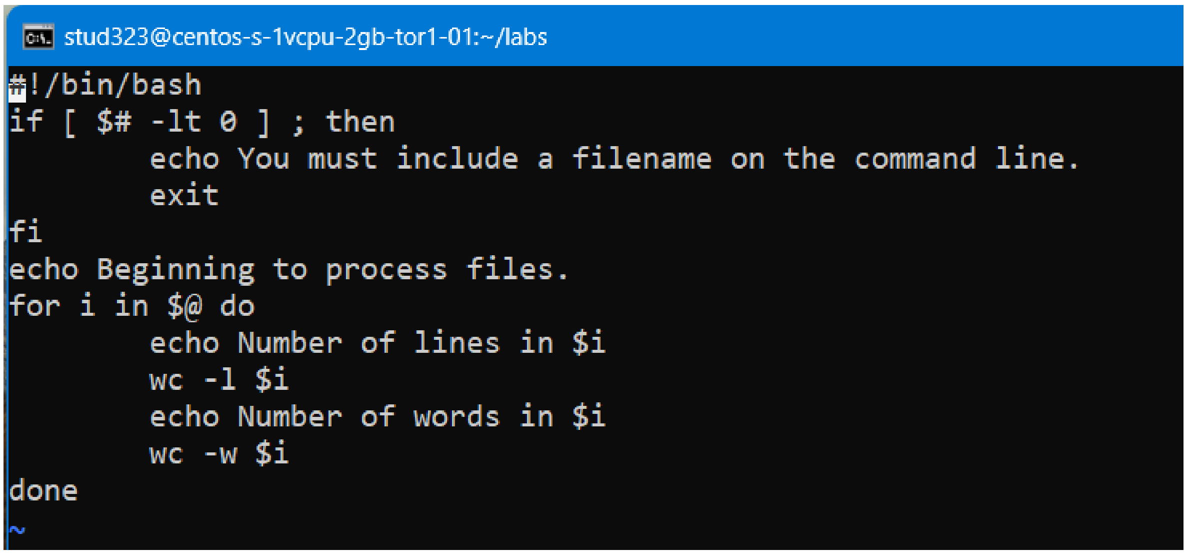
CIS341

Lab 02

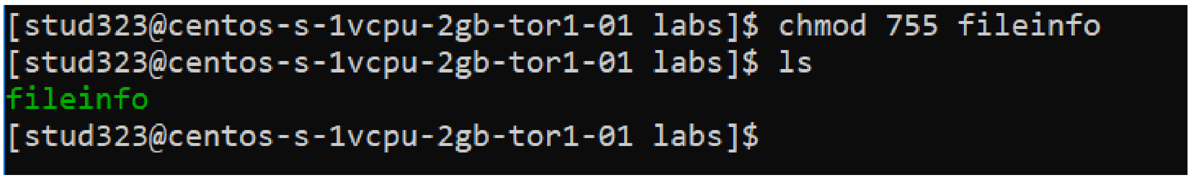
Name: Ishan Prasad *ID: 300167203*

1. Section 1
   a. *vi fileinfo* and editing the file



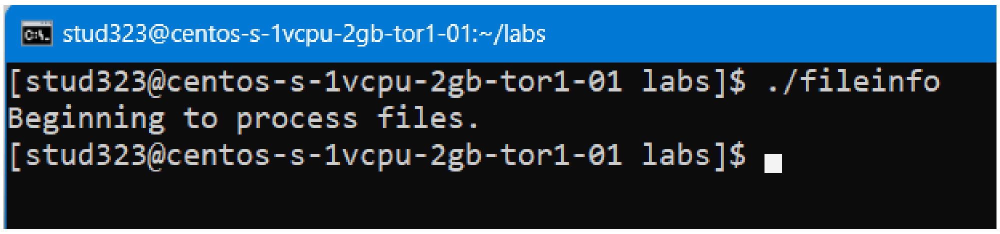
```
stud323@centos-s-1vcpu-2gb-tor1-01:~/labs
#!/bin/bash
if [ $# -lt 0 ] ; then
        echo You must include a filename on the command line.
        exit
fi
echo Beginning to process files.
for i in $@ do
        echo Number of lines in $i
        wc -l $i
        echo Number of words in $i
        wc -w $i
done
~
```

   b. *Chmod 755:* The equivalent command in letters would be ***chmod u+rwx g+rx o+rx fileinfo***.



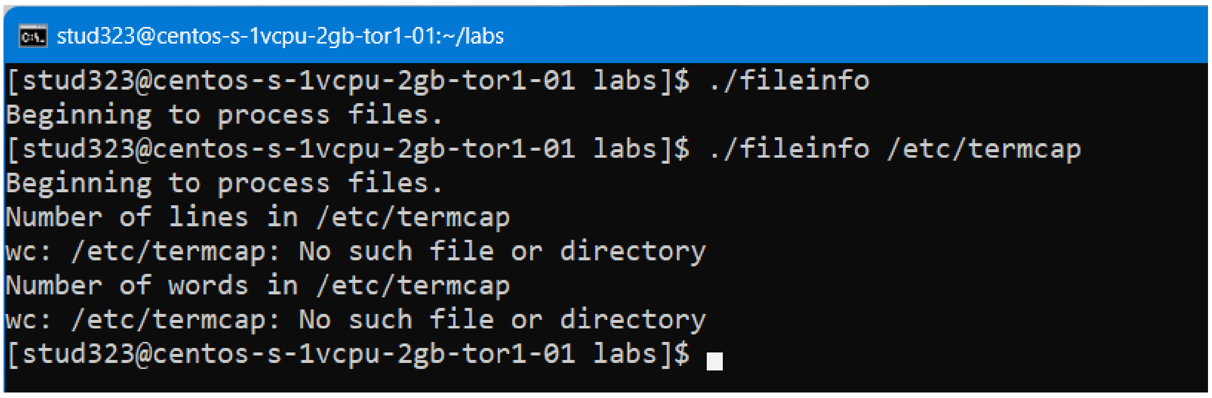
```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ chmod 755 fileinfo
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ls
fileinfo
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$
```

   c. *./fileinfo*: There is no output except the print statement used. The code checks for any command line parameters and if its less than 0, it would print out the "You must include a filename" statement. We can also use *$# > a* where a is any number, or *$# -ne 0*, where *ne* is not equal to some number (0 in this case).



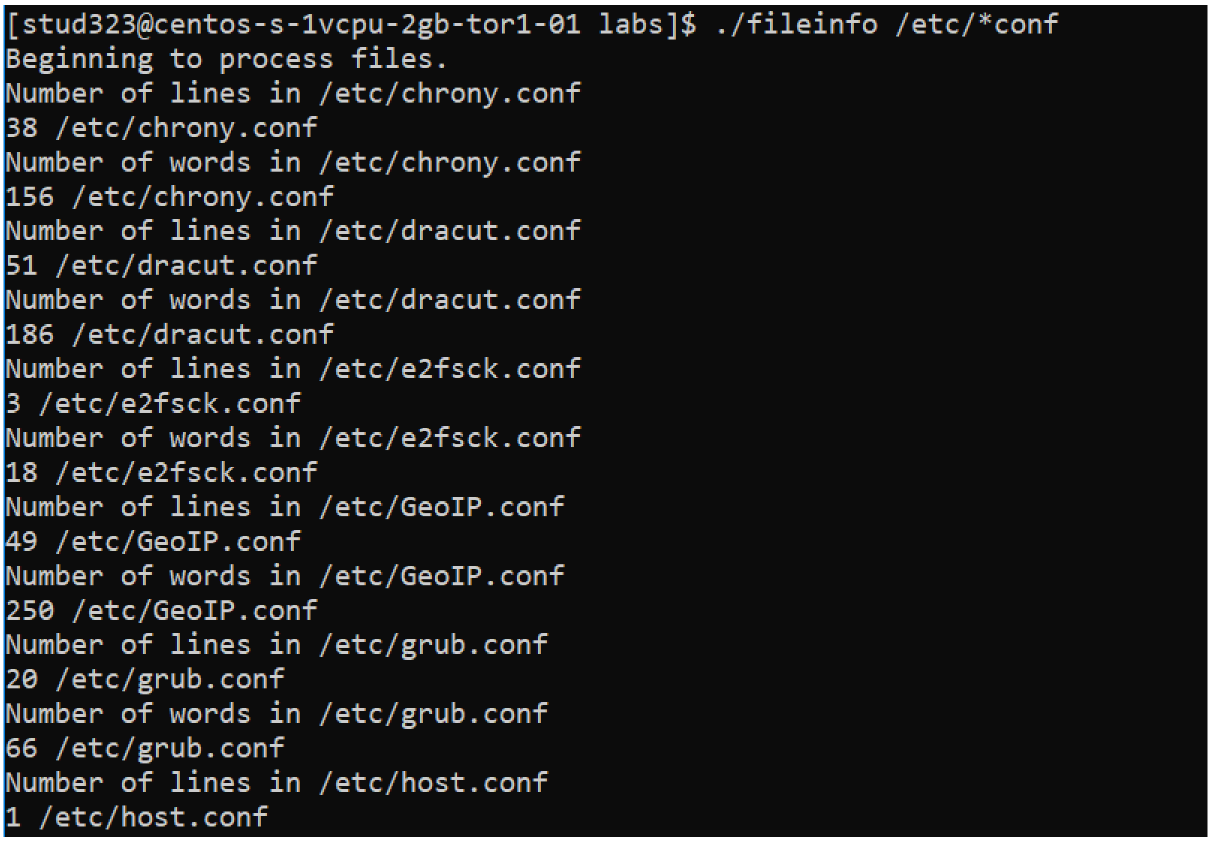
```
stud323@centos-s-1vcpu-2gb-tor1-01:~/labs
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./fileinfo
Beginning to process files.
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$
```

   d. *./fileinfo /etc/termcap:* Nothing happens since the file was not found on the system.


stud323@centos-s-1vcpu-2gb-tor1-01:~/labs

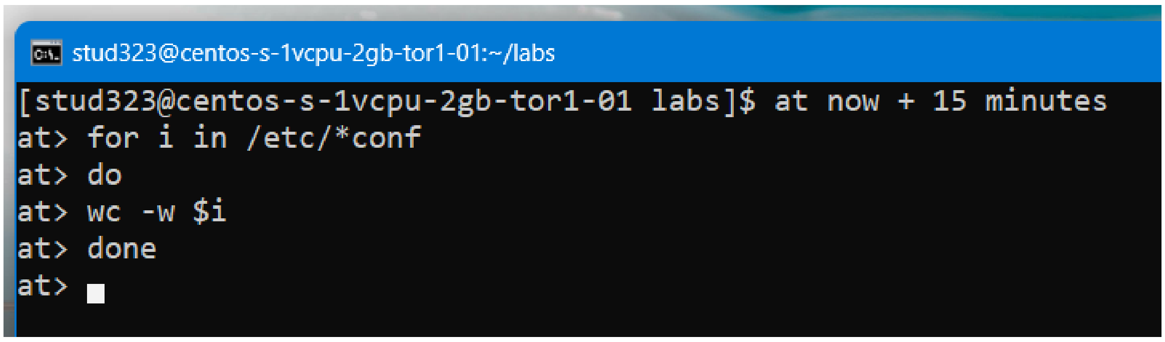
```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./fileinfo
Beginning to process files.
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./fileinfo /etc/termcap
Beginning to process files.
Number of lines in /etc/termcap
wc: /etc/termcap: No such file or directory
Number of words in /etc/termcap
wc: /etc/termcap: No such file or directory
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ 
```

e. *./fileinfo /etc/*conf:* This piece of code prints all the count of words and lines in any file which has *conf* succeeding a filename in the directory /etc. The only error messages found were permission denied.



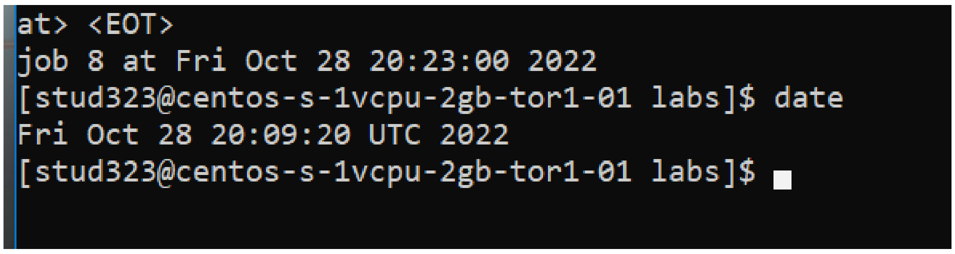
```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./fileinfo /etc/*conf
Beginning to process files.
Number of lines in /etc/chrony.conf
38 /etc/chrony.conf
Number of words in /etc/chrony.conf
156 /etc/chrony.conf
Number of lines in /etc/dracut.conf
51 /etc/dracut.conf
Number of words in /etc/dracut.conf
186 /etc/dracut.conf
Number of lines in /etc/e2fsck.conf
3 /etc/e2fsck.conf
Number of words in /etc/e2fsck.conf
18 /etc/e2fsck.conf
Number of lines in /etc/GeoIP.conf
49 /etc/GeoIP.conf
Number of words in /etc/GeoIP.conf
250 /etc/GeoIP.conf
Number of lines in /etc/grub.conf
20 /etc/grub.conf
Number of words in /etc/grub.conf
66 /etc/grub.conf
Number of lines in /etc/host.conf
1 /etc/host.conf
```

2. Section 2
   a. *At now + 15 minutes:*



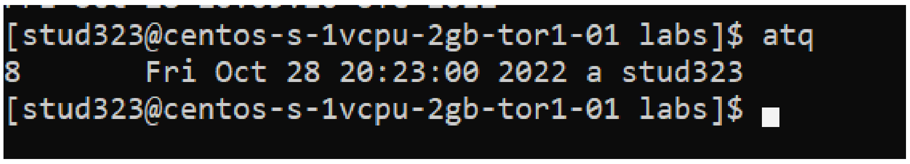
```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ at now + 15 minutes
at> for i in /etc/*conf
at> do
at> wc -w $i
at> done
at> 
```

   b. Time: the time shown was absolute time.



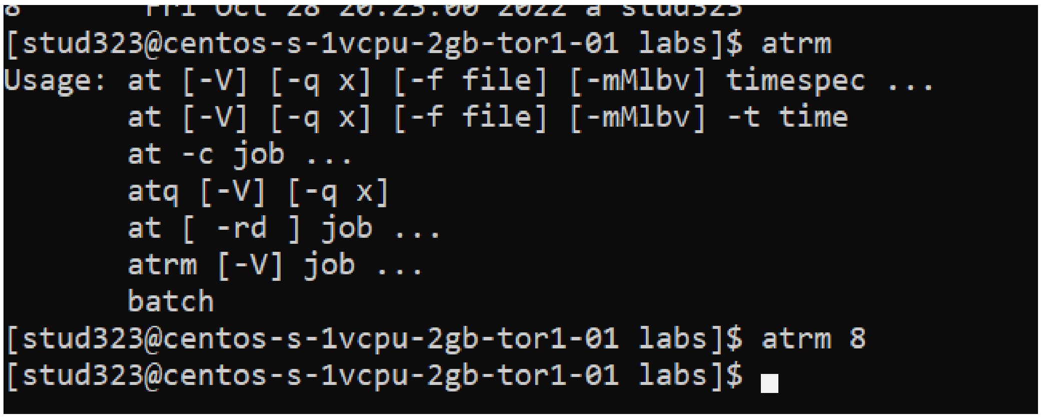
```
at> <EOT>
job 8 at Fri Oct 28 20:23:00 2022
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ date
Fri Oct 28 20:09:20 UTC 2022
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ 
```

   c. *atq:* The output shows the job number, time and date when it is going to be executed, and the user on which the job will be executed.



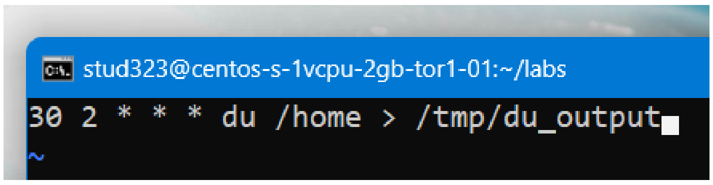
```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ atq
8       Fri Oct 28 20:23:00 2022 a stud323
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ 
```

   d. Atrm



```
8       Fri Oct 28 20:23:00 2022 a stud323
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ atrm
Usage: at [-V] [-q x] [-f file] [-mMlbv] timespec ...
       at [-V] [-q x] [-f file] [-mMlbv] -t time
       at -c job ...
       atq [-V] [-q x]
       at [ -rd ] job ...
       atrm [-V] job ...
       batch
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ atrm 8
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ 
```

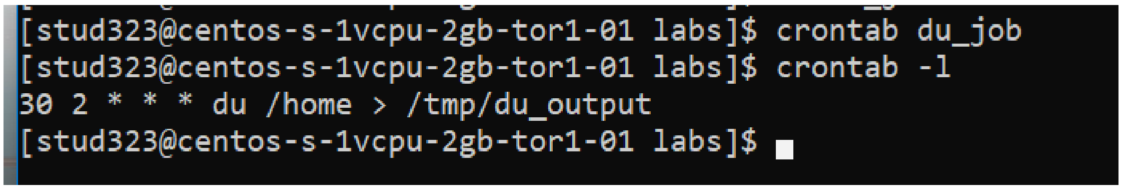
3. Section 3
   a. *Vi du_job:*



stud323@centos-s-1vcpu-2gb-tor1-01:~/labs

```
30 2 * * * du /home > /tmp/du_output
~
```

   b. *Crontab -l:* Displays the contents of the file.

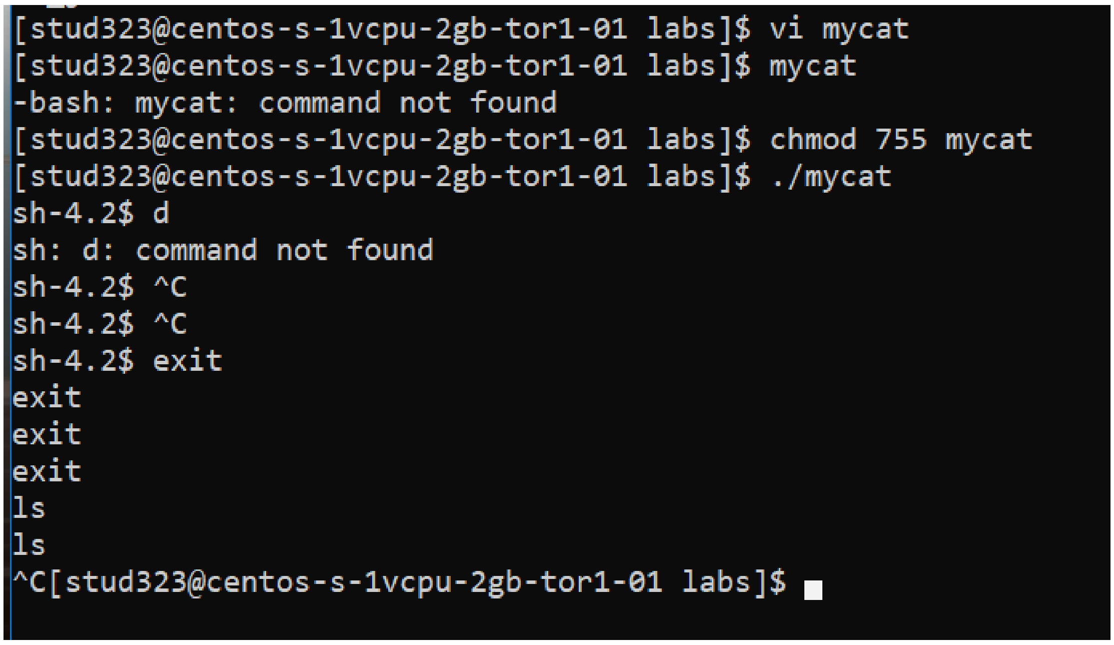


```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ crontab du_job
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ crontab -l
30 2 * * * du /home > /tmp/du_output
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$
```

4. Section 4
   a. Code:

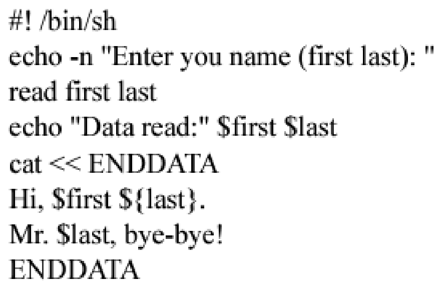
```
! /bin/sh -f
read s
while [ -n "$s" ]
do
  echo $s
  read s
done
```

Answer: This script changes the shell from bash to *sh*. The while loop keeps the shell to *sh* and reads the user inputs until the user exits voluntarily.



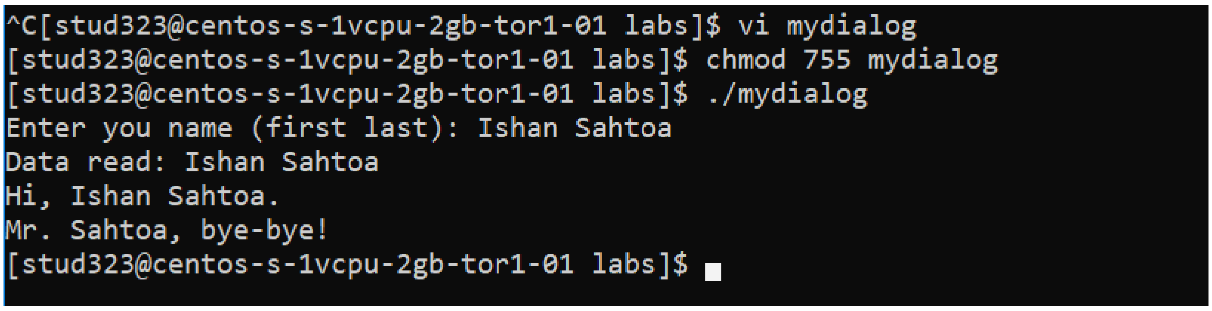
```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ vi mycat
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ mycat
-bash: mycat: command not found
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ chmod 755 mycat
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./mycat
sh-4.2$ d
sh: d: command not found
sh-4.2$ ^C
sh-4.2$ ^C
sh-4.2$ exit
exit
exit
exit
ls
ls
^C[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$
```

b. Code:



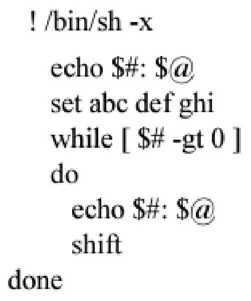
```
#! /bin/sh
echo -n "Enter you name (first last): "
read first last
echo "Data read:" $first $last
cat << ENDDATA
Hi, $first ${last}.
Mr. $last, bye-bye!
ENDDATA
```

Answer: The first line changes the shell to *sh*. Line 2 prints a statement for the user to input their name (first + last). The third line reads the user input and the next couple of lines output what the user input. ENDDATA is used for printing multiple lines of text without using echo every time.



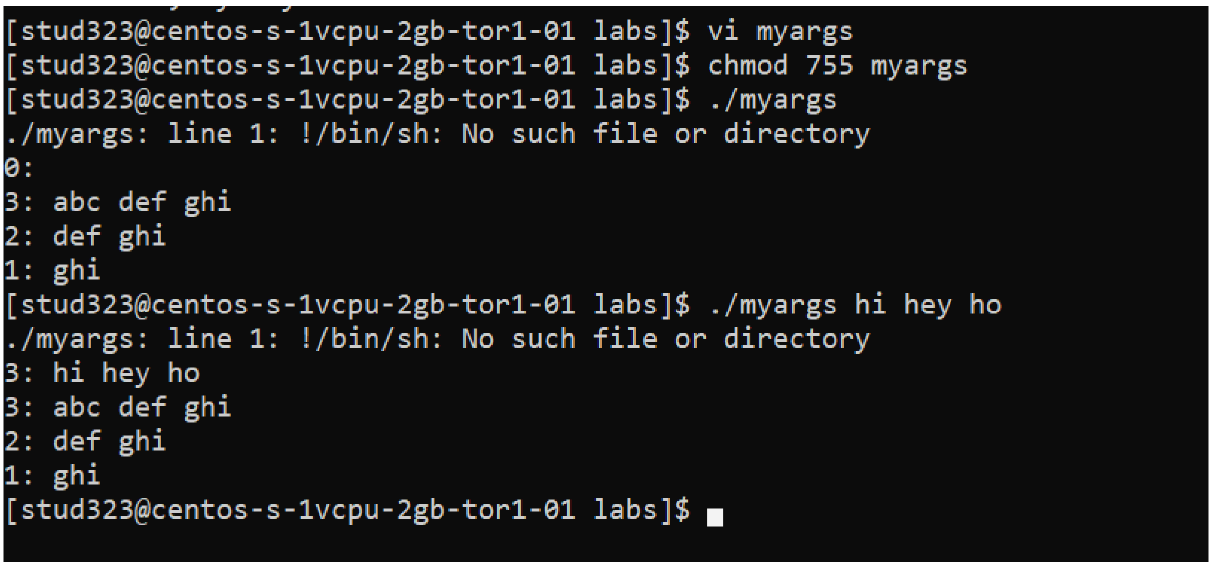
```
^C[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ vi mydialog
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ chmod 755 mydialog
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./mydialog
Enter you name (first last): Ishan Sahtoa
Data read: Ishan Sahtoa
Hi, Ishan Sahtoa.
Mr. Sahtoa, bye-bye!
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ 
```

c. Code:



```
 ! /bin/sh -x
   echo $#: $@
   set abc def ghi
   while [ $# -gt 0 ]
   do
     echo $#: $@
     shift
done
```

Answer: The first sets the shell to sh. Line 2 prints the number and the content of the command line arguments supplied. Line 3 sets 3 arguments as *abc, def, ghi.* Next we have a while loop on Line 4 which checks if the number of arguments is greater than 0 and if yes, we have the *do* code (Line 5) which prints (Line 6) *3: abc def ghi*, which it got from Line 3. Line 7 shifts the arguments to one step behind.



```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ vi myargs
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ chmod 755 myargs
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./myargs
./myargs: line 1: !/bin/sh: No such file or directory
0:
3: abc def ghi
2: def ghi
1: ghi
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./myargs hi hey ho
./myargs: line 1: !/bin/sh: No such file or directory
3: hi hey ho
3: abc def ghi
2: def ghi
1: ghi
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$
```

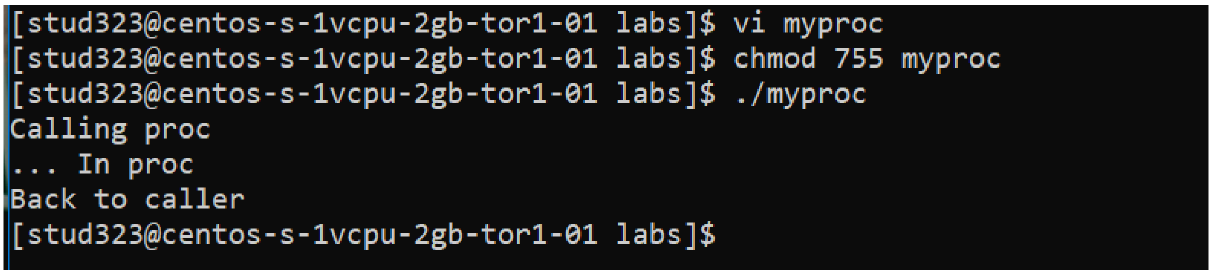
d. Code:

```
main()
{
  proc1()
  {
    echo ... In proc
    return 0
  }

  echo Calling proc
  proc1
  echo Back to caller
  return 0
}

main
```

Answer: First we define a *main()* function in Line 1. In *main()* function, we define a *proc1()* function (line 3) which prints "… In proc" statement (line 5) and returns 0 (line 6). Next, we print "Calling proc" on line 8 and call the *proc1* function (line 9). Then we print another statement which indicates we are back to the main function (line 10) and then return 0 (line 11). On line 12, we call the *main* function to start the whole process.



```
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ vi myproc
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ chmod 755 myproc
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$ ./myproc
Calling proc
... In proc
Back to caller
[stud323@centos-s-1vcpu-2gb-tor1-01 labs]$
```