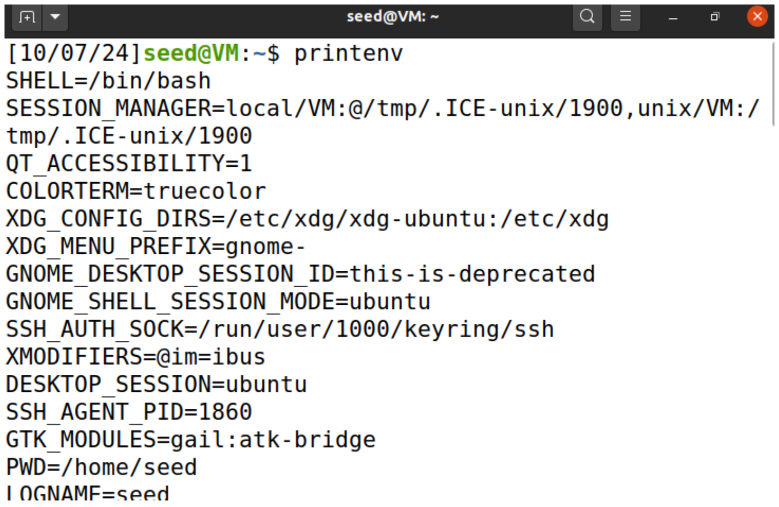
SEEDLabs–Environment Variable and Set-UID Program Lab

2 LabTasks

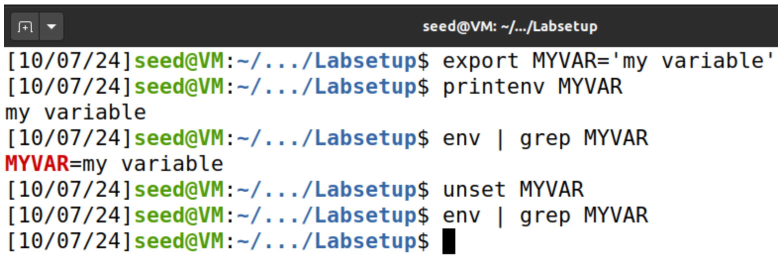
2.1 Task 1:

To print the environment variables, use the printenv command:



```
[10/07/24]seed@VM:~$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1900,unix/VM:/
tmp/.ICE-unix/1900
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1860
GTK_MODULES=gail:atk-bridge
PWD=/home/seed
LOGNAME=seed
```

Using the command unset:

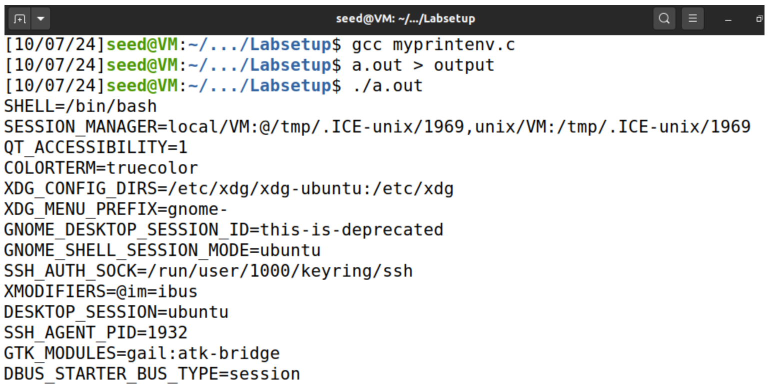


```
[10/07/24]seed@VM:~/.../Labsetup$ export MYVAR='my variable'
[10/07/24]seed@VM:~/.../Labsetup$ printenv MYVAR
my variable
[10/07/24]seed@VM:~/.../Labsetup$ env | grep MYVAR
MYVAR=my variable
[10/07/24]seed@VM:~/.../Labsetup$ unset MYVAR
[10/07/24]seed@VM:~/.../Labsetup$ env | grep MYVAR
[10/07/24]seed@VM:~/.../Labsetup$ 
```

2.2 Task 2:

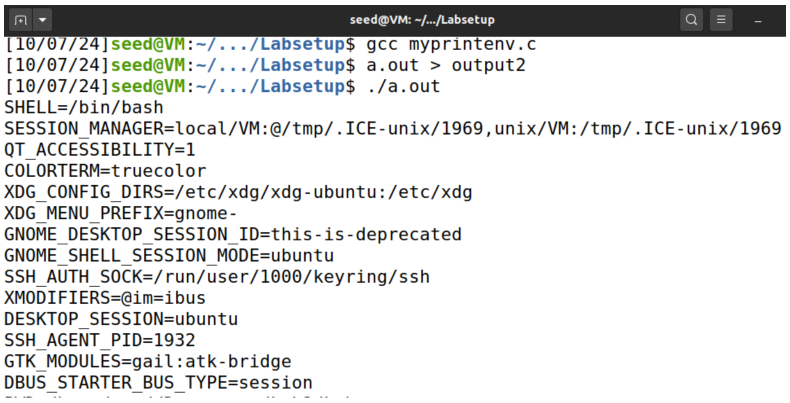
Step 1:

1. Added the myprintenv file to Labsetup folder and ran the compilation.
2. Executed the file and kept the output file.



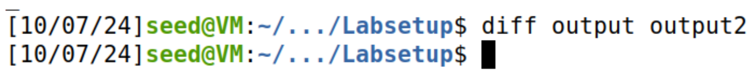
```
[10/07/24]seed@VM:~/.../Labsetup$ gcc myprintenv.c
[10/07/24]seed@VM:~/.../Labsetup$ a.out > output
[10/07/24]seed@VM:~/.../Labsetup$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1969,unix/VM:/tmp/.ICE-unix/1969
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1932
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
```

Step 2:
1. The `printenv()` statement in the parent process was uncommented, while the one in the child process was left commented out. The file was then saved and compiled.
2. The program was executed, and the results were saved in the `output2` file.



```
[10/07/24]seed@VM:~/.../Labsetup$ gcc myprintenv.c
[10/07/24]seed@VM:~/.../Labsetup$ a.out > output2
[10/07/24]seed@VM:~/.../Labsetup$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1969,unix/VM:/tmp/.ICE-unix/1969
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1932
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
```

Step 3 :
Using the `diff` command to find differences.

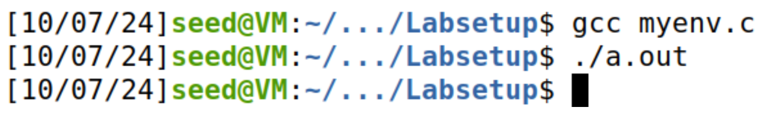


```
[10/07/24]seed@VM:~/.../Labsetup$ diff output output2
[10/07/24]seed@VM:~/.../Labsetup$ 
```

Conclusion: There is no difference between the two cases because the child process inherits the environment variables from the parent process.

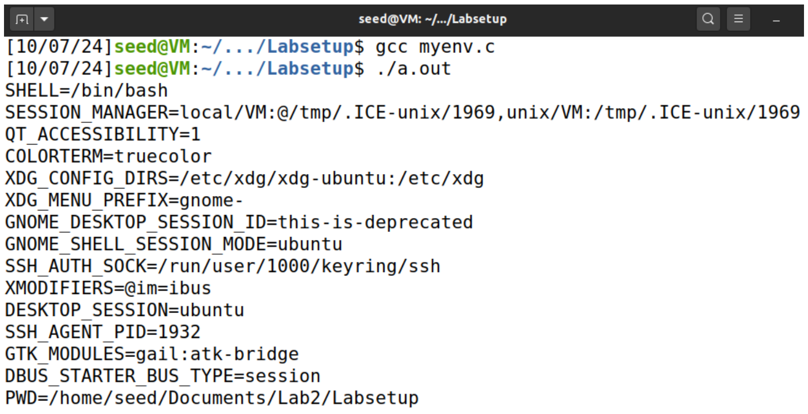
2.3 Task 3:

Step 1:
1. After loading the myenv file into the folder, the file was compiled.
2. Executed the file



```
[10/07/24]seed@VM:~/.../Labsetup$ gcc myenv.c
[10/07/24]seed@VM:~/.../Labsetup$ ./a.out
[10/07/24]seed@VM:~/.../Labsetup$ 
```

Step 2:
1. Updated the `myenv.c` file by modifying the line from `execve("/usr/bin/env", argv, NULL)` to `execve("/usr/bin/env", argv, environ)`.
2. Recompiled the file and executed it again.

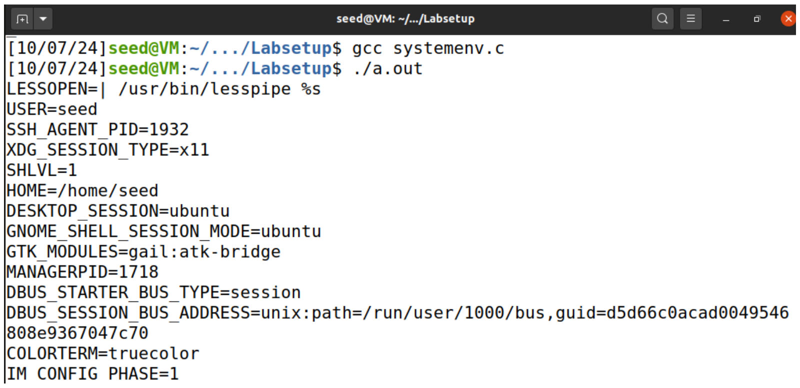


```
[10/07/24]seed@VM:~/.../Labsetup$ gcc myenv.c
[10/07/24]seed@VM:~/.../Labsetup$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1969,unix/VM:/tmp/.ICE-unix/1969
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1932
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Documents/Lab2/Labsetup
```

Step 3: Conclusion
In the initial step, the function execve("/usr/bin/env", argv, NULL) is executed without passing any environment variables. In the following step, the current environment variables are used to overwrite the first process by calling the function execve("/usr/bin/env", argv, environ).

2.4 Task 4:

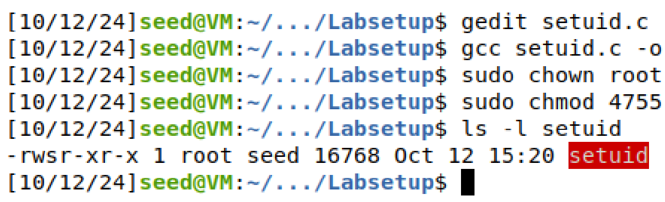
The environment variables of the calling process are passed to the new program /bin/sh when the system() function is executed through the creation of the file systemenv.c.



```
[10/07/24]seed@VM:~/.../Labsetup$ gcc systemenv.c
[10/07/24]seed@VM:~/.../Labsetup$ ./a.out
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=1932
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=1718
DBUS_STARTER_BUS_TYPE=session
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=d5d66c0acad0049546
808e9367047c70
COLORTERM=truecolor
IM_CONFIG_PHASE=1
```

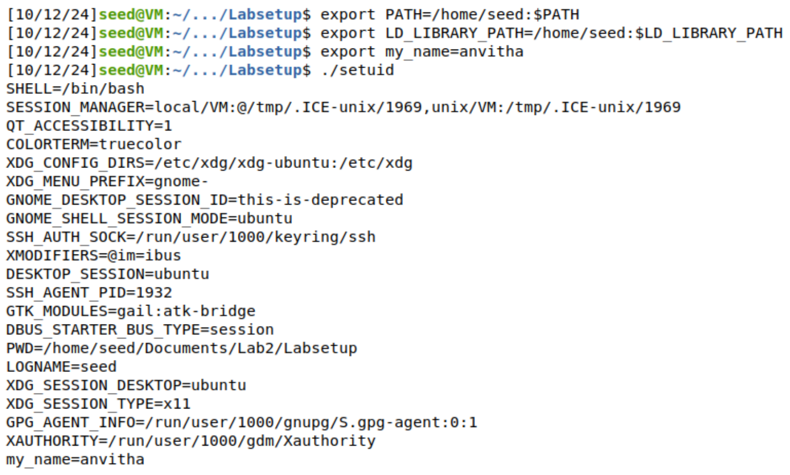
2.5 Task 5:

Step 1: A file named `setuid.c` was created, containing code to display the environmental variables.
Step 2: The `setuid.c` file was compiled, with the root user as the owner, and granted the necessary permissions for reading, writing, and executing.

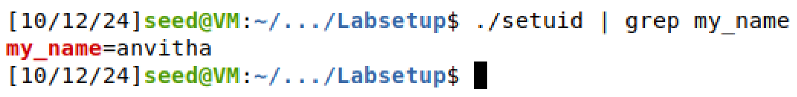


```
[10/12/24]seed@VM:~/.../Labsetup$ gedit setuid.c
[10/12/24]seed@VM:~/.../Labsetup$ gcc setuid.c -o setuid
[10/12/24]seed@VM:~/.../Labsetup$ sudo chown root setuid
[10/12/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 setuid
[10/12/24]seed@VM:~/.../Labsetup$ ls -l setuid
-rwsr-xr-x 1 root seed 16768 Oct 12 15:20 setuid
[10/12/24]seed@VM:~/.../Labsetup$
```

Step 3: an environment variable was established using the commands PATH, LD_LIBRARY_PATH, or any other specified name.



```
[10/12/24]seed@VM:~/.../Labsetup$ export PATH=/home/seed:$PATH
[10/12/24]seed@VM:~/.../Labsetup$ export LD_LIBRARY_PATH=/home/seed:$LD_LIBRARY_PATH
[10/12/24]seed@VM:~/.../Labsetup$ export my_name=anvitha
[10/12/24]seed@VM:~/.../Labsetup$ ./setuid
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1969,unix/VM:/tmp/.ICE-unix/1969
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1932
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Documents/Lab2/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
my_name=anvitha
```

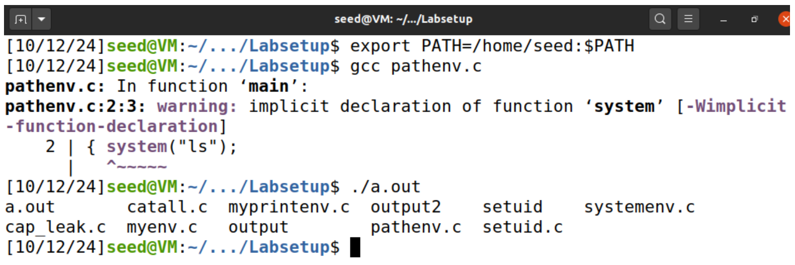


```
[10/12/24]seed@VM:~/.../Labsetup$ ./setuid | grep my_name
my_name=anvitha
[10/12/24]seed@VM:~/.../Labsetup$
```

Observation: This action allows for potential attacks on users by altering the variables.

2.6 Task 6:

I created a file called `pathenv.c` and compiled it.



seed@VM: ~/.../Labsetup

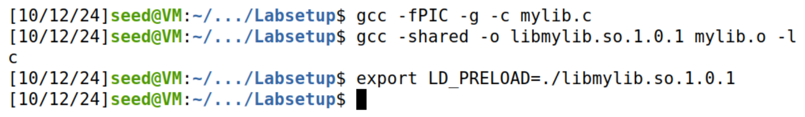
```
[10/12/24]seed@VM:~/.../Labsetup$ export PATH=/home/seed:$PATH
[10/12/24]seed@VM:~/.../Labsetup$ gcc pathenv.c
pathenv.c: In function ‘main’:
pathenv.c:2:3: warning: implicit declaration of function ‘system’ [-Wimplicit
-function-declaration]
    2 | { system("ls");
      |   ^~~~~~
[10/12/24]seed@VM:~/.../Labsetup$ ./a.out
a.out       catall.c  myprintenv.c  output2    setuid    systemenv.c
cap_leak.c  myenv.c   output        pathenv.c  setuid.c
[10/12/24]seed@VM:~/.../Labsetup$
```

The application is actually running the `ls` command in the current directory instead of executing `/bin/ls`. This method can be exploited by attackers, who may modify the code to take advantage of users. For example, an attacker could alter the file to steal a user's login credentials.

2.7 Task 7:

Step 1:

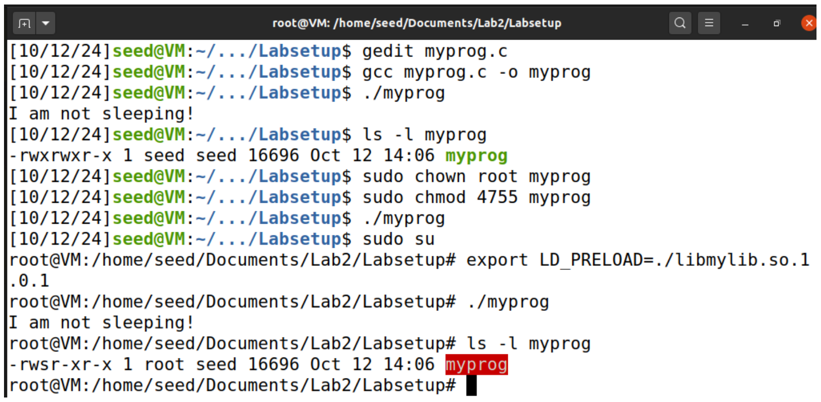
1. Developed a program and named it "mylib."
2. Compiled the program.
3. Configured the `LD_PRELOAD` environment variable.



```
[10/12/24]seed@VM:~/.../Labsetup$ gcc -fPIC -g -c mylib.c
[10/12/24]seed@VM:~/.../Labsetup$ gcc -shared -o libmylib.so.1.0.1 mylib.o -l
c
[10/12/24]seed@VM:~/.../Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[10/12/24]seed@VM:~/.../Labsetup$ 
```

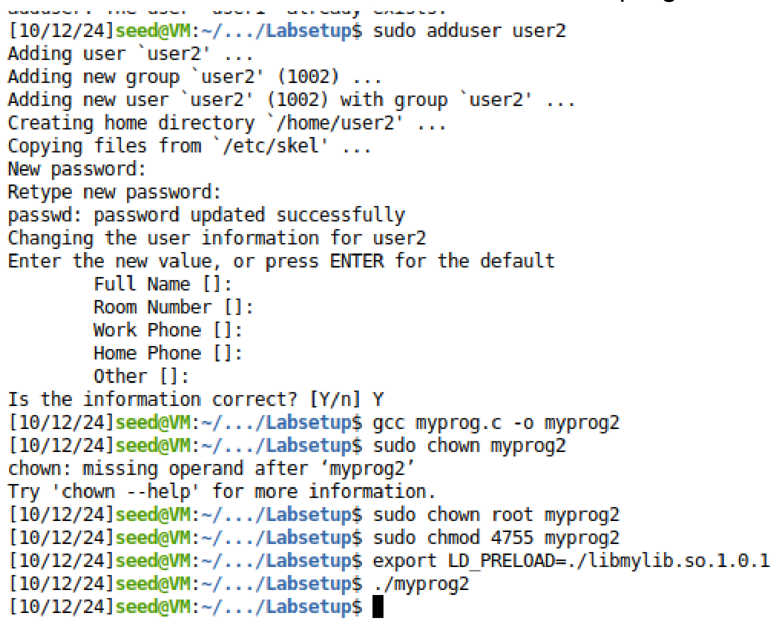
Step 2:

I developed a program named `myprog` and executed it with several conditions.



```
[10/12/24]seed@VM:~/.../Labsetup$ gedit myprog.c
[10/12/24]seed@VM:~/.../Labsetup$ gcc myprog.c -o myprog
[10/12/24]seed@VM:~/.../Labsetup$ ./myprog
I am not sleeping!
[10/12/24]seed@VM:~/.../Labsetup$ ls -l myprog
-rwxrwxr-x 1 seed seed 16696 Oct 12 14:06 myprog
[10/12/24]seed@VM:~/.../Labsetup$ sudo chown root myprog
[10/12/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog
[10/12/24]seed@VM:~/.../Labsetup$ ./myprog
[10/12/24]seed@VM:~/.../Labsetup$ sudo su
root@VM:/home/seed/Documents/Lab2/Labsetup# export LD_PRELOAD=./libmylib.so.1
.0.1
root@VM:/home/seed/Documents/Lab2/Labsetup# ./myprog
I am not sleeping!
root@VM:/home/seed/Documents/Lab2/Labsetup# ls -l myprog
-rwsr-xr-x 1 root seed 16696 Oct 12 14:06 myprog
root@VM:/home/seed/Documents/Lab2/Labsetup# 
```

I turned `myprog` into a Set-UID program for user2, then switched to a different user's account, set the `LD_PRELOAD` environment variable, and executed the program.

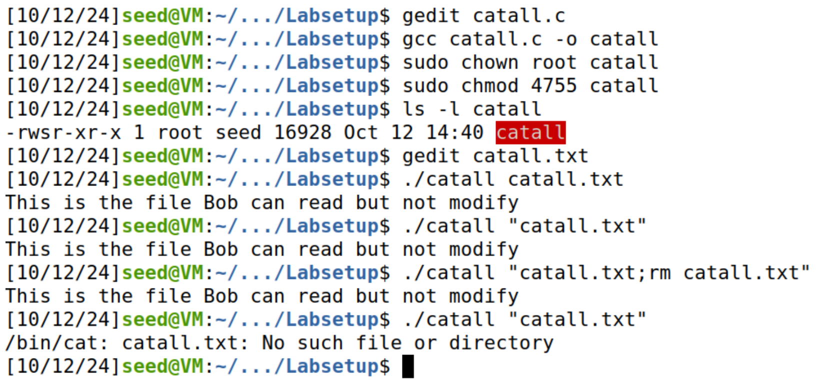


```
[10/12/24]seed@VM:~/.../Labsetup$ sudo adduser user2
Adding user `user2' ...
Adding new group `user2' (1002) ...
Adding new user `user2' (1002) with group `user2' ...
Creating home directory `/home/user2' ...
Copying files from `/etc/skel' ...
New password:
Retype new password:
passwd: password updated successfully
Changing the user information for user2
Enter the new value, or press ENTER for the default
        Full Name []:
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
[10/12/24]seed@VM:~/.../Labsetup$ gcc myprog.c -o myprog2
[10/12/24]seed@VM:~/.../Labsetup$ sudo chown myprog2
chown: missing operand after ‘myprog2’
Try 'chown --help' for more information.
[10/12/24]seed@VM:~/.../Labsetup$ sudo chown root myprog2
[10/12/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog2
[10/12/24]seed@VM:~/.../Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[10/12/24]seed@VM:~/.../Labsetup$ ./myprog2
[10/12/24]seed@VM:~/.../Labsetup$ 
```

Step 3: Observation: Child processes do not inherit the `LD_PRELOAD` environment variable. To execute the malicious script, the effective user ID must either be 0 or match the real user ID.

2.8 Task 8:

Step1:

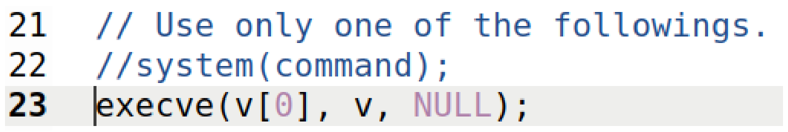


```
[10/12/24]seed@VM:~/.../Labsetup$ gedit catall.c
[10/12/24]seed@VM:~/.../Labsetup$ gcc catall.c -o catall
[10/12/24]seed@VM:~/.../Labsetup$ sudo chown root catall
[10/12/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 catall
[10/12/24]seed@VM:~/.../Labsetup$ ls -l catall
-rwsr-xr-x 1 root seed 16928 Oct 12 14:40 catall
[10/12/24]seed@VM:~/.../Labsetup$ gedit catall.txt
[10/12/24]seed@VM:~/.../Labsetup$ ./catall catall.txt
This is the file Bob can read but not modify
[10/12/24]seed@VM:~/.../Labsetup$ ./catall "catall.txt"
This is the file Bob can read but not modify
[10/12/24]seed@VM:~/.../Labsetup$ ./catall "catall.txt;rm catall.txt"
This is the file Bob can read but not modify
[10/12/24]seed@VM:~/.../Labsetup$ ./catall "catall.txt"
/bin/cat: catall.txt: No such file or directory
[10/12/24]seed@VM:~/.../Labsetup$ 
```

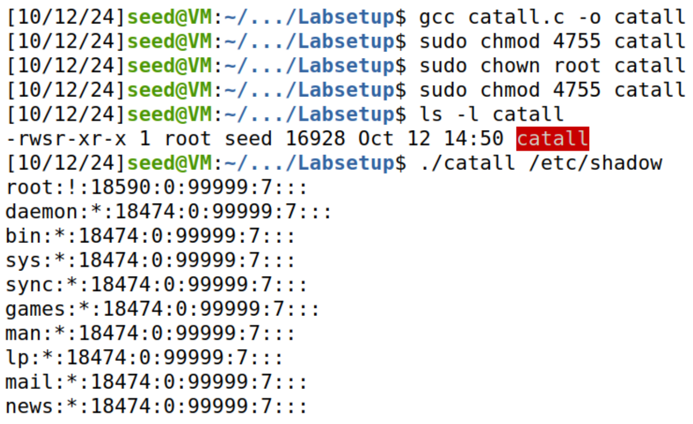
Observation: Bob has permission to read the file, but he is unable to modify it.

Step2:

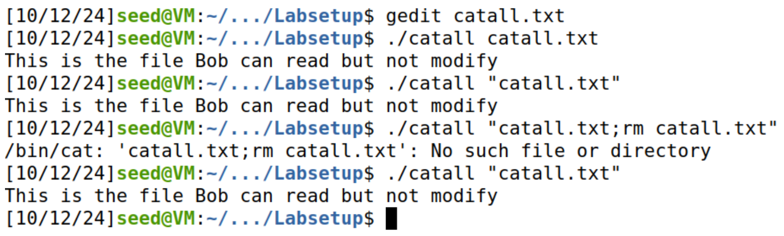
1. I commented out the `system(command)` statement and uncommented the `execve()` statement in the file.
2. After compiling it, I changed its permissions to `4755` and executed the file.



```
// Use only one of the followings.
//system(command);
execve(v[0], v, NULL);
```



```
[10/12/24]seed@VM:~/.../Labsetup$ gcc catall.c -o catall
[10/12/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 catall
[10/12/24]seed@VM:~/.../Labsetup$ sudo chown root catall
[10/12/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 catall
[10/12/24]seed@VM:~/.../Labsetup$ ls -l catall
-rwsr-xr-x 1 root seed 16928 Oct 12 14:50 catall
[10/12/24]seed@VM:~/.../Labsetup$ ./catall /etc/shadow
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
sync:*:18474:0:99999:7:::
games:*:18474:0:99999:7:::
man:*:18474:0:99999:7:::
lp:*:18474:0:99999:7:::
mail:*:18474:0:99999:7:::
news:*:18474:0:99999:7:::
```

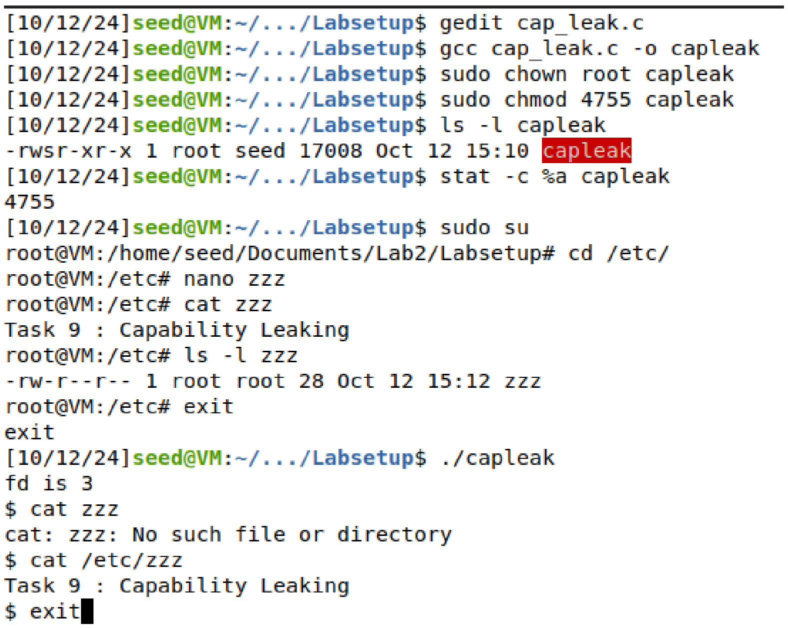


```
[10/12/24]seed@VM:~/.../Labsetup$ gedit catall.txt
[10/12/24]seed@VM:~/.../Labsetup$ ./catall catall.txt
This is the file Bob can read but not modify
[10/12/24]seed@VM:~/.../Labsetup$ ./catall "catall.txt"
This is the file Bob can read but not modify
[10/12/24]seed@VM:~/.../Labsetup$ ./catall "catall.txt;rm catall.txt"
/bin/cat: 'catall.txt;rm catall.txt': No such file or directory
[10/12/24]seed@VM:~/.../Labsetup$ ./catall "catall.txt"
This is the file Bob can read but not modify
[10/12/24]seed@VM:~/.../Labsetup$ 
```

Observation: When the `system` function is executed, it does not run the command directly. In the case of a setuid program, the user temporarily gains root privileges, allowing them to delete files. However, when using the `rm` command, user privileges revert, preventing file deletion in this context. The file was removed when using the `system()` function because the `execve()` function was unable to execute the `rm` command, leading to an error. Therefore, using `system()` in programs can pose significant security risks.

2.9 Task 9:

Step 1: I created a file named `/etc/zzz`.
Step 2: I changed its ownership to the root user and configured it as a Set-UID program.



```
[10/12/24]seed@VM:~/.../Labsetup$ gedit cap_leak.c
[10/12/24]seed@VM:~/.../Labsetup$ gcc cap_leak.c -o capleak
[10/12/24]seed@VM:~/.../Labsetup$ sudo chown root capleak
[10/12/24]seed@VM:~/.../Labsetup$ sudo chmod 4755 capleak
[10/12/24]seed@VM:~/.../Labsetup$ ls -l capleak
-rwsr-xr-x 1 root seed 17008 Oct 12 15:10 capleak
[10/12/24]seed@VM:~/.../Labsetup$ stat -c %a capleak
4755
[10/12/24]seed@VM:~/.../Labsetup$ sudo su
root@VM:/home/seed/Documents/Lab2/Labsetup# cd /etc/
root@VM:/etc# nano zzz
root@VM:/etc# cat zzz
Task 9 : Capability Leaking
root@VM:/etc# ls -l zzz
-rw-r--r-- 1 root root 28 Oct 12 15:12 zzz
root@VM:/etc# exit
exit
[10/12/24]seed@VM:~/.../Labsetup$ ./capleak
fd is 3
$ cat zzz
cat: zzz: No such file or directory
$ cat /etc/zzz
Task 9 : Capability Leaking
$ exit
```

Observation: This program contains a capability leaking vulnerability that can be exploited. The user IDs (UIDs) are all set to the effective UID of the seed, which is 1000, allowing us to write to the file. Because the application uses setuid, the user seed in this context lacks privileges. This allows attackers to manipulate private files in this manner.