Mohammad Fakhruddin Babar
ID: M564K763

Task 1: Linux Access Control Warm-up

**Q1.1: Search for Linux file permissions online.
What permission setting are you specifying with 0644?**

Ans: User will get read and write access, group and others will get only read access. And it’s not a set- UID program.

**Q1.2: Did the name of the file changed?
If so, why where you successful in doing so this time?**

Ans: Yeah, name of the file is changed. We can see that user1, wushock are in the same group and they can create a file, rename a file in that folder. But it cant write anything into that file.

Q 1.3 Why do you see an operation not permitted error?

Ans: As this file is owned by the user1, so wushock doesn’t have any access to change the permission.

Q 1.4 Whose permissions did you change using the above chmod command?

Ans: By using above command, both the owner and group lost their write privilege and others got write privilege, but it doesn’t have read privilege.

**Q 1.5 Why does running cat on the file produce a permission denied error, but the echo/append
command does not?**

Ans: We have removed wushock from the group and it became others. And others have only written access, not the read access. That’s why we couldn’t read the file. but able to write into the file.

Q 1.6 Are you still able to change the name but not the mode (permissions) of the file?

Ans: No, this time I couldn’t change the file name too.

Q 1.7 What do you see displayed?

Ans:
Play Angry
Go Shockers

Q 1.8 Why do you see a permission denied error?

Ans: Group has only the read access, so it couldn’t write into a file.

Q1.9: What umask setting is returned and how do you calculate the default permissions of a new file?

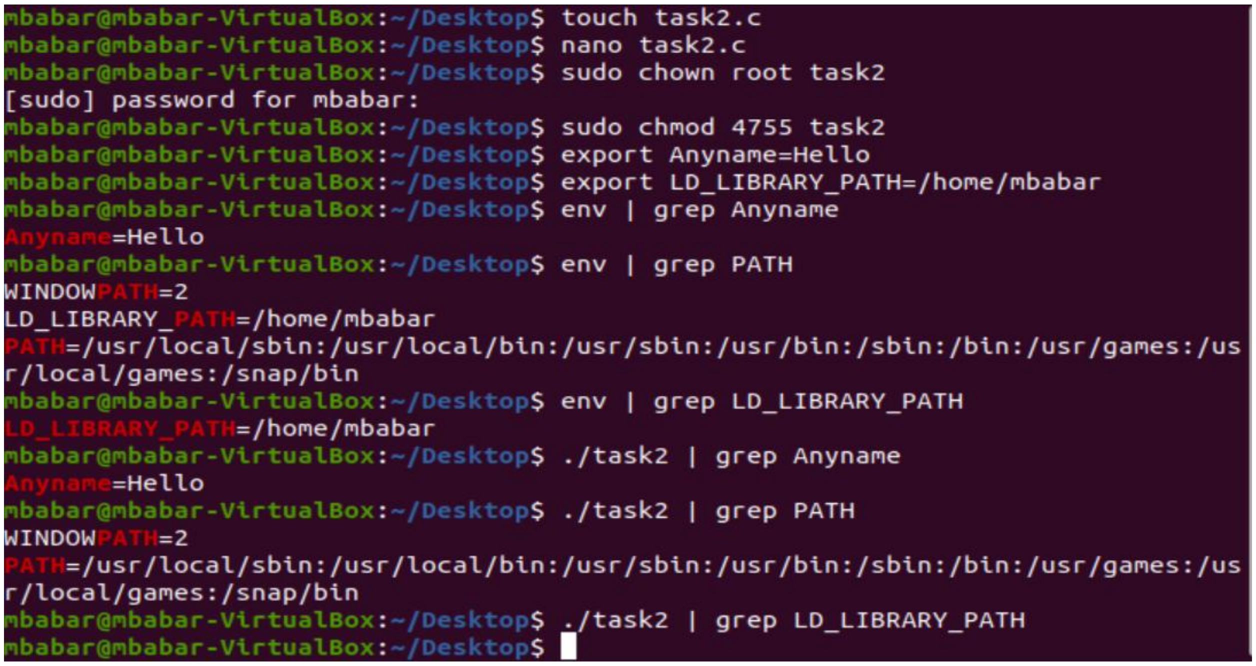
Ans: Unmask value is 002. By subtracting this from 777, we get 775. So, user has rwx, group has rwx and others have r-x permission

Q 1.10 What do each of these umask settings mean for users attempting to access any new files or directories you create?

Ans: Those different umask are resetting the file permission.

Task 2: Environment Variable and Set-UID Programs

Ans:2.1:



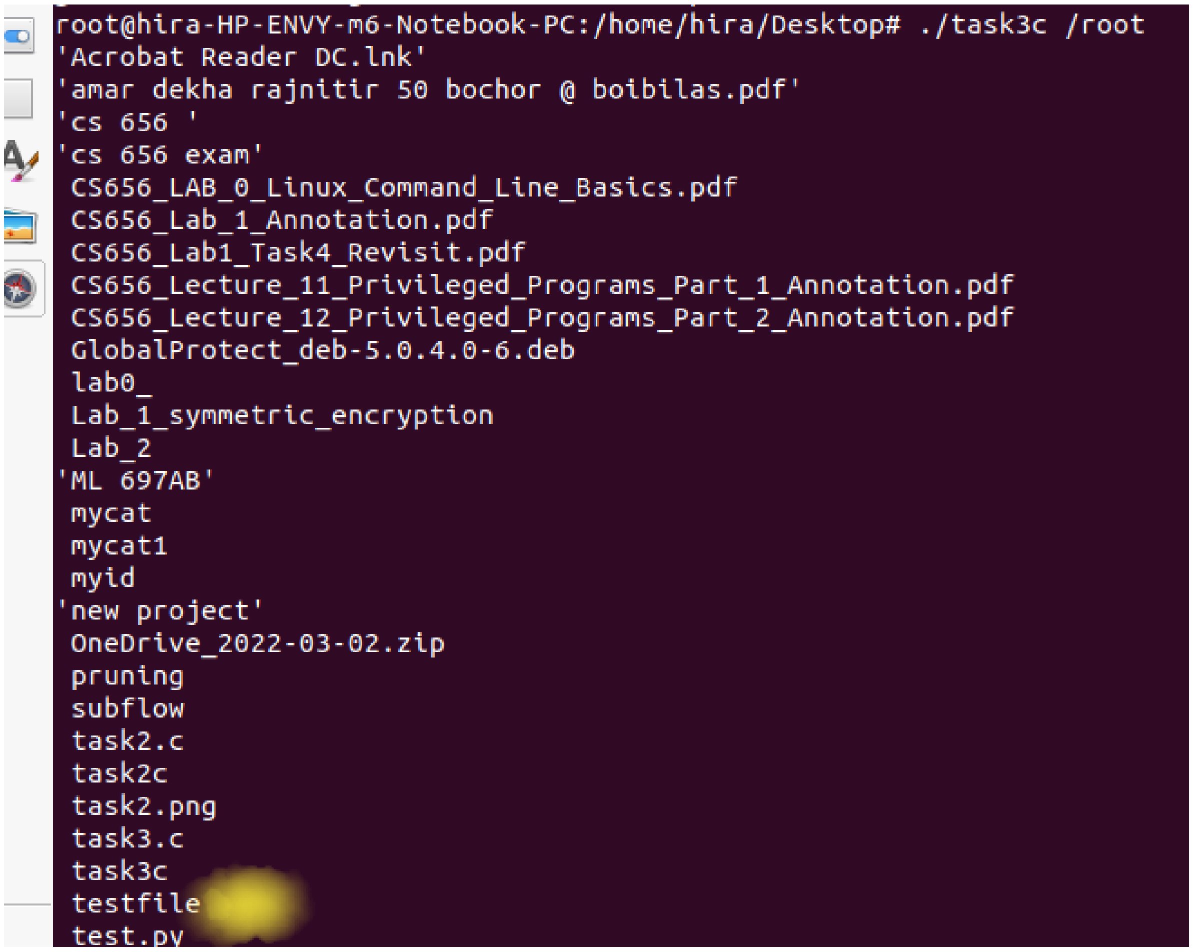
```
mbabar@mbabar-VirtualBox:~/Desktop$ touch task2.c
mbabar@mbabar-VirtualBox:~/Desktop$ nano task2.c
mbabar@mbabar-VirtualBox:~/Desktop$ sudo chown root task2
[sudo] password for mbabar:
mbabar@mbabar-VirtualBox:~/Desktop$ sudo chmod 4755 task2
mbabar@mbabar-VirtualBox:~/Desktop$ export Anyname=Hello
mbabar@mbabar-VirtualBox:~/Desktop$ export LD_LIBRARY_PATH=/home/mbabar
mbabar@mbabar-VirtualBox:~/Desktop$ env | grep Anyname
Anyname=Hello
mbabar@mbabar-VirtualBox:~/Desktop$ env | grep PATH
WINDOWPATH=2
LD_LIBRARY_PATH=/home/mbabar
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/local/games:/snap/bin
mbabar@mbabar-VirtualBox:~/Desktop$ env | grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/home/mbabar
mbabar@mbabar-VirtualBox:~/Desktop$ ./task2 | grep Anyname
Anyname=Hello
mbabar@mbabar-VirtualBox:~/Desktop$ ./task2 | grep PATH
WINDOWPATH=2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/local/games:/snap/bin
mbabar@mbabar-VirtualBox:~/Desktop$ ./task2 | grep LD_LIBRARY_PATH
mbabar@mbabar-VirtualBox:~/Desktop$ 
```

Ans:2.2:

All variables must pass from the parent to the child when it comes to passing environment variables (e.g., PATH, Anyname). However, we can see that the LD LIBRARY PATH was not passed to the child. This is because the dynamic linker on most operating systems removes variables that control dynamic linking from the environment of setuid executables for security reasons.

Task 3: **The PATH Environment Variable and Set-UID Programs**

Ans:3.1: “tesfile’’ has been created.



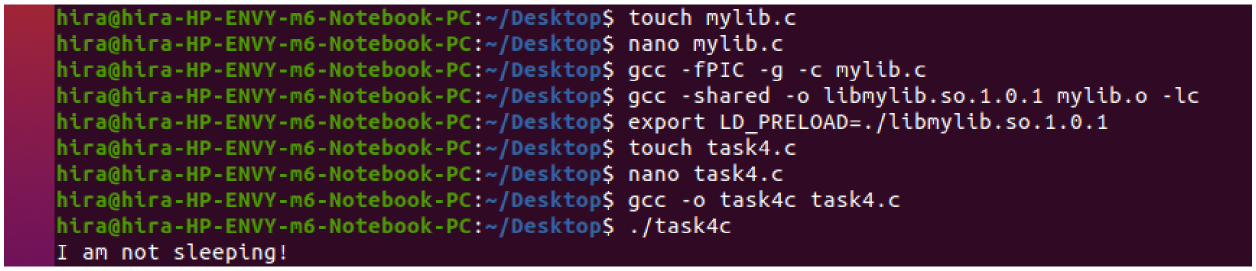
```
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# ./task3c /root
'Acrobat Reader DC.lnk'
'amar dekha rajnitir 50 bochor @ boibilas.pdf'
'cs 656 '
'cs 656 exam'
 CS656_LAB_0_Linux_Command_Line_Basics.pdf
 CS656_Lab_1_Annotation.pdf
 CS656_Lab1_Task4_Revisit.pdf
 CS656_Lecture_11_Privileged_Programs_Part_1_Annotation.pdf
 CS656_Lecture_12_Privileged_Programs_Part_2_Annotation.pdf
 GlobalProtect_deb-5.0.4.0-6.deb
 lab0_
 Lab_1_symmetric_encryption
 Lab_2
'ML 697AB'
 mycat
 mycat1
 myid
'new project'
 OneDrive_2022-03-02.zip
 pruning
 subflow
 task2.c
 task2c
 task2.png
 task3.c
 task3c
 testfile
 test.py
```

Ans:3.2:

Task3 will run with root privileges, the task3 is owned by the root and it’s a Set-UID program, so it will run with the root privilege

Task 4: The LD PRELOAD Environment Variable and Set-UID Programs

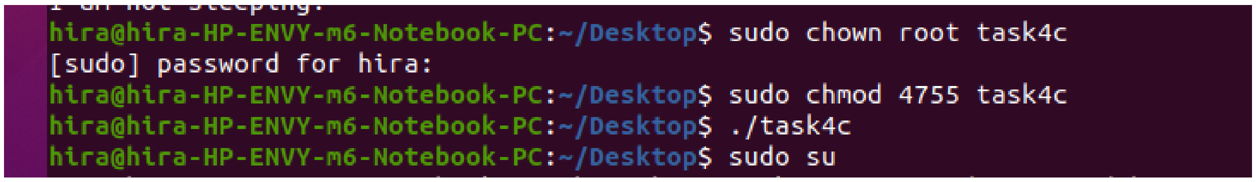
Ans:4.1:



```
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ touch mylib.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ nano mylib.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ gcc -fPIC -g -c mylib.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ export LD_PRELOAD=./libmylib.so.1.0.1
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ touch task4.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ nano task4.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ gcc -o task4c task4.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ ./task4c
I am not sleeping!
```

Task4 used the malicious library and printed "I'm not sleeping." As we have exported the environment variable LD_PRELOAD, the sleep function has been linked to that function instead of the libc standard sleep function.

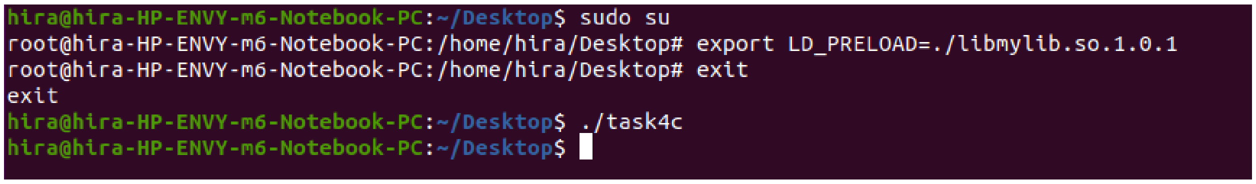
Ans:4.2:



```
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ sudo chown root task4c
[sudo] password for hira:
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ sudo chmod 4755 task4c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ ./task4c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ sudo su
```

Because of security concerns, the program uses the standard libc library. If the program is Set-UID and the effective user ID differs from the id of the owner process, then the LD PRELOAD environment variable is ignored.

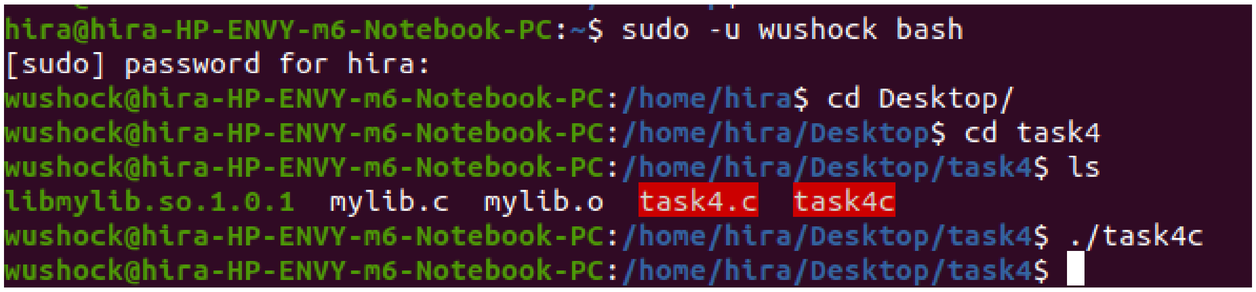
Ans:4.3:



```
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ sudo su
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# export LD_PRELOAD=./libmylib.so.1.0.1
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# exit
exit
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ ./task4c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ 
```

This time task4 uses the standard libc library, the same as the previous answer. If the effective user ID is different than the process’s real ID, then the LD_PRELOAD won’t be taken into consideration in the complication if the program is Set-UID. Here we can see that this operation has no relation to whether we export the environment variable as root or normal user.

Ans:4.4:

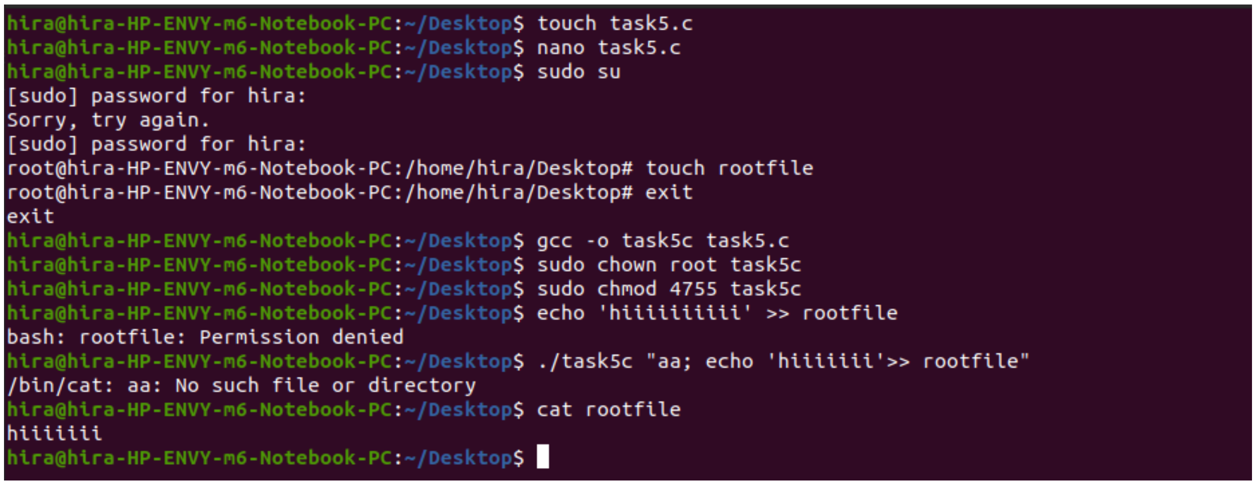


```
hira@hira-HP-ENVY-m6-Notebook-PC:~$ sudo -u wushock bash
[sudo] password for hira:
wushock@hira-HP-ENVY-m6-Notebook-PC:/home/hira$ cd Desktop/
wushock@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop$ cd task4
wushock@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop/task4$ ls
libmylib.so.1.0.1  mylib.c  mylib.o  task4.c  task4c
wushock@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop/task4$ ./task4c
wushock@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop/task4$
```

We made Task4 a Set-UID program owned by a non-root user different from the regular user, say wushock. Export the LD PRELOAD environment variable again. However, this time in the wushock account. And executed task 4 as wushock. Task 4 employs the standard libc library, producing the same result as the previous task.

Task 5: **Invoking External Programs Using system**() **versus execve**()

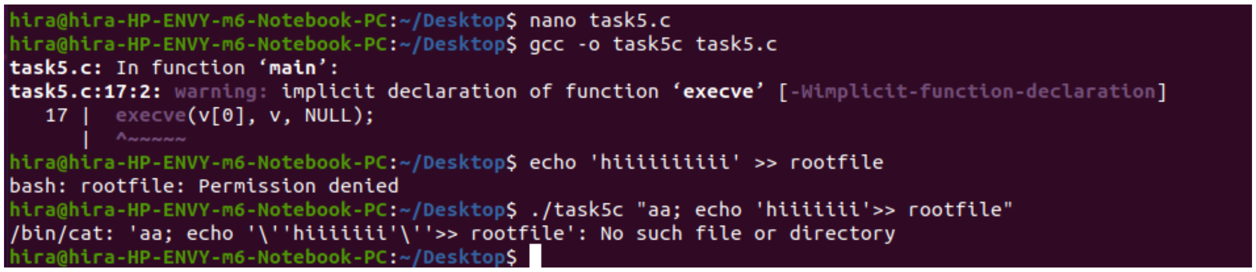
Ans:**5.1**:



```
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ touch task5.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ nano task5.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ sudo su
[sudo] password for hira:
Sorry, try again.
[sudo] password for hira:
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# touch rootfile
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# exit
exit
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ gcc -o task5c task5.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ sudo chown root task5c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ sudo chmod 4755 task5c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ echo 'hiiiiiiiii' >> rootfile
bash: rootfile: Permission denied
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ ./task5c "aa; echo 'hiiiiii'>> rootfile"
/bin/cat: aa: No such file or directory
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ cat rootfile
hiiiiii
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$
```

We create a file named “rootfile” as a root user. We attempted to echo a message and it denied that. But after exploiting the program we able to write into that file as a normal user.

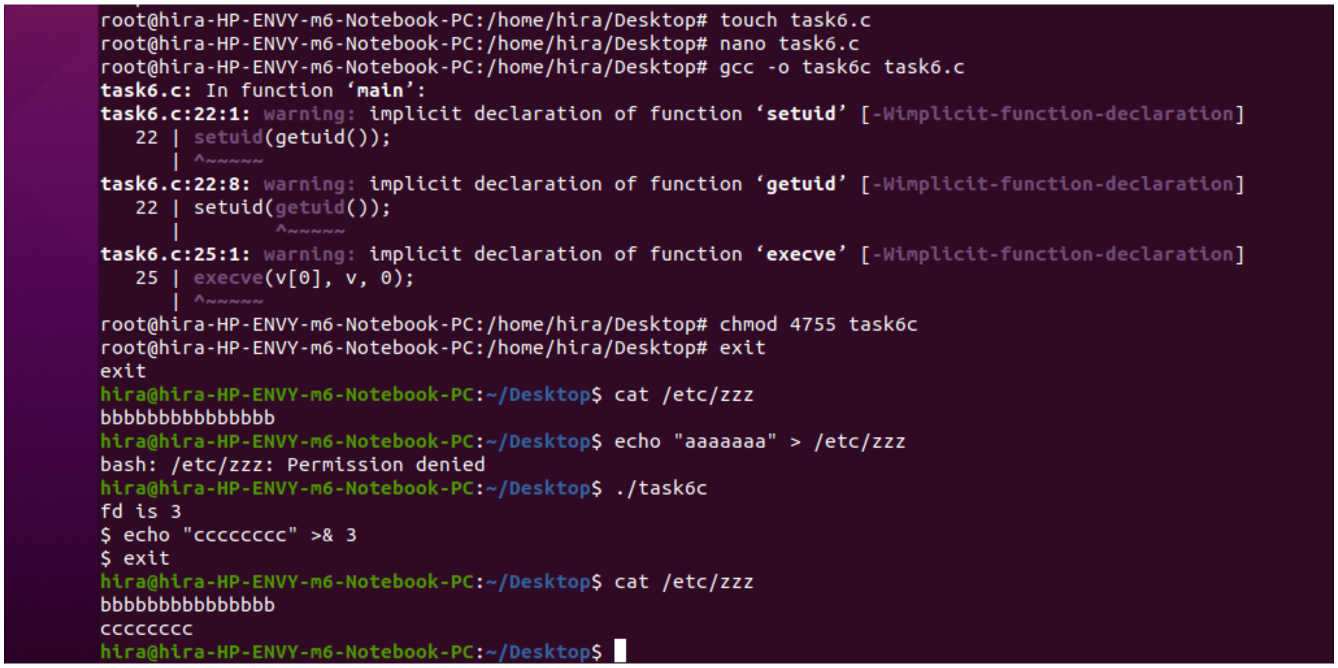
Ans:5.2: commenting system() command and uncomment execv() command.



```
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ nano task5.c
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ gcc -o task5c task5.c
task5.c: In function ‘main’:
task5.c:17:2: warning: implicit declaration of function ‘execve’ [-Wimplicit-function-declaration]
   17 |  execve(v[0], v, NULL);
      |  ^~~~~~
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ echo 'hiiiiiiiii' >> rootfile
bash: rootfile: Permission denied
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ ./task5c "aa; echo 'hiiiiii'>> rootfile"
/bin/cat: 'aa; echo '\''hiiiiii'\''>> rootfile': No such file or directory
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ 
```

We cannot modify the file. At first we used the system command, we could modify the file, because in fact the system command will open at first a shell (/bin/sh) after that it passes the command to it, and in the shell, we can add pass and execute more than a command, we just need to seperate with a semicolon. In the other hand the execve command will directly execute the command and doesn’t invoke the shell.

Task 6: Capability Leaking:



```
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# touch task6.c
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# nano task6.c
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# gcc -o task6c task6.c
task6.c: In function ‘main’:
task6.c:22:1: warning: implicit declaration of function ‘setuid’ [-Wimplicit-function-declaration]
   22 | setuid(getuid());
      | ^~~~~~
task6.c:22:8: warning: implicit declaration of function ‘getuid’ [-Wimplicit-function-declaration]
   22 | setuid(getuid());
      |        ^~~~~~
task6.c:25:1: warning: implicit declaration of function ‘execve’ [-Wimplicit-function-declaration]
   25 | execve(v[0], v, 0);
      | ^~~~~~
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# chmod 4755 task6c
root@hira-HP-ENVY-m6-Notebook-PC:/home/hira/Desktop# exit
exit
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ cat /etc/zzz
bbbbbbbbbbbbbbb
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ echo "aaaaaaa" > /etc/zzz
bash: /etc/zzz: Permission denied
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ ./task6c
fd is 3
$ echo "cccccccc" >& 3
$ exit
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ cat /etc/zzz
bbbbbbbbbbbbbbb
cccccccc
hira@hira-HP-ENVY-m6-Notebook-PC:~/Desktop$ 
```

The file /etc/zzz will be modified, the reason is that the file was opened with a root privilege, which has the capability of modifying the file, so even after downgrading the privileges, the process still remains with that capability and it passes to the child, consequently, the child can modify the file. To overcome this problem, we need to close the file before downgrading the privileges to clean up all the gained privileges.