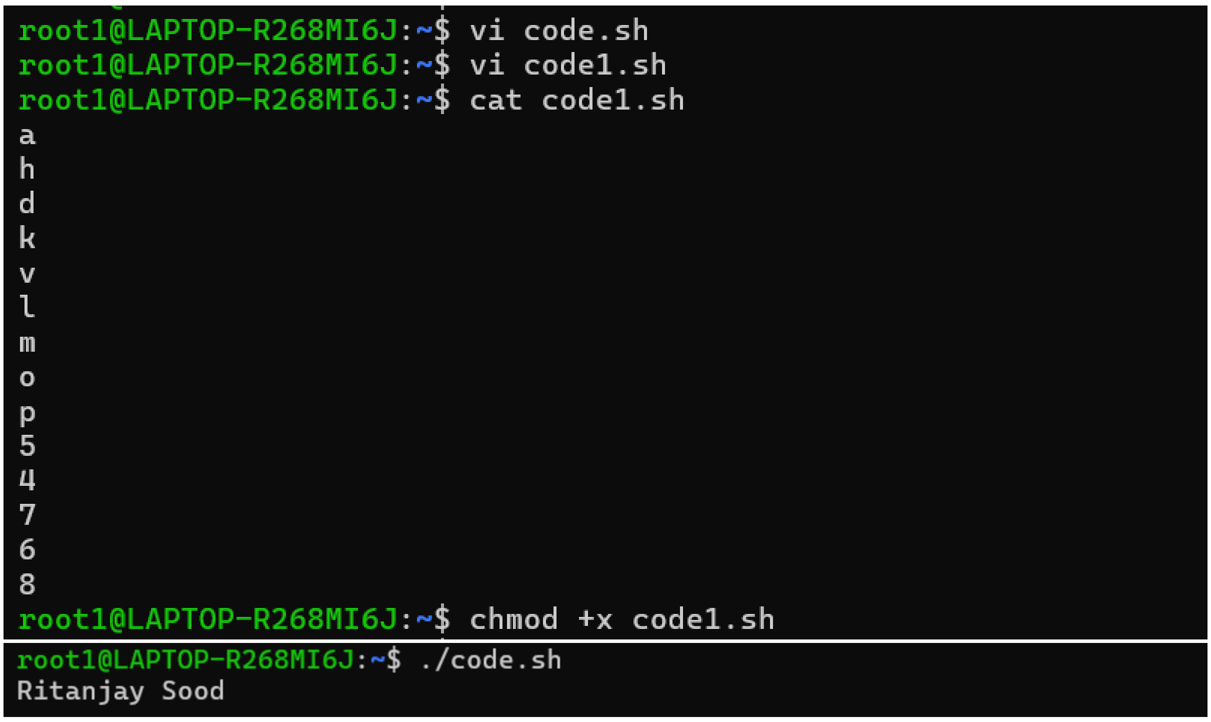
```
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33$ sudo apt install ncdu
[sudo] password for rootjinesh:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  ncdu
0 upgraded, 1 newly installed, 0 to remove and 32 not upgraded.
Need to get 43.4 kB of archives.
After this operation, 106 kB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 ncdu amd64 1.15.1-1 [43.4 kB]
Fetched 43.4 kB in 1s (40.9 kB/s)
Selecting previously unselected package ncdu.
(Reading database ... 42586 files and directories currently installed.)
Preparing to unpack .../ncdu_1.15.1-1_amd64.deb ...
Unpacking ncdu (1.15.1-1) ...
Setting up ncdu (1.15.1-1) ...
Processing triggers for man-db (2.10.2-1) ...
rootjinesh@DESKTOP-KN25Q06:/mnt/c/Users/srs33$ ncdu .
```

Procedure to create a file with .sh extension and to run it:



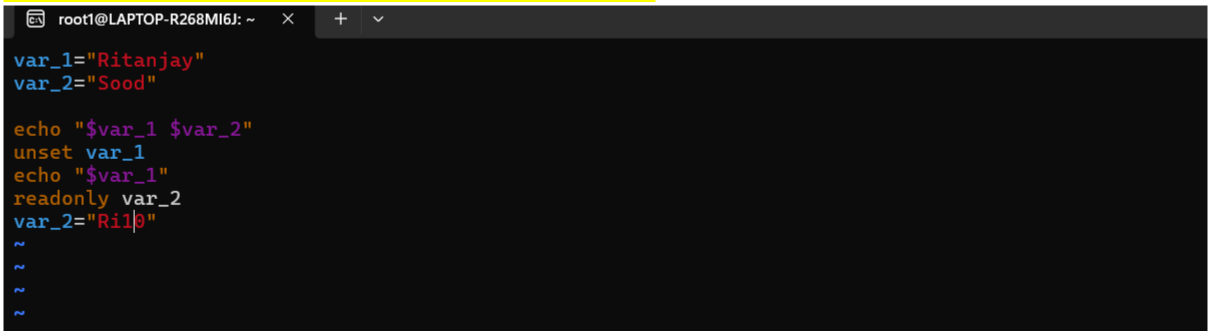
```
rootjinesh@DESKTOP-KN25Q06:~$ ./code.sh
-bash: ./code.sh: Permission denied
rootjinesh@DESKTOP-KN25Q06:~$ chmod 444 code.sh
rootjinesh@DESKTOP-KN25Q06:~$ chmod 744 code.sh
rootjinesh@DESKTOP-KN25Q06:~$ ./code.sh
jinesh
rootjinesh@DESKTOP-KN25Q06:~$ |
```

- Firstly we create a file with extension ".sh" using vi editor.
- And write respective script code in the file.
- And then we change the permission of that file so that we can execute the file.
- And then we simply run the file.

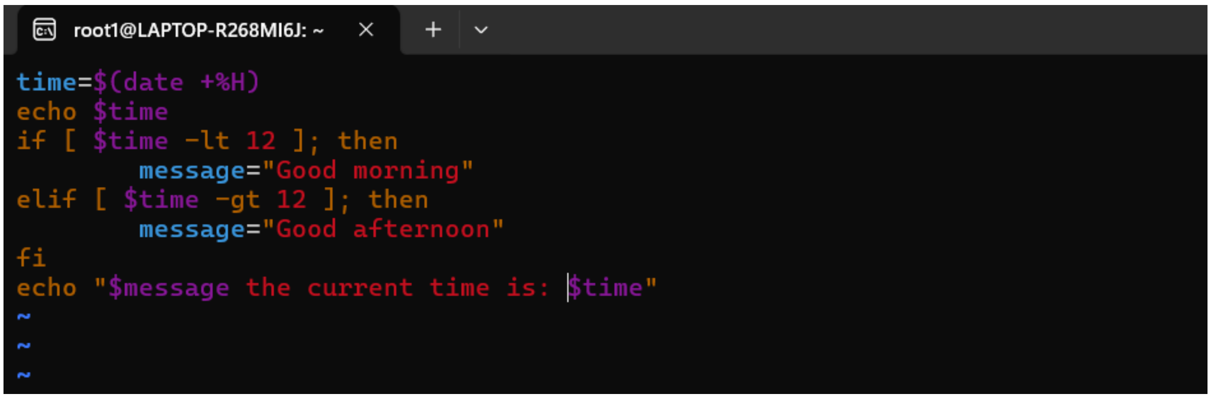


```
root1@LAPTOP-R268MI6J:~$ vi code.sh
root1@LAPTOP-R268MI6J:~$ vi code1.sh
root1@LAPTOP-R268MI6J:~$ cat code1.sh
a
h
d
k
v
l
m
o
p
5
4
7
6
8
root1@LAPTOP-R268MI6J:~$ chmod +x code1.sh
root1@LAPTOP-R268MI6J:~$ ./code.sh
Ritanjay Sood
```

Some coding syntaxes are as follows:



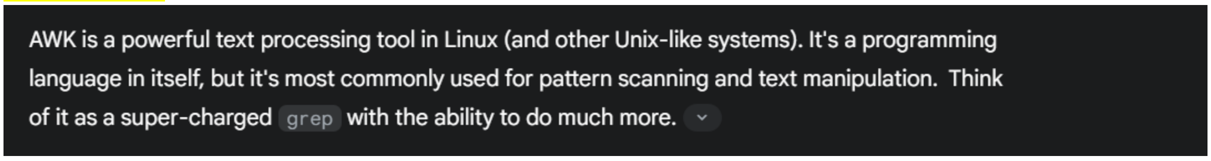
```
var_1="Ritanjay"
var_2="Sood"

echo "$var_1 $var_2"
unset var_1
echo "$var_1"
readonly var_2
var_2="Ri10"
~
~
~
~
```



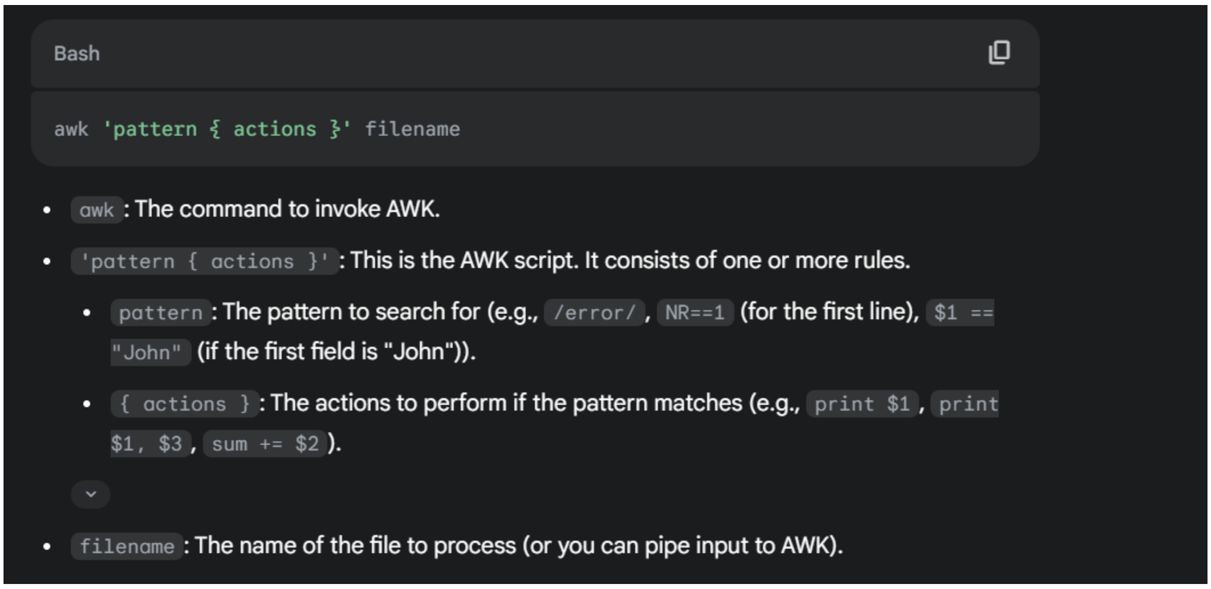
root1@LAPTOP-R268MI6J: ~

```
time=$(date +%H)
echo $time
if [ $time -lt 12 ]; then
        message="Good morning"
elif [ $time -gt 12 ]; then
        message="Good afternoon"
fi
echo "$message the current time is: $time"
~
~
~
```

AWK: awk cuts from file and returns us with results based on COLUMN



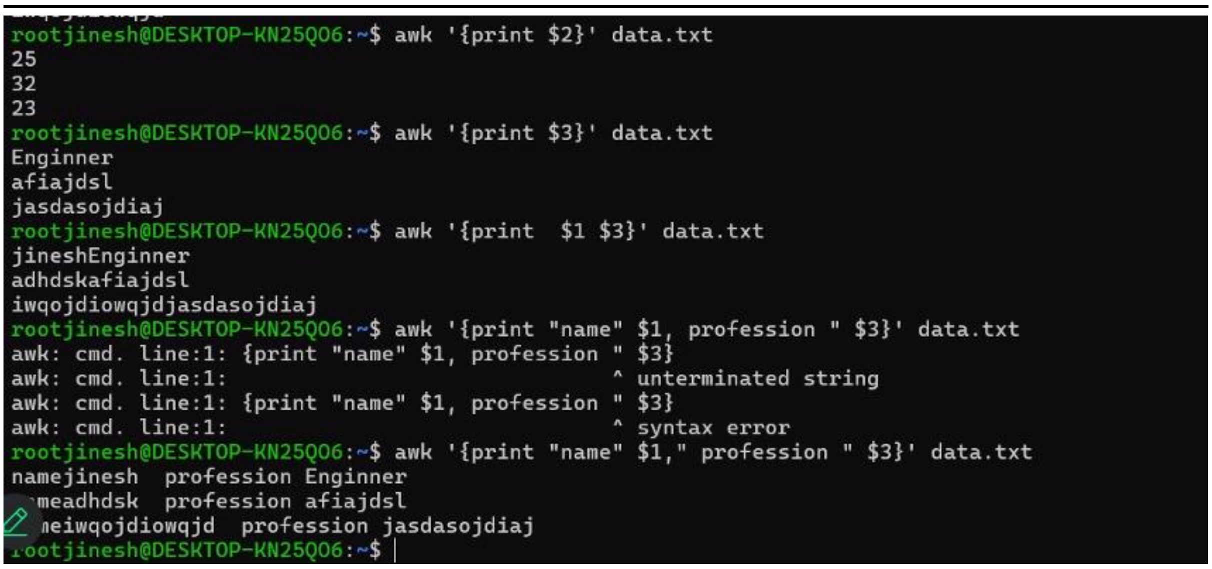
AWK is a powerful text processing tool in Linux (and other Unix-like systems). It's a programming language in itself, but it's most commonly used for pattern scanning and text manipulation. Think of it as a super-charged `grep` with the ability to do much more.



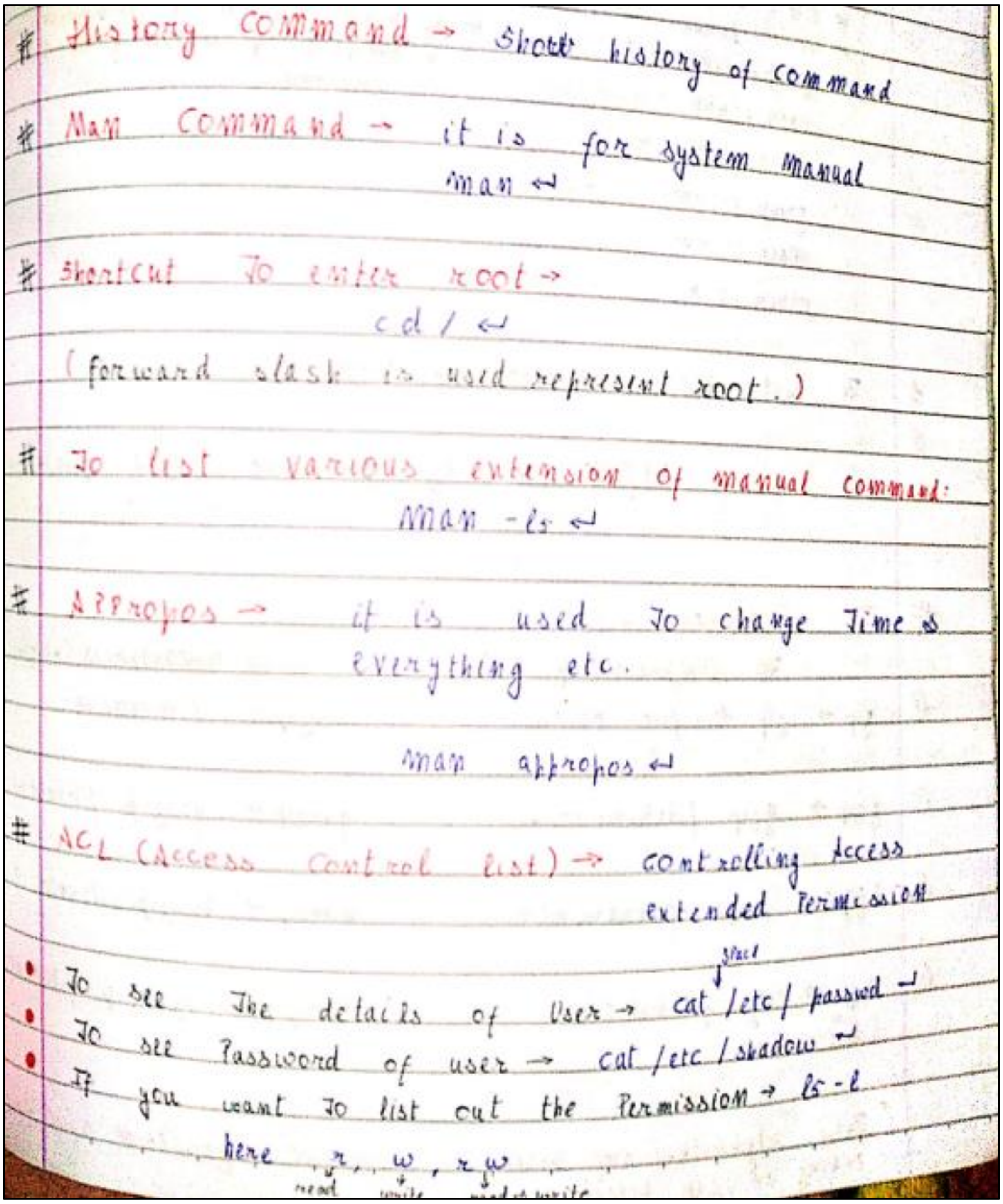
Bash

```bash
awk 'pattern { actions }' filename
```

- `awk`: The command to invoke AWK.
- `'pattern { actions }'`: This is the AWK script. It consists of one or more rules.
  - `pattern`: The pattern to search for (e.g., `/error/`, `NR==1` (for the first line), `$1 == "John"` (if the first field is "John")).
  - `{ actions }`: The actions to perform if the pattern matches (e.g., `print $1`, `print $1, $3`, `sum += $2`).
- `filename`: The name of the file to process (or you can pipe input to AWK).



```
rootjinesh@DESKTOP-KN25Q06:~$ awk '{print $2}' data.txt
25
32
23
rootjinesh@DESKTOP-KN25Q06:~$ awk '{print $3}' data.txt
Enginner
afiajdsl
jasdasojdiaj
rootjinesh@DESKTOP-KN25Q06:~$ awk '{print  $1 $3}' data.txt
jineshEnginner
adhdskafiajdsl
iwqojdiowqjdjasdasojdiaj
rootjinesh@DESKTOP-KN25Q06:~$ awk '{print "name" $1, profession " $3}' data.txt
awk: cmd. line:1: {print "name" $1, profession " $3}
awk: cmd. line:1:                               ^ unterminated string
awk: cmd. line:1: {print "name" $1, profession " $3}
awk: cmd. line:1:                               ^ syntax error
rootjinesh@DESKTOP-KN25Q06:~$ awk '{print "name" $1," profession " $3}' data.txt
namejinesh  profession Enginner
 meadhdsk  profession afiajdsl
 meiwqojdiowqjd  profession jasdasojdiaj
rootjinesh@DESKTOP-KN25Q06:~$ 
```



History command → Show history of command

Man Command → it is for system manual
man ↵

Shortcut To enter root →
cd / ↵
(forward slash is used represent root.)

To list various extension of manual command:
Man -ls ↵

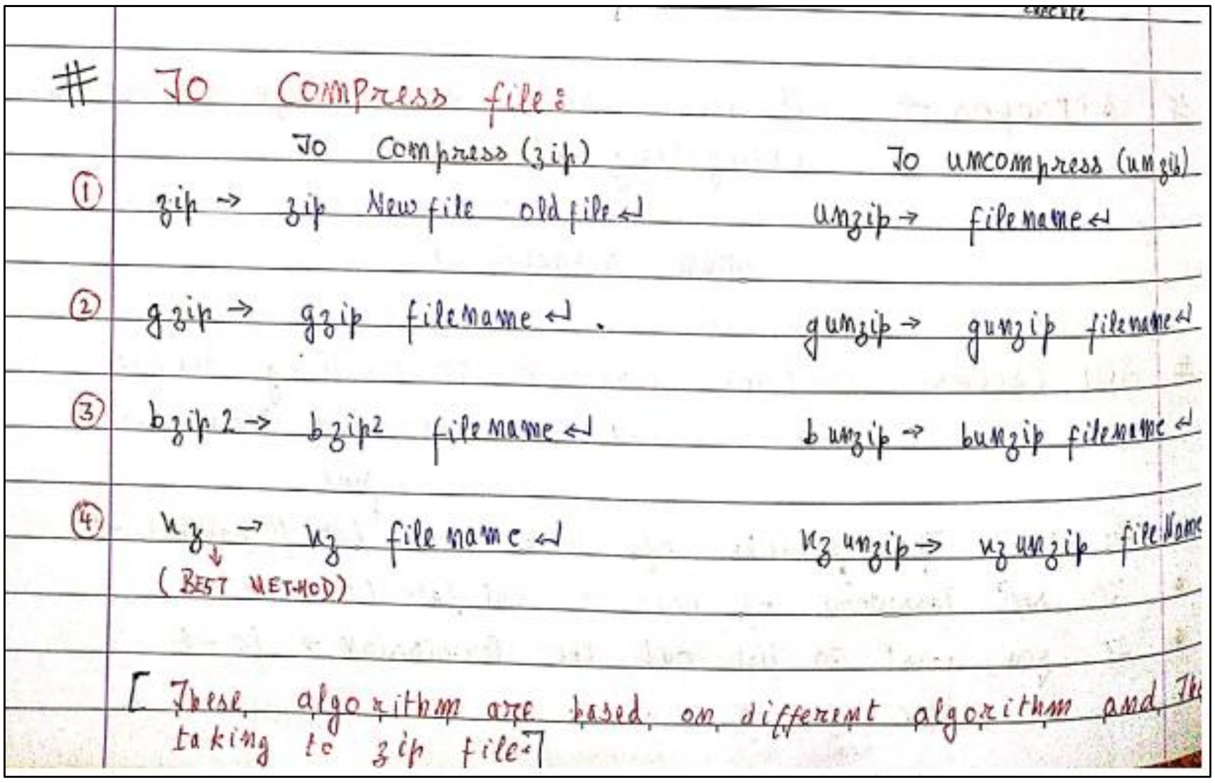
Appropos → it is used To change Time & everything etc.

man appropos ↵

ACL (Access Control list) → controlling Access extended Permission

- To see The details of User → cat /etc/ passwd ↵ (space)
- To see Password of user → cat /etc / shadow ↵
- If you want To list out the Permission → ls -l
  here r (read), w (write), rw (read & write)

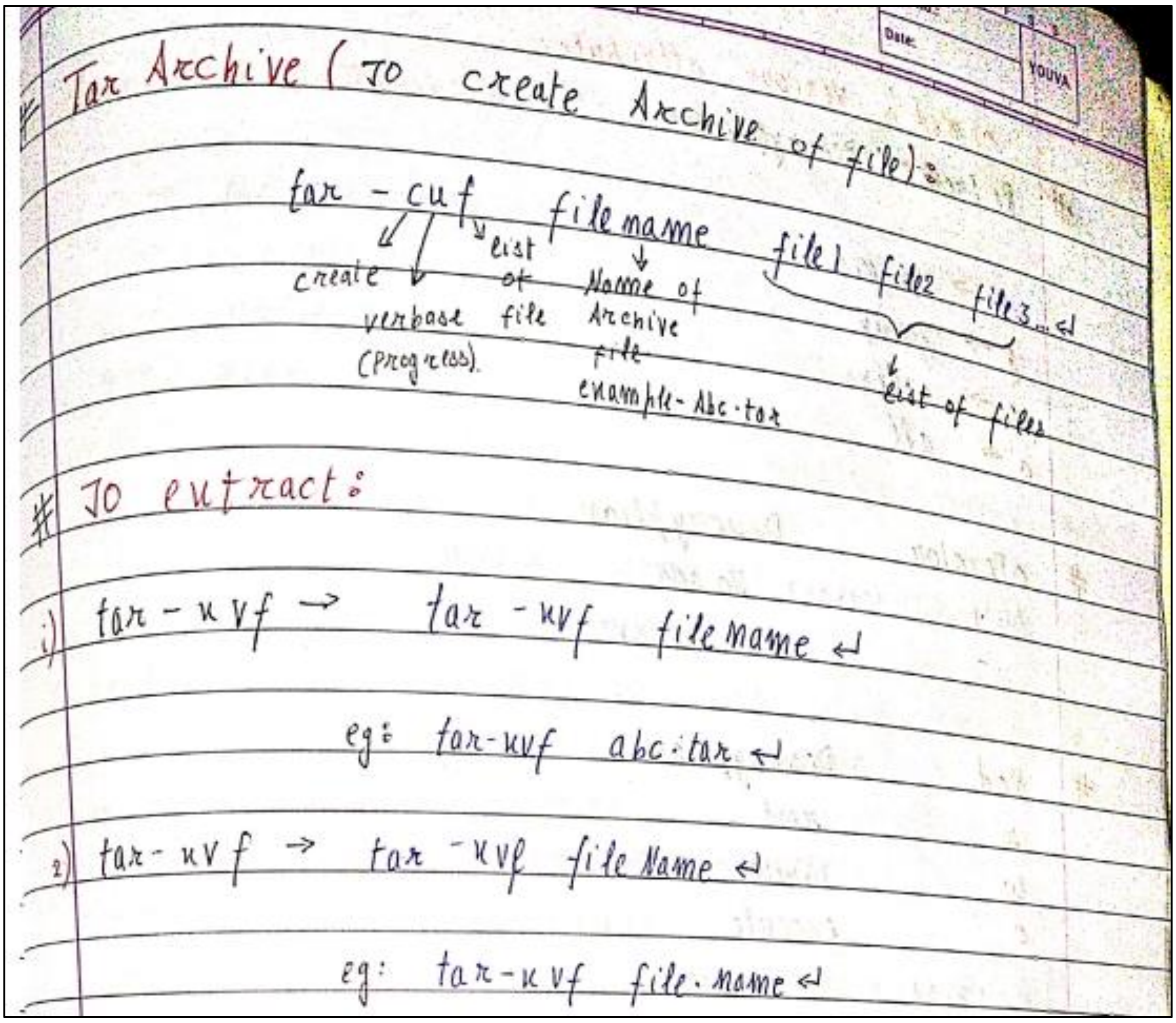
In man command if we want to toggle after search then we can simply press n to toggle through all that.



To Compress file:

	To Compress (zip)	To uncompress (unzip)
①	zip → zip New file old file ↵	unzip → filename ↵
②	gzip → gzip filename ↵.	gunzip → gunzip filename ↵
③	bzip2 → bzip2 filename ↵	bunzip → bunzip filename ↵
④	xz → xz file name ↵ (BEST METHOD)	xz unzip → xz unzip fileName

[These algorithm are based on different algorithm and The taking to zip file.]



Tar Archive (To create Archive of file):

$$\underbrace{\text{tar}}\ -\overset{}{c}vf\ \ \text{filename}\ \ \underbrace{\text{file1 file2 file3...}}_{\text{list of files}} \hookleftarrow$$

- c → create
- v → verbose (Progress).
- f → list of file
- filename → Name of Archive ~~file~~ example- Abc.tar
- file1 file2 file3... ↵ → list of files

To extract:

1) tar -xvf → tar -xvf filename ↵

   eg: tar-xvf abc.tar ↵

2) tar-xvf → tar -xvf file Name ↵

   eg: tar-xvf file.name ↵

To change owner of file & directory:

Chown →

1.) Change by name → SUDO Chown New user FileName ↵

2.) Change by id → SUDO Chown USERID FileName ↵

3.) Add new user → useradd main12 ↵
logout ↵

4.) if we want new owner for file vm.txt ↵

Syntax → SUDO Chown main 12 vm.txt ↵
Password: ↵

[NOTES: To check new owner getfacl fileName ↵]

LINUX Filters:

To achieve desired output from The files we apply filters.
filters means which is going To be filter.
There are many filters in unix/ Linux.

1) head → used To display The first top part of file (By default 10).
we can use '-n' To desired no. of line.

Syntax - $ head filename ↵

If we want Three lines.

for Multiple files:

```
$ head file1 file2 file3 ↵
```

[NOTE: retreiving lines from Multiple file isn't applicable On Tail Command.]

) tail →
It is used To display last Part of file (By default 10).
We can use '-n' to display desired no. of lines.

Syntax:
```
$ tail filename ↵
$ tail -n filename ↵
```

) Pipe (|): it is used To combine Two or more commands & in This The output of one file acts as input To The other file.

Syntax →
$$\underbrace{\text{head -n filename}}_{\substack{\text{The output of}\\ \text{This file would}\\ \text{be treated as}\\ \text{input of other}\\ \text{command.}}} \mid \text{tail -n}$$

eg: head - 8 abc.txt | tail - 3

) To add number as prefix in any file

```
# nl filename ↵
```

And if we want To save output in new file

$$\$ \text{ nl } \underbrace{b}_{\text{old filename}} > \underbrace{b2}_{\text{new file name}}$$

5) sort(): it is used To sort a files content arranging records in Particular orders.

Sorting Priority when sorting:

1) Numeric order
2) Alphabetical order
3) Lower case - To upper Case

# Syntax:

```
$ sort filename ↵
```

★ To sort Multiple files:

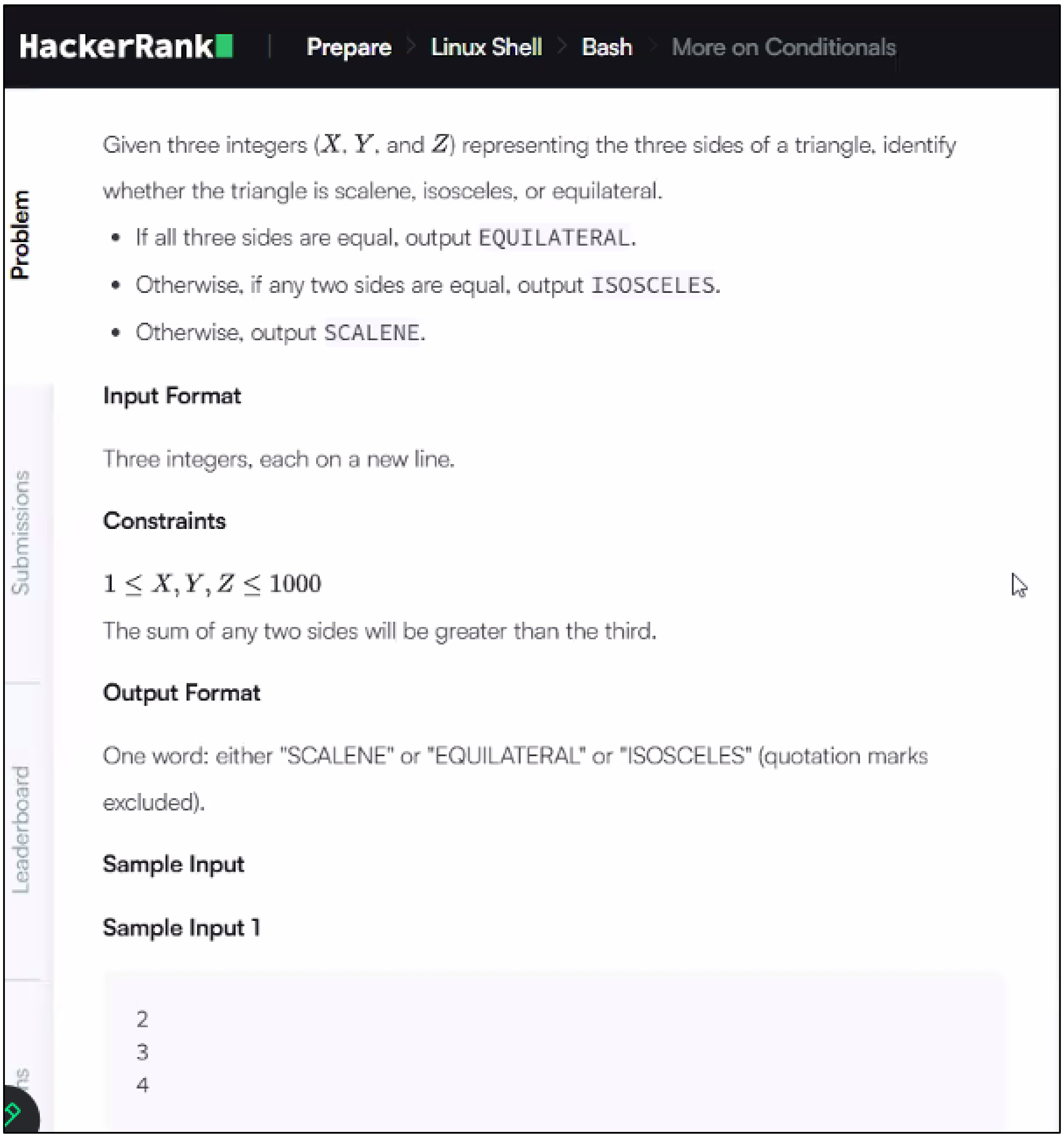
```
$ sort file1 file2 file3 ↵
```

eg: $ Sort ab (eg with values as
av
bg1
2
3
4
9
6
8
0
7)

```
output: 0
        2
        3
        4
        6
        8
        9
        ab
        as
        av
        ds
        bg1
```

6) To reverse content → $ sort -r filename ↵

7) To sort file numerically → Asscending order → $ sort -n filename ↵
Descending order → $ sort -rn filename ↵
or
$ sort -r filename ↵



Given three integers (X, Y, and Z) representing the three sides of a triangle, identify whether the triangle is scalene, isosceles, or equilateral.

- If all three sides are equal, output `EQUILATERAL`.
- Otherwise, if any two sides are equal, output `ISOSCELES`.
- Otherwise, output `SCALENE`.

Input Format

Three integers, each on a new line.

Constraints

$1 \leq X, Y, Z \leq 1000$

The sum of any two sides will be greater than the third.

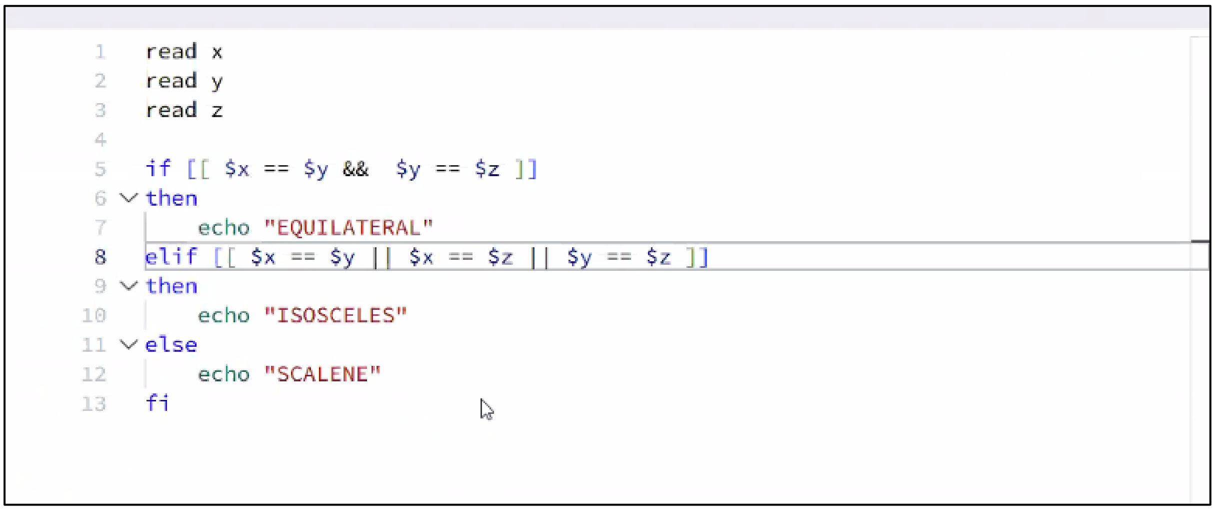
Output Format

One word: either "SCALENE" or "EQUILATERAL" or "ISOSCELES" (quotation marks excluded).

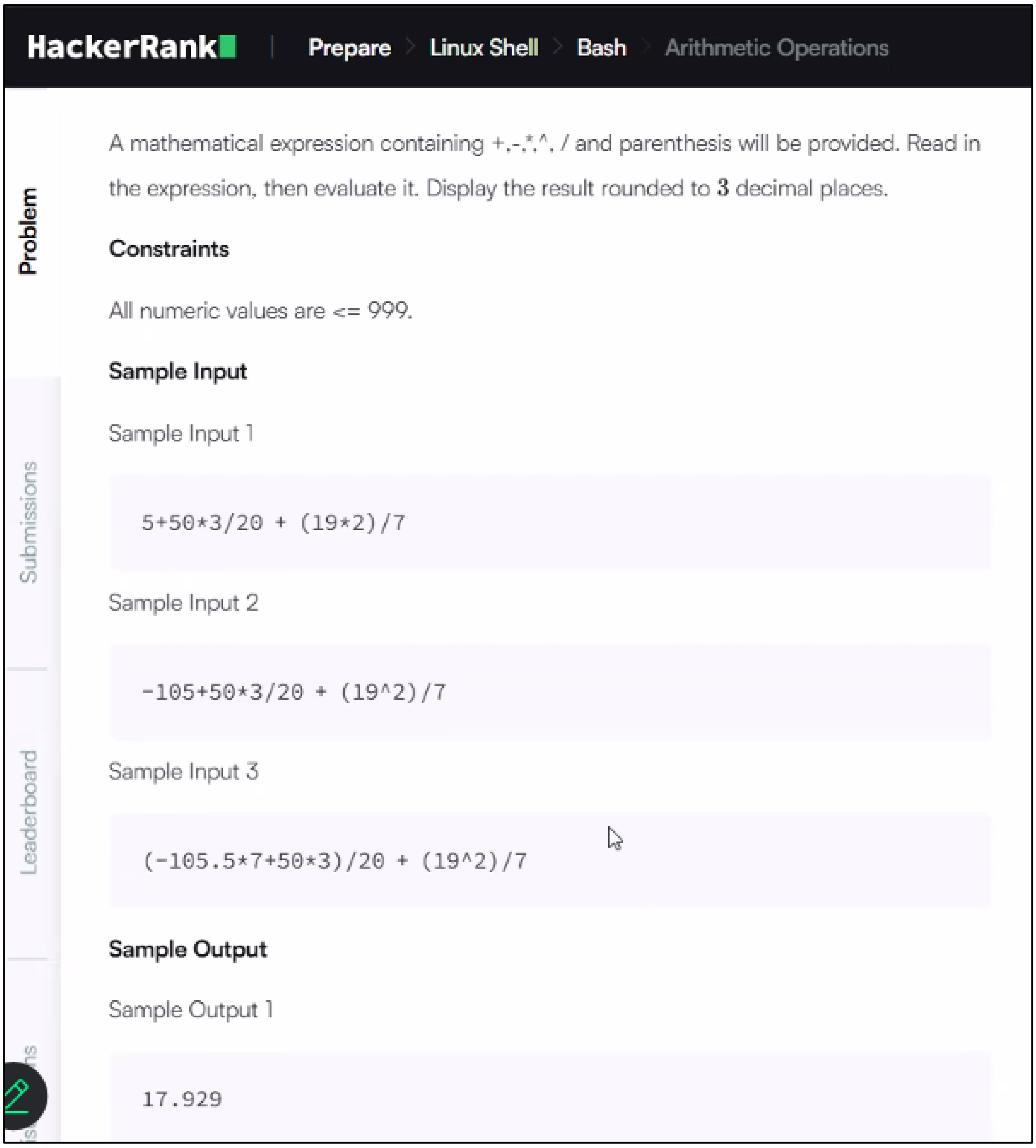
Sample Input

Sample Input 1

```
2
3
4
```



```
read x
read y
read z

if [[ $x == $y &&  $y == $z ]]
then
    echo "EQUILATERAL"
elif [[ $x == $y || $x == $z || $y == $z ]]
then
    echo "ISOSCELES"
else
    echo "SCALENE"
fi
```



A mathematical expression containing +,-,*,^, / and parenthesis will be provided. Read in the expression, then evaluate it. Display the result rounded to 3 decimal places.

Constraints

All numeric values are <= 999.

Sample Input

Sample Input 1

```
5+50*3/20 + (19*2)/7
```

Sample Input 2

```
-105+50*3/20 + (19^2)/7
```

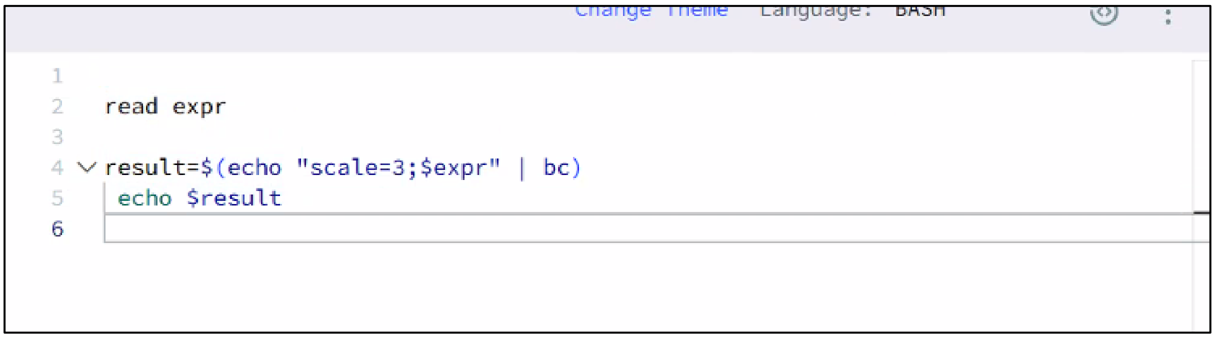
Sample Input 3

```
(-105.5*7+50*3)/20 + (19^2)/7
```

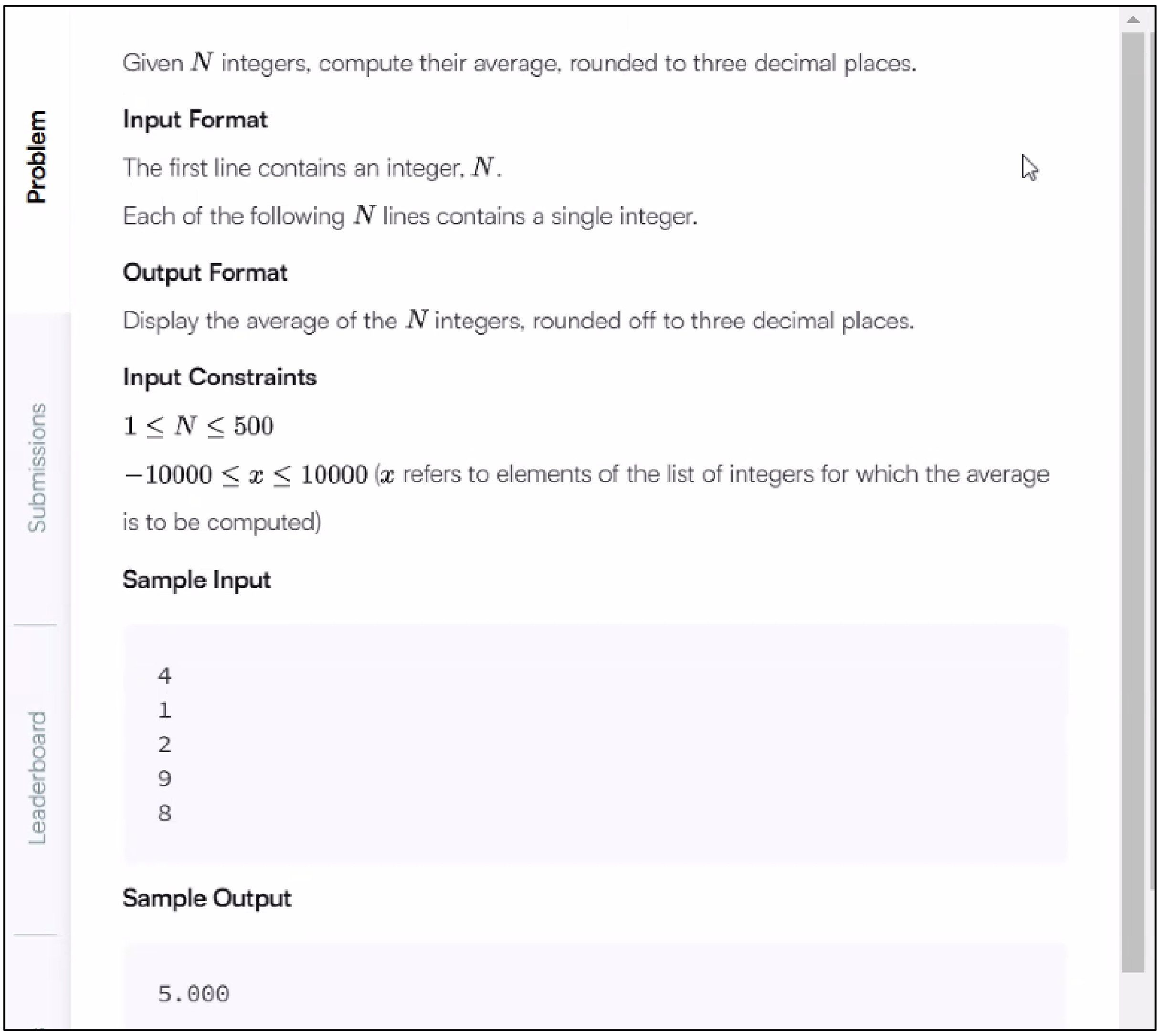
Sample Output

Sample Output 1

```
17.929
```



```

read expr

result=$(echo "scale=3;$expr" | bc)
 echo $result

```



Given N integers, compute their average, rounded to three decimal places.

Input Format

The first line contains an integer, N.

Each of the following N lines contains a single integer.

Output Format

Display the average of the N integers, rounded off to three decimal places.

Input Constraints

$1 \leq N \leq 500$

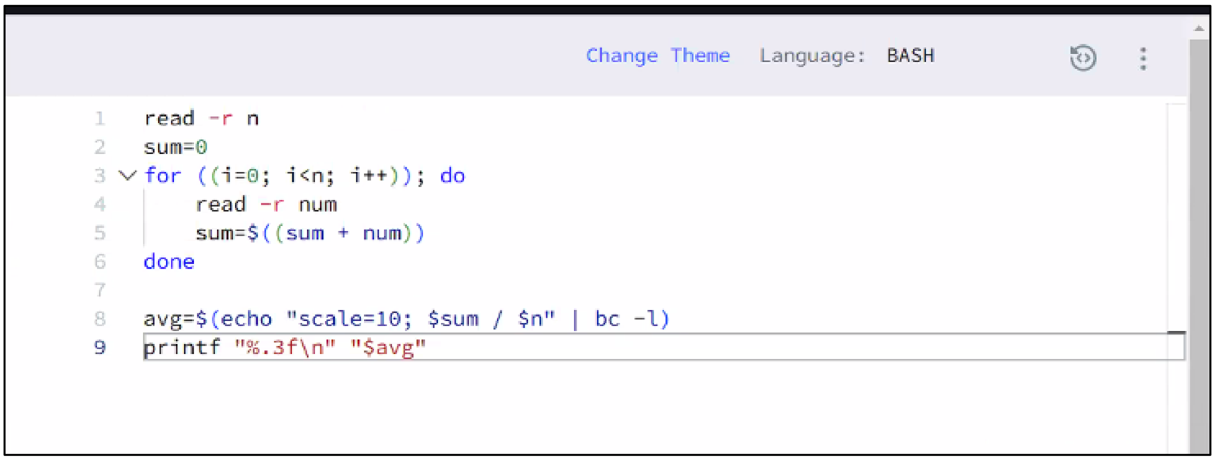
$-10000 \leq x \leq 10000$ (x refers to elements of the list of integers for which the average is to be computed)

Sample Input

```
4
1
2
9
8
```

Sample Output

```
5.000
```



```
read -r n
sum=0
for ((i=0; i<n; i++)); do
    read -r num
    sum=$((sum + num))
done

avg=$(echo "scale=10; $sum / $n" | bc -l)
printf "%.3f\n" "$avg"
```