NAME: VARSHA D KULKARNI SRN: PES1UG19EC339

OS-LAB-WEEK-1

1)Basic LINUX commands (ANY 10) executed in the lab should be submitted in the following way:

Command: What does the command do?

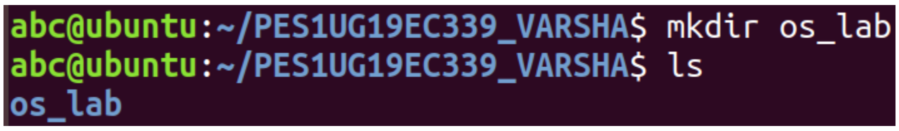
Any two options (i.e flags or arguments) regarding the command

Outcome of the command

1. mkdir

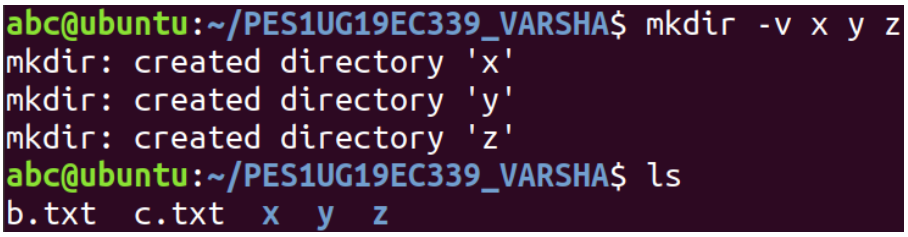
Creates a new directory.

Syntax: mkdir [flags] [Directories]



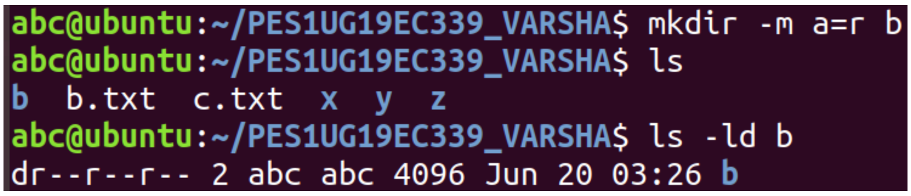
```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ mkdir os_lab
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
os_lab
```

- mkdir -v x y z (creates 3 directories(x,y and z) at a time)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ mkdir -v x y z
mkdir: created directory 'x'
mkdir: created directory 'y'
mkdir: created directory 'z'
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
b.txt  c.txt  x  y  z
```

- mkdir -m a=r b (Set file modes,i.e permissions,etc for the created directories)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ mkdir -m a=r b
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
b  b.txt  c.txt  x  y  z
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls -ld b
dr--r--r-- 2 abc abc 4096 Jun 20 03:26 b
```

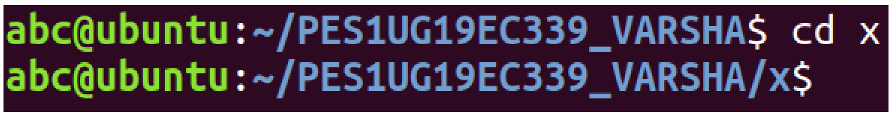
Here ‘a=r’ will allow users only to read from the directories.

2. cd

Change directory

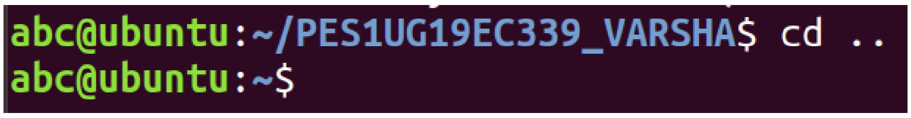
Syntax: cd [directories]

Ex: cd x (change directory to x)



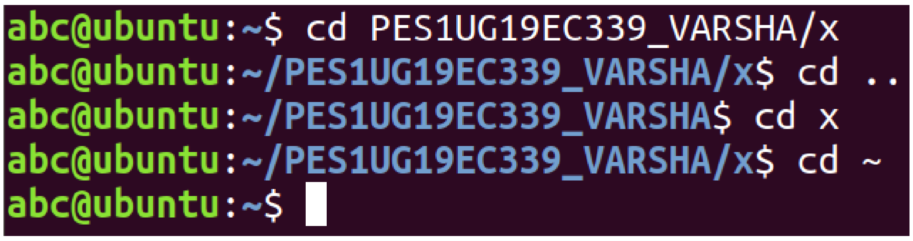
```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cd x
abc@ubuntu:~/PES1UG19EC339_VARSHA/x$
```

- cd .. (Comes out from current directory)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cd ..
abc@ubuntu:~$
```

- cd ~ (Change directory to the home directory)



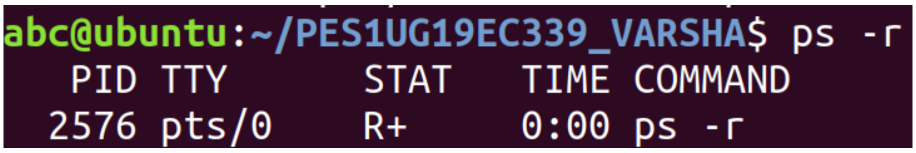
```
abc@ubuntu:~$ cd PES1UG19EC339_VARSHA/x
abc@ubuntu:~/PES1UG19EC339_VARSHA/x$ cd ..
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cd x
abc@ubuntu:~/PES1UG19EC339_VARSHA/x$ cd ~
abc@ubuntu:~$
```

3. <u>ps</u>

It is used for viewing information about processes on a system which stands for 'process status'

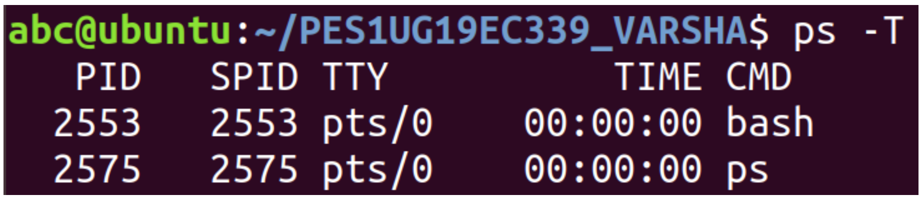
Syntax: ps [option]

- ps -r (View all the running processes)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ps -r
   PID TTY      STAT   TIME COMMAND
  2576 pts/0    R+     0:00 ps -r
```

- ps -T (View all the processes associated with this terminal)



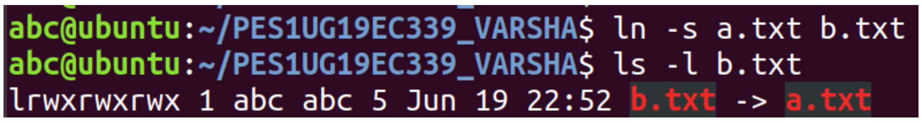
```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ps -T
   PID   SPID TTY          TIME CMD
  2553   2553 pts/0    00:00:00 bash
  2575   2575 pts/0    00:00:00 ps
```

4. <u>ln</u>

This command is used to create a symbolic link between the files

Syntax: ln [flag] file_name linkfile_name

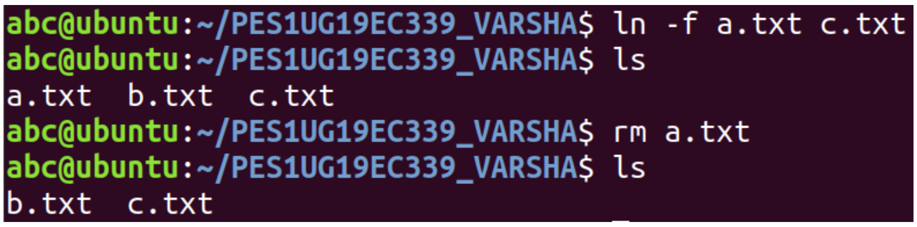
- ln -s a.txt b.txt



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ln -s a.txt b.txt
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls -l b.txt
lrwxrwxrwx 1 abc abc 5 Jun 19 22:52 b.txt -> a.txt
```

A symbolic/soft link is created between a.txt and b.txt which means, if a.txt is removed b.txt becomes worthless.

- ln -f a.txt b.txt



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ln -f a.txt c.txt
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
a.txt  b.txt  c.txt
abc@ubuntu:~/PES1UG19EC339_VARSHA$ rm a.txt
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
b.txt  c.txt
```

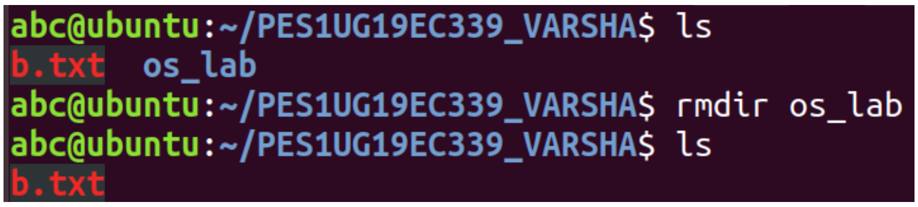
A hard link is created between a.txt and c.txt which means, if a.txt is removed b.txt is still accessible

5. <u>rmdir</u>

This Command is used to delete existing directory

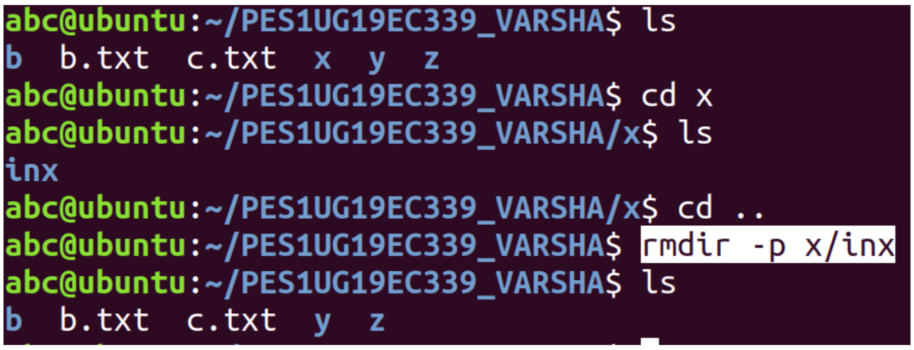
Syntax: rmdir dir_name

- rmdir os_lab (os_lab is removed from pwd)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
b.txt  os_lab
abc@ubuntu:~/PES1UG19EC339_VARSHA$ rmdir os_lab
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
b.txt
```

- rmdir -p x/inx (This will first remove child directory and then the parent directory)

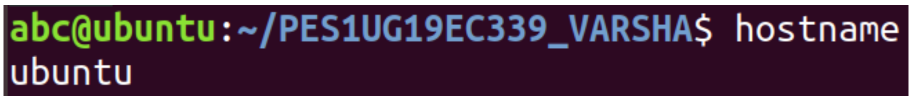


```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
b  b.txt  c.txt  x  y  z
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cd x
abc@ubuntu:~/PES1UG19EC339_VARSHA/x$ ls
inx
abc@ubuntu:~/PES1UG19EC339_VARSHA/x$ cd ..
abc@ubuntu:~/PES1UG19EC339_VARSHA$ rmdir -p x/inx
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
b  b.txt  c.txt  y  z
```

6. <u>hostname</u>

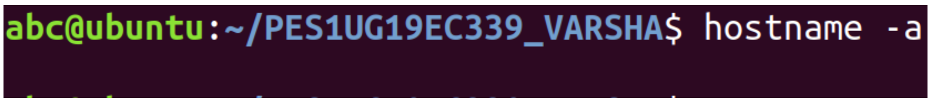
It is used to get the system hostname

Syntax: hostname –[option]



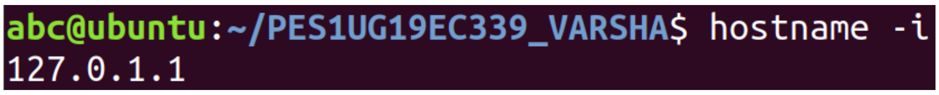
```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ hostname
ubuntu
```

- hostname -a (Gives alias name of name of the host system)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ hostname -a
```

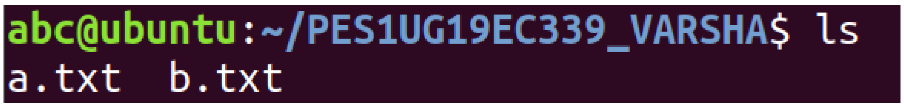
- hostname -i (Used to get IP addresses)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ hostname -i
127.0.1.1
```

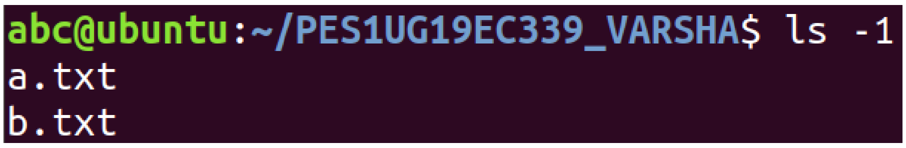
7. ls

This command gives list of files/directories in present working directory



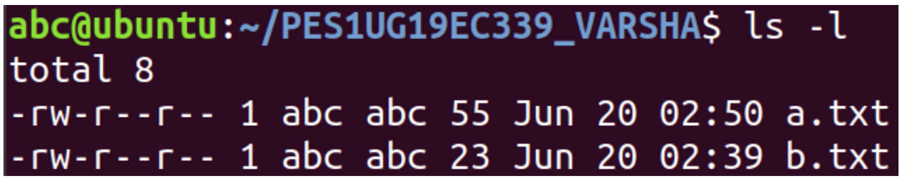
```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls
a.txt  b.txt
```

- ls -1 (Display one file per line)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls -1
a.txt
b.txt
```

- ls -l (Display all information about the file/directory)



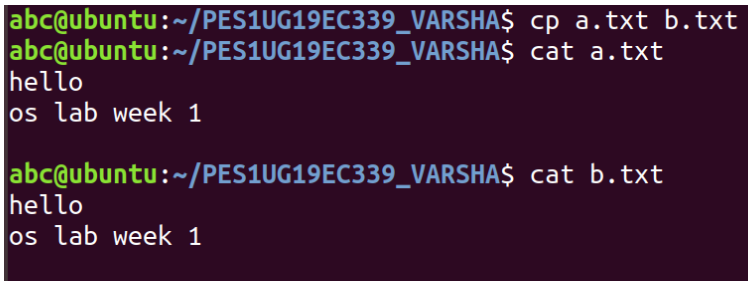
```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ls -l
total 8
-rw-r--r-- 1 abc abc 55 Jun 20 02:50 a.txt
-rw-r--r-- 1 abc abc 23 Jun 20 02:39 b.txt
```

8. cp

This command is used to copy content of one file to another file

Syntax: cp [options] source_file destination_file

Ex: cp a.txt b.txt (Here the content of a.txt is copied to b.txt)



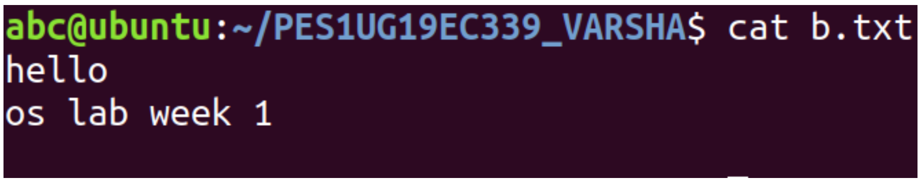
```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cp a.txt b.txt
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cat a.txt
hello
os lab week 1

abc@ubuntu:~/PES1UG19EC339_VARSHA$ cat b.txt
hello
os lab week 1
```

9. cat

It reads data from the file and gives their content as output.

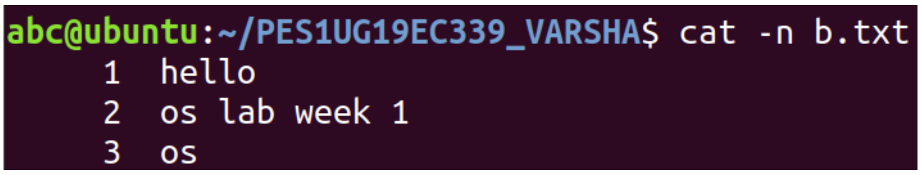
Syntax: cat file_name

Ex: cat b.txt (Here it displays the content of b.txt)



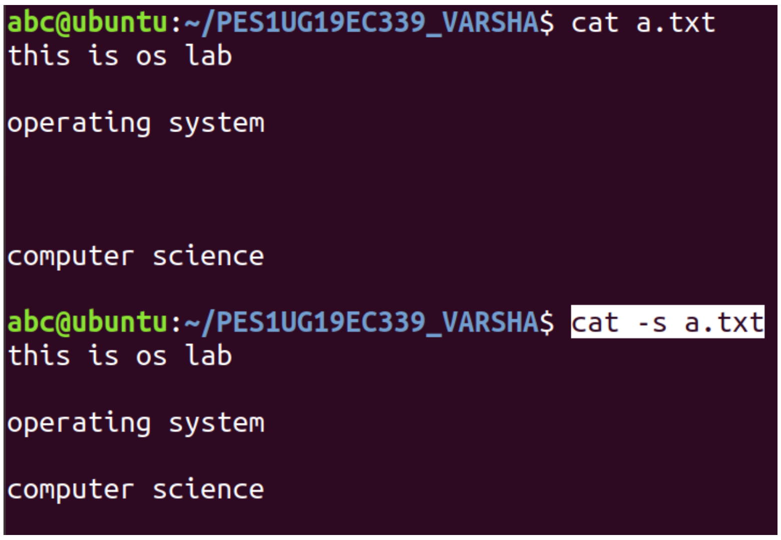
```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cat b.txt
hello
os lab week 1
```

- cat -n b.txt (Display content with line number)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cat -n b.txt
     1  hello
     2  os lab week 1
     3  os
```

- cat -s a.txt (Suppress repeated empty lines in output)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cat a.txt
this is os lab

operating system



computer science

abc@ubuntu:~/PES1UG19EC339_VARSHA$ cat -s a.txt
this is os lab

operating system

computer science
```

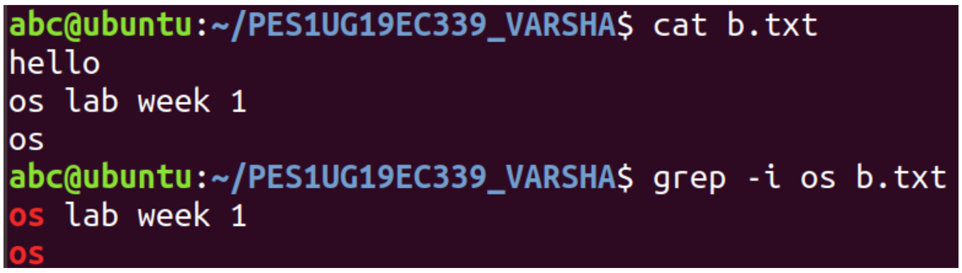
10.<u>grep</u>

This command search for text from a file,it can return where it finds a

a match or the lines where it doesn’t.

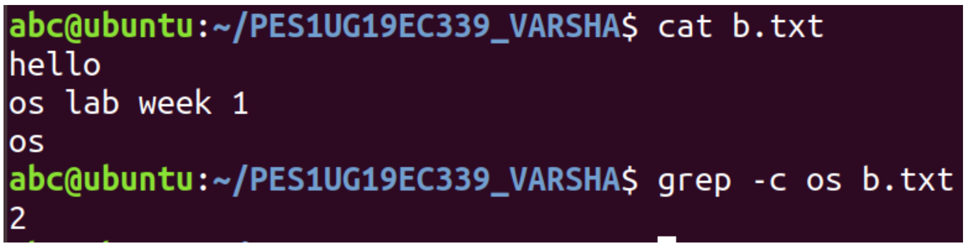
Syntax: grep [option] pattern file_name

- grep -i os b.txt (display a line that contains pattern)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cat b.txt
hello
os lab week 1
os
abc@ubuntu:~/PES1UG19EC339_VARSHA$ grep -i os b.txt
os lab week 1
os
```

- grep -c os b.txt (display count of the lines that match a pattern)



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ cat b.txt
hello
os lab week 1
os
abc@ubuntu:~/PES1UG19EC339_VARSHA$ grep -c os b.txt
2
```

2) Write a C program to display an array in reverse using index. Create Makefile (ex: make.mk below) and other files as shown below: (Hint: Refer to Makefile tutorial sent before to create these files)

Program:

header.h:

```
void rev_array(int *,int );

void array_print(int *,int);
```

client.c

```
#include<stdio.h>

#include"rev_head.h"

int main()

{

int n;

printf("Enter array size\n:");

scanf("%d",&n);

int arr[n];

printf("Enter array elements\n");

for(int i=0;i<n;i++)

	scanf("%d",&arr[i]);

for(int i=0;i<n;i++)

	printf("%d\n",arr[i]);

rev_array(arr,n);

}
```

server.c:

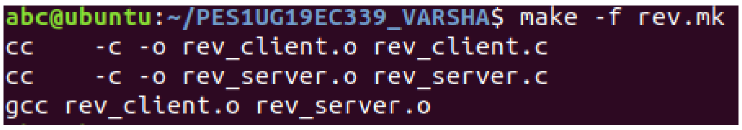
```
#include<stdio.h>
void array_print(int arr[],int n)
{
for(int i=0;i<n;i++)
	printf("%d\n",arr[i]);
}
void rev_array(int arr[],int n)
{
int b[n],i=0;
while(i<n)
	i++;
int j=i-1;
printf("after reversing\n");
for(int i=0;i<n;i++){
	b[i]=arr[j];
	j--;
}
array_print(b,n);
}
```

Makefile rev.mk:

```
a.out:rev_client.o rev_server.o
	gcc rev_client.o rev_server.c
client.o:rev_client.c rev_header.h
	gcc -c rev_client.c
server.o:rev_server.c rev_header.h
	gcc -c rev_server.c
```

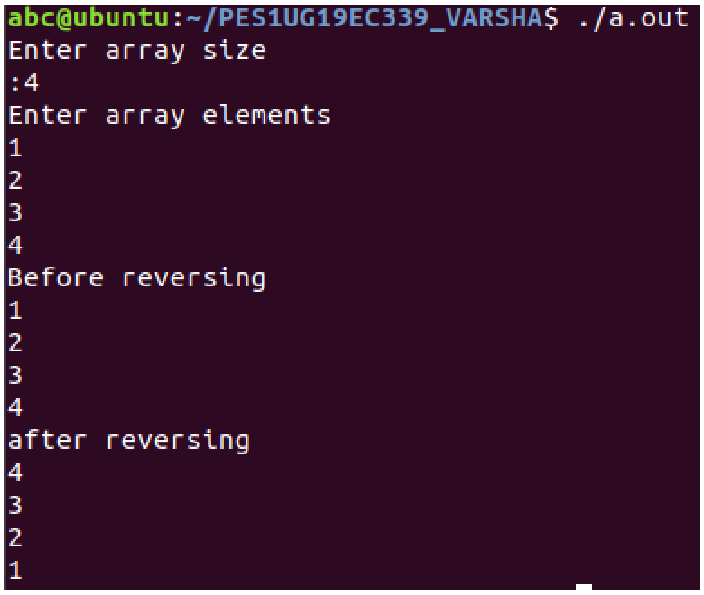
Steps to execute Makefile:

1. make -f filename
2. ./a.out



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ make -f rev.mk
cc    -c -o rev_client.o rev_client.c
cc    -c -o rev_server.o rev_server.c
gcc rev_client.o rev_server.o
```

OUTPUT:



```
abc@ubuntu:~/PES1UG19EC339_VARSHA$ ./a.out
Enter array size
:4
Enter array elements
1
2
3
4
Before reversing
1
2
3
4
after reversing
4
3
2
1
```

3) Answer the following questions

• Why do we use Makefile?

Makefile makes compilation much faster.

the **purpose** of a **makefile** is to be easily build an executable that might take many commands to create

• Is Makefile a shell script?

It is not a shell script. But shell script can be written using makefile.

• What does “clean” do in Makefile?

$make clean

It allows to get rid of object and executable files .

• How does make learn about the last modified files to be complied?

Make works out dependencies between targets and their dependencies, and then looks to see whether the files exist.If they d,It asks the operating system for the time and date the file was last modifies.

- What does Cflags in Makefile mean?

  Cflags is a variable that is most commonly used to add arguments to the compiler.

- Why do we use -f option with make command?

  By default,make command expects filename to be either makefile or makefile. If the make filename is different than any of these two -f is used along with the make command. i.e make -f filename.txt