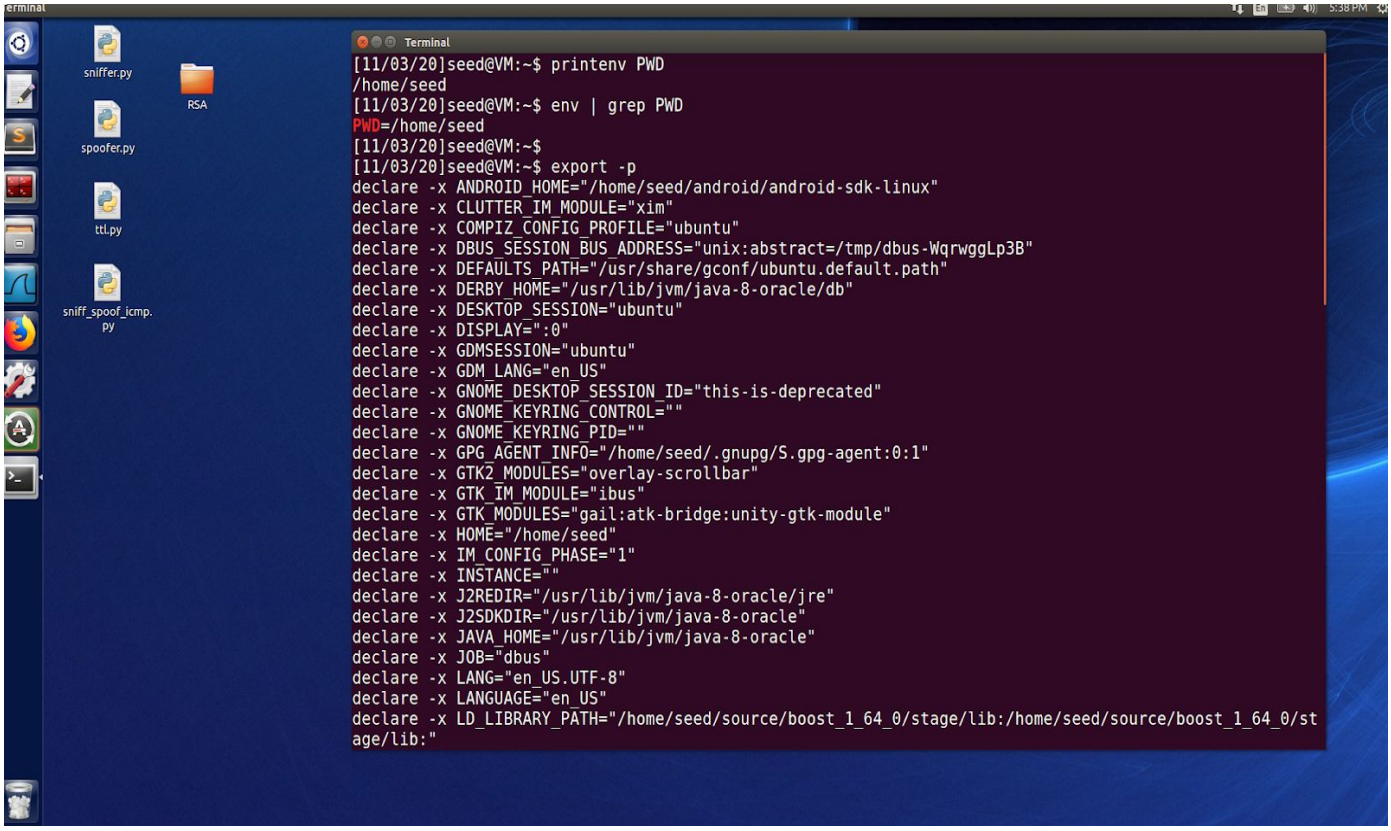
intrusion Detection
Group 14
Environment Variable and Set-UID Program Lab
Vlad Smirnov, Raim Aliyev, Mark Musseau
11-4-2020

Task 1: Manipulating Environment Variables

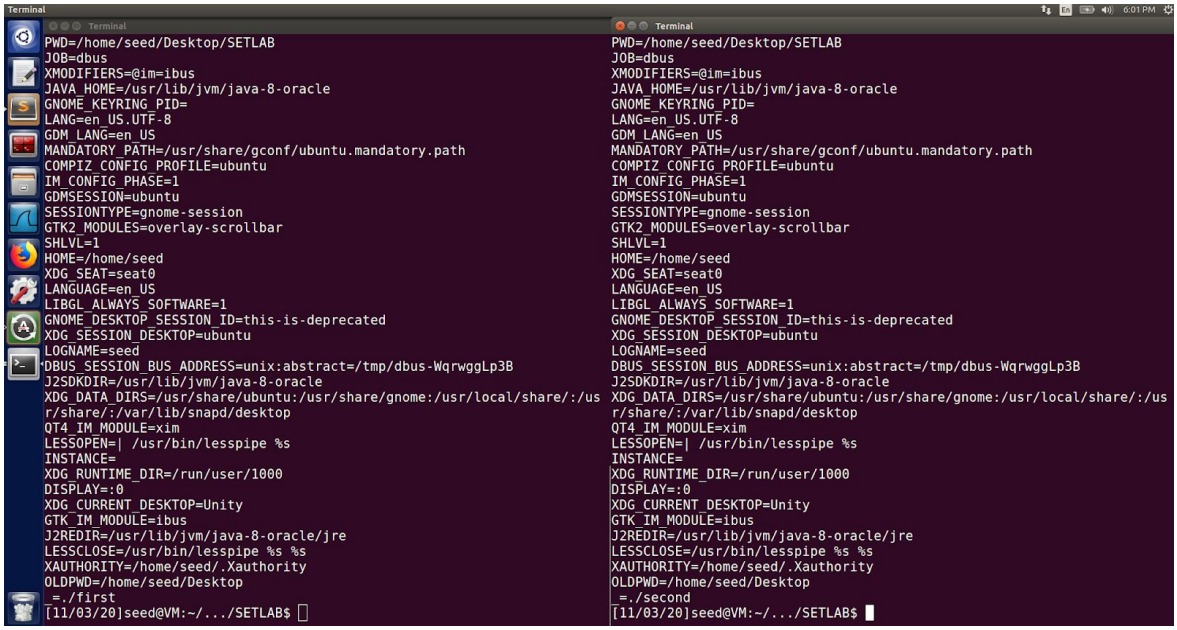
Printenv and env are commands that allow us to print out environment variables and export or unset allows us to change/ modify them here is a screenshot to show them used.



For example I can do export NEW = 1 will set a new variable and give it a value of 1.

Task 2: Passing Environment Variables from Parent Process to Child Process

In this process we will see how a child process gets its environment variables from its parent. fork() creates a new process by duplicating the calling process and is then called a child and the original is called the parent. You can see the screenshot on the next page.

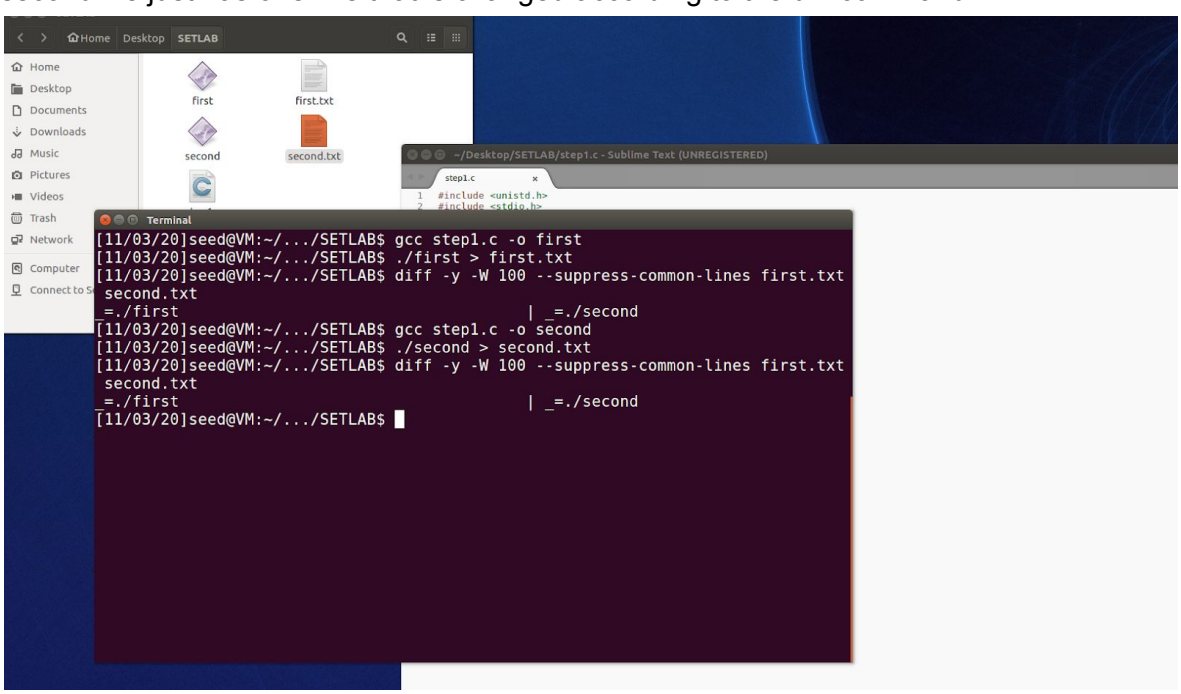


We changed the code and in the left it printed with the print env being enabled in child but in the right it was not and only in the parent. The printouts look identical and it is hard to find the differences so we save the files and use the diff command to help us find where the two outputs are different.We ran this command diff -y -W 100 --suppress-common-lines first.txt second.txt to tell us where the differences are

|: shows if line is changed in second file

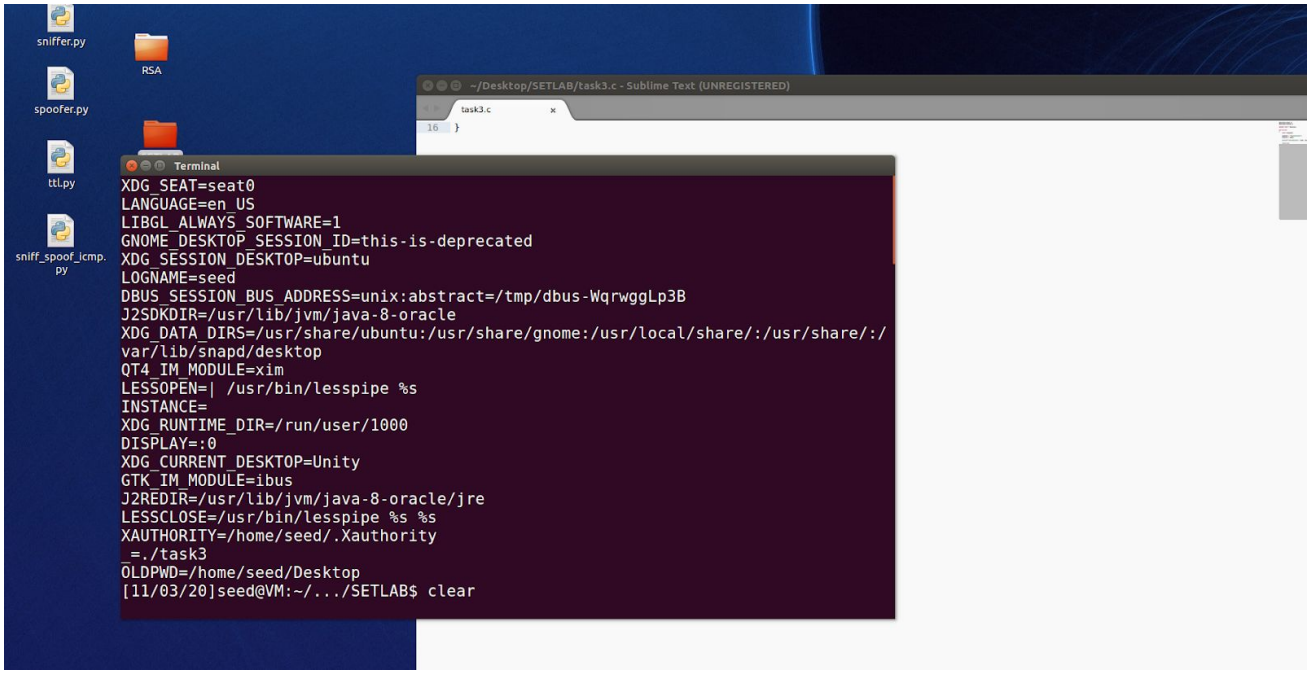
<: shows line was deleted in second file

>:shows line was added to second file and not in firstIf you look at the picture you will notice that the second file just has one line that is changed according to the diff command.



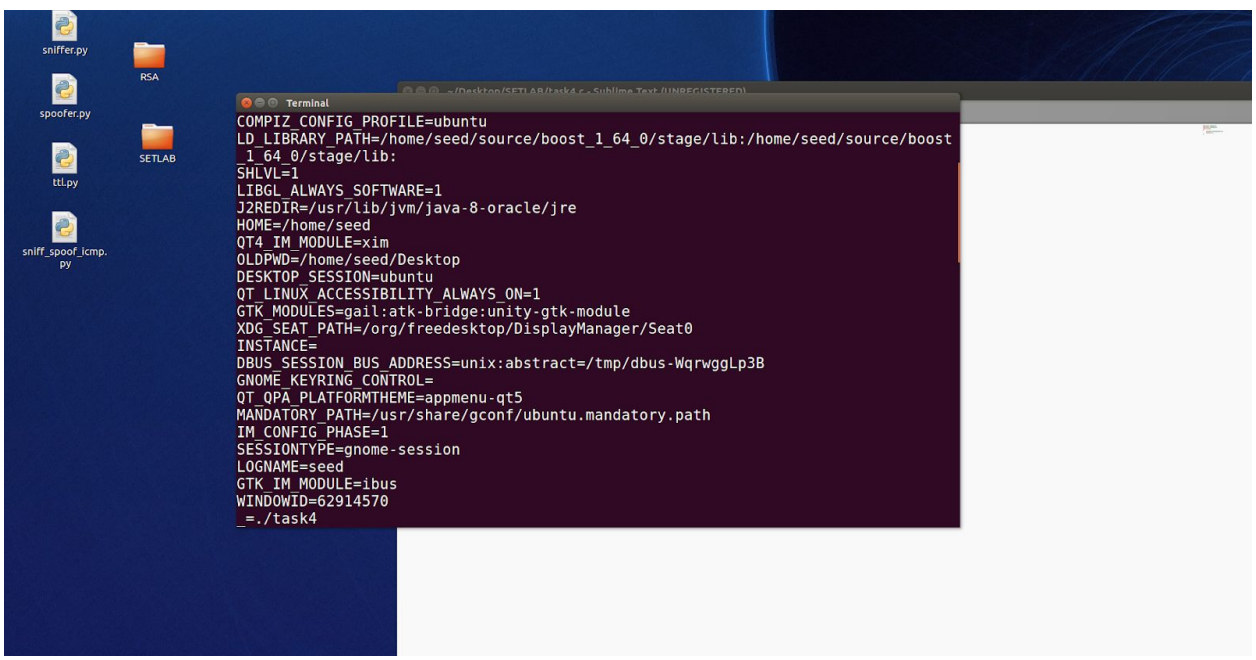
Task 3: Environment Variables and execve()

The program is supposed to print out current environmental variables of the current process. When the argument was null it was empty but when we changed the argument to environ then we saw something very similar to what happened with the previous tasks in what was printed.



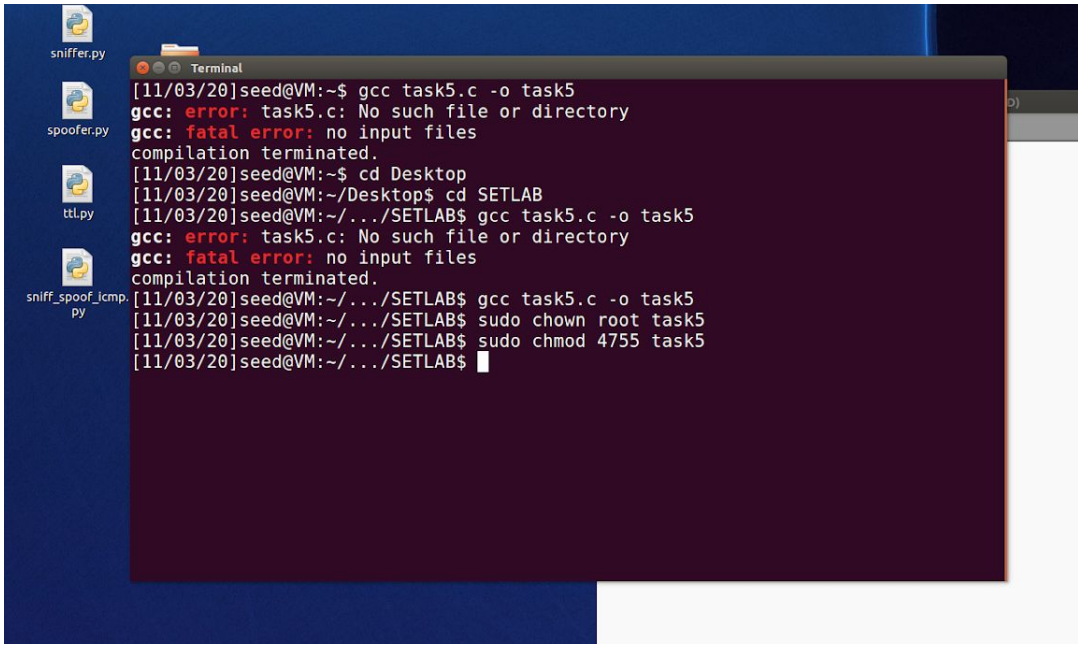
Task 4: Environment Variables and system()

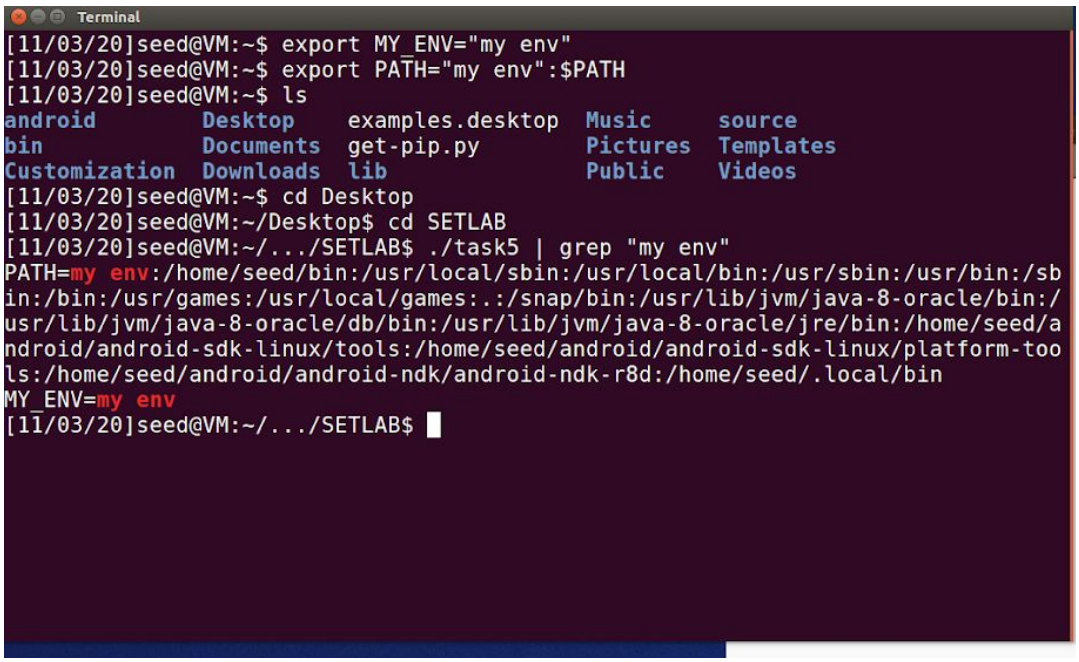
In this task we will use system() instead of the previous command and shee what happens. The same thing happened because system actually uses execv for some of the implementation therefore we expect similar results.



Task 5: Environment Variable and Set-UID Programs

We are given a program to compile and change its ownership to root and then make it a SETUID program. And we did that in the below screenshot





We made an environment and then run the program and the shell forks a process and uses the child to run it and the environment variables from the parent go to the child process which is running the program.

Task 6: The PATH Environment Variable and Set-UID Programs

The shell program invoked is calling system() within the setuid program. The behavior of the shell program can be modified by the environment variables like PATH. You can change the variables and control the behavior of the program and can make it run with root privilege.

```
root@VM: /home/seed/Desktop/SETLAB
[11/03/20]seed@VM:~$ export PATH=/home/seed:$PATH
[11/03/20]seed@VM:~$ cd Desktop
[11/03/20]seed@VM:~/Desktop$ cd SETLAB
[11/03/20]seed@VM:~/.../SETLAB$ gcc task6.c -o task6
task6.c: In function 'main':
task6.c:3:5: warning: implicit declaration of function 'system' [-Wimplicit-func
tion-declaration]
     system("ls");
     ^
[11/03/20]seed@VM:~/.../SETLAB$ su
Password:
root@VM:/home/seed/Desktop/SETLAB# chmod u+s task6
root@VM:/home/seed/Desktop/SETLAB# exit
exit
[11/03/20]seed@VM:~/.../SETLAB$ ./task6
first      second      step1.c  task3.c  task4.c  task5.c  task6.c
first.txt  second.txt  task3    task4    task5    task6
[11/03/20]seed@VM:~/.../SETLAB$
```

Task 7: The LD PRELOAD Environment Variable and Set-UID Programs

We will make a program that overrides the sleep() function in lib, then we compile and set the environment variable. Then we compile another program in the same environment variable. The same program will basically run but with different behaviors.

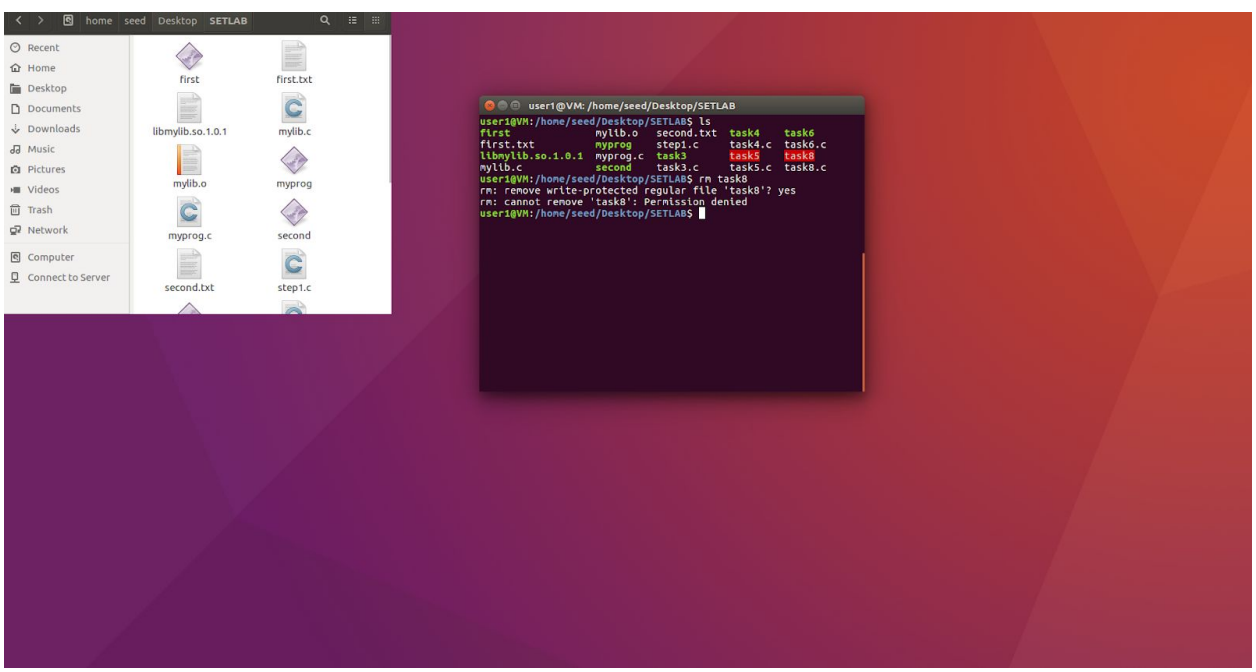
The program does not always give the same results, it is because the environment variable is missing in the time it does not give the same result.

```
Terminal
root@VM: /home/seed/Desktop/SETLAB
[11/04/20]seed@VM:~$ cd Desktop
[11/04/20]seed@VM:~/Desktop$ cd SETLAB
[11/04/20]seed@VM:~/.../SETLAB$ gcc -fPIC -g -c mylib.c
[11/04/20]seed@VM:~/.../SETLAB$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[11/04/20]seed@VM:~/.../SETLAB$ export LD_PRELOAD=./libmylib.so.1.0.1
[11/04/20]seed@VM:~/.../SETLAB$ export PATH=/home/seed:$PATH
[11/04/20]seed@VM:~/.../SETLAB$ gcc -o myprog myprog.c
myprog.c: In function 'main':
myprog.c:4:5: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
     sleep(1);
     ^
[11/04/20]seed@VM:~/.../SETLAB$ ./myprog
I am not sleeping!
[11/04/20]seed@VM:~/.../SETLAB$ sudo chmod u+s myprog
[11/04/20]seed@VM:~/.../SETLAB$ ./myprog
I am not sleeping!
[11/04/20]seed@VM:~/.../SETLAB$ sudo su
root@VM:/home/seed/Desktop/SETLAB# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/Desktop/SETLAB# ./myprog
root@VM:/home/seed/Desktop/SETLAB# exit
exit
[11/04/20]seed@VM:~/.../SETLAB$ ./myprog
I am not sleeping!
[11/04/20]seed@VM:~/.../SETLAB$ sudo su
root@VM:/home/seed/Desktop/SETLAB# useradd -d /usr/user1 -m user1
root@VM:/home/seed/Desktop/SETLAB# chown user1 myprog
root@VM:/home/seed/Desktop/SETLAB# chgrp user1 myprog
root@VM:/home/seed/Desktop/SETLAB# exit
exit
[11/04/20]seed@VM:~/.../SETLAB$ export LD_PRELOAD=./libmylib.so.1.0.1
[11/04/20]seed@VM:~/.../SETLAB$ ./myprog
I am not sleeping!
[11/04/20]seed@VM:~/.../SETLAB$
```

Task 8: Invoking External Programs Using system() versus execve()

We make a program and make it owned by root and make it a setuid program, can we remove a file not writable to us? We logged into user1 that was created from the above task and used terminal to locate to the task8 program and were not able to delete it from user1.

```
Terminal
first.txt          mylib.o  second      task3    task4.c  task6
libmylib.so.1.0.1  myprog   second.txt  task3.c  task5    task6.c
[11/04/20]seed@VM:~/.../SETLAB$ clear
[11/04/20]seed@VM:~/.../SETLAB$ gcc task8.c -o task8
[11/04/20]seed@VM:~/.../SETLAB$ sudo chown root task8
[11/04/20]seed@VM:~/.../SETLAB$ sudo chmod u+s task8
[11/04/20]seed@VM:~/.../SETLAB$ ./task8
Please type a file name.
[11/04/20]seed@VM:~/.../SETLAB$ ls
first              mylib.c  myprog.c    step1.c  task4    task5.c  task8
first.txt          mylib.o  second      task3    task4.c  task6    task8.c
libmylib.so.1.0.1  myprog   second.txt  task3.c  task5    task6.c
[11/04/20]seed@VM:~/.../SETLAB$ sudo passwd user1
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
[11/04/20]seed@VM:~/.../SETLAB$
```



Task 9: Capability Leaking

In this task we compile the given program and give it root privileges and make it a setuid program and run it and see if the file zzz was modified. And the file was modified because the file was open before being set to uid and therefore was modified.