Department of Computer Science

California State University, Channel Islands

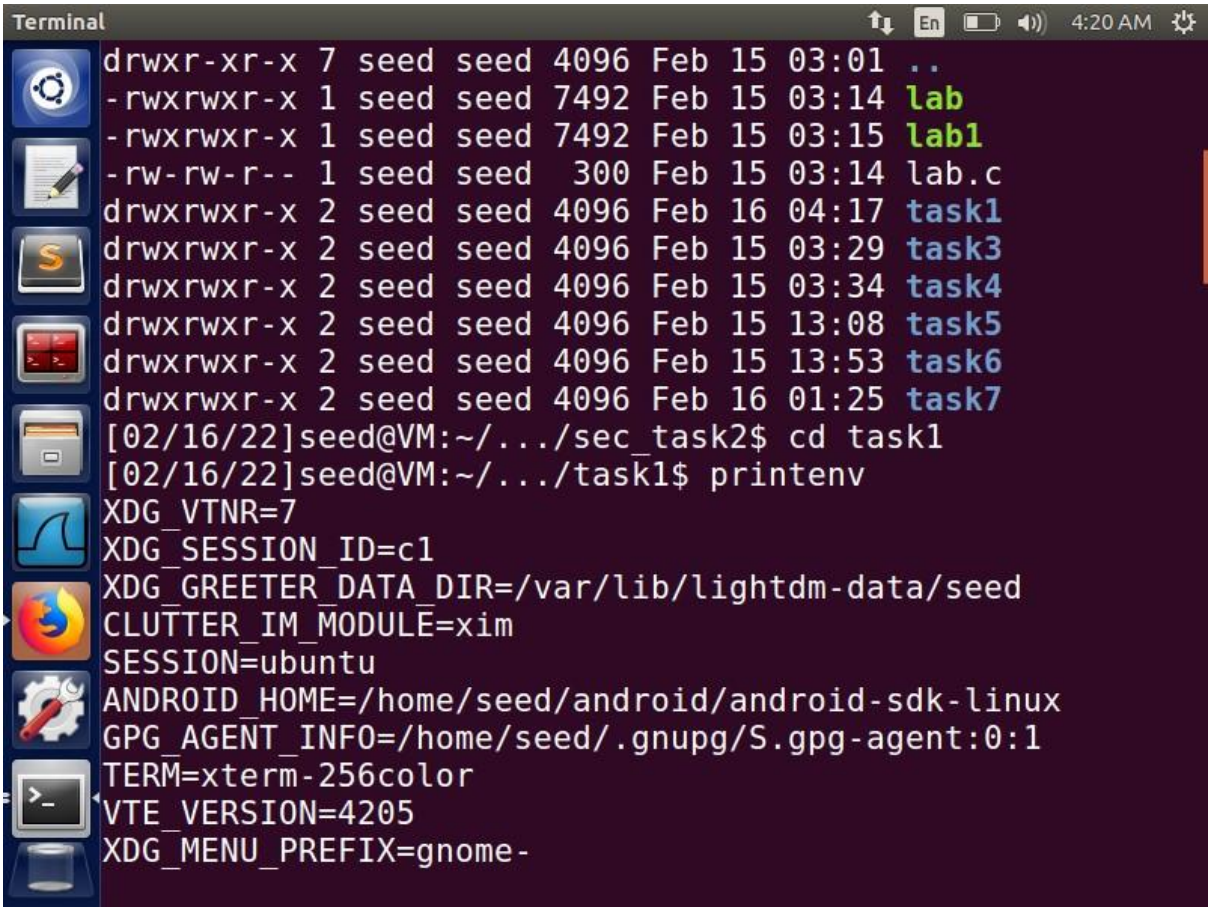
COMP-524: Security

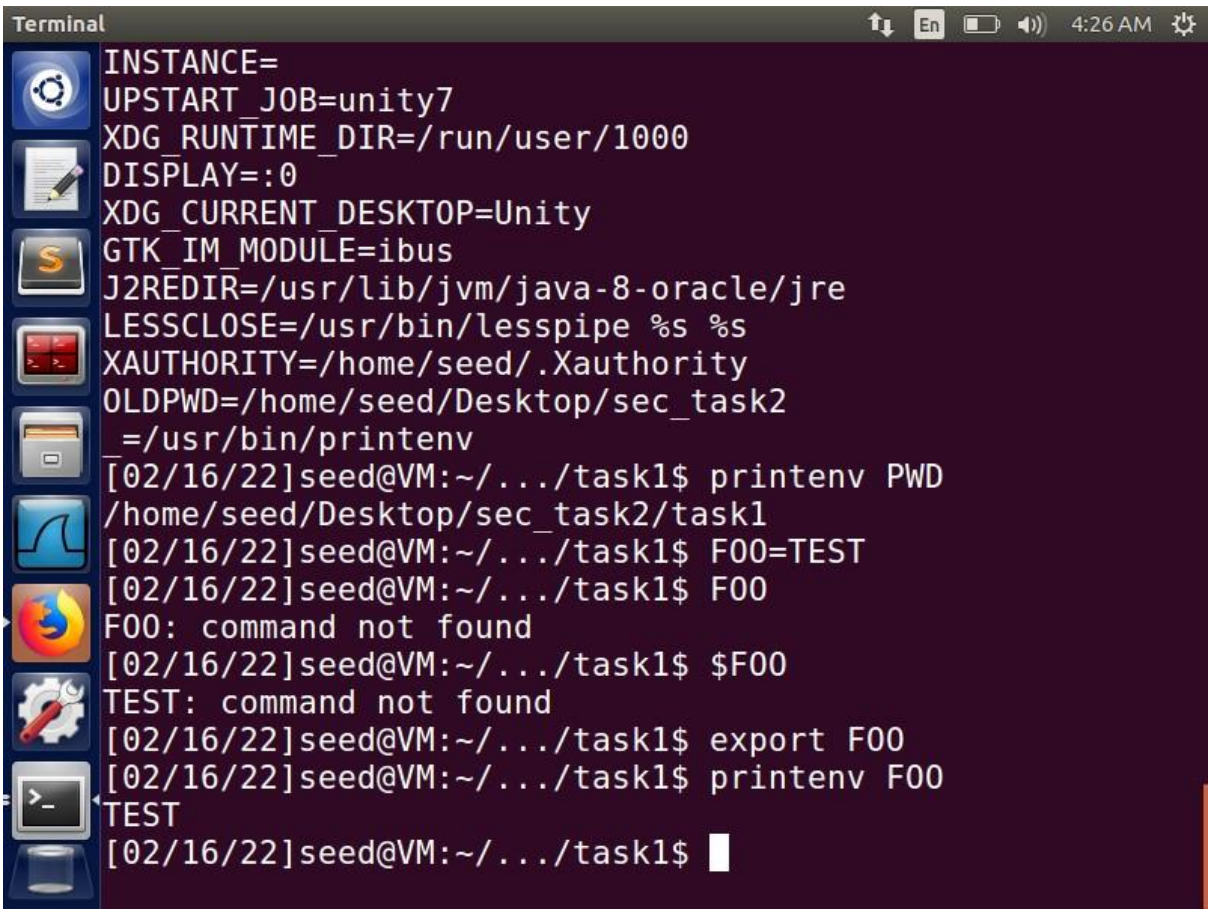
Lab Report

Lab Number: 3

Lab Topic: Environment Variable

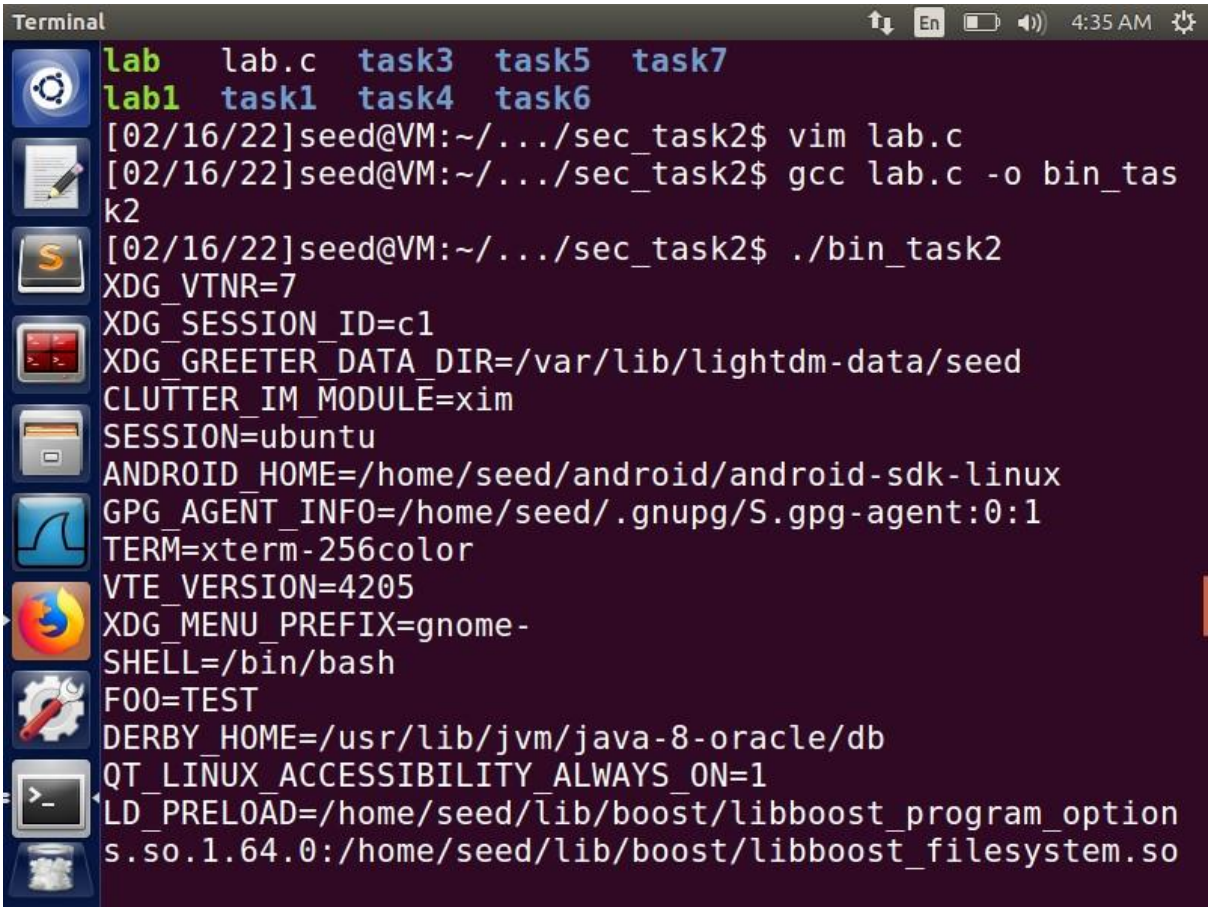
TASK - 01:


Terminal
4:20 AM
drwxr-xr-x 7 seed seed 4096 Feb 15 03:01 ..
-rwxrwxr-x 1 seed seed 7492 Feb 15 03:14 lab
-rwxrwxr-x 1 seed seed 7492 Feb 15 03:15 lab1
-rw-rw-r-- 1 seed seed 300 Feb 15 03:14 lab.c
drwxrwxr-x 2 seed seed 4096 Feb 16 04:17 task1
drwxrwxr-x 2 seed seed 4096 Feb 15 03:29 task3
drwxrwxr-x 2 seed seed 4096 Feb 15 03:34 task4
drwxrwxr-x 2 seed seed 4096 Feb 15 13:08 task5
drwxrwxr-x 2 seed seed 4096 Feb 15 13:53 task6
drwxrwxr-x 2 seed seed 4096 Feb 16 01:25 task7
[02/16/22]seed@VM:~/.../sec_task2$ cd task1
[02/16/22]seed@VM:~/.../task1$ printenv
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-



The printenv job prints all of the environment variables, and the export keyword causes the new environment variable to be exported. **We can try try with PWD and PATH env but syntax will be “printenv PWD” and “printenv PATH”.**

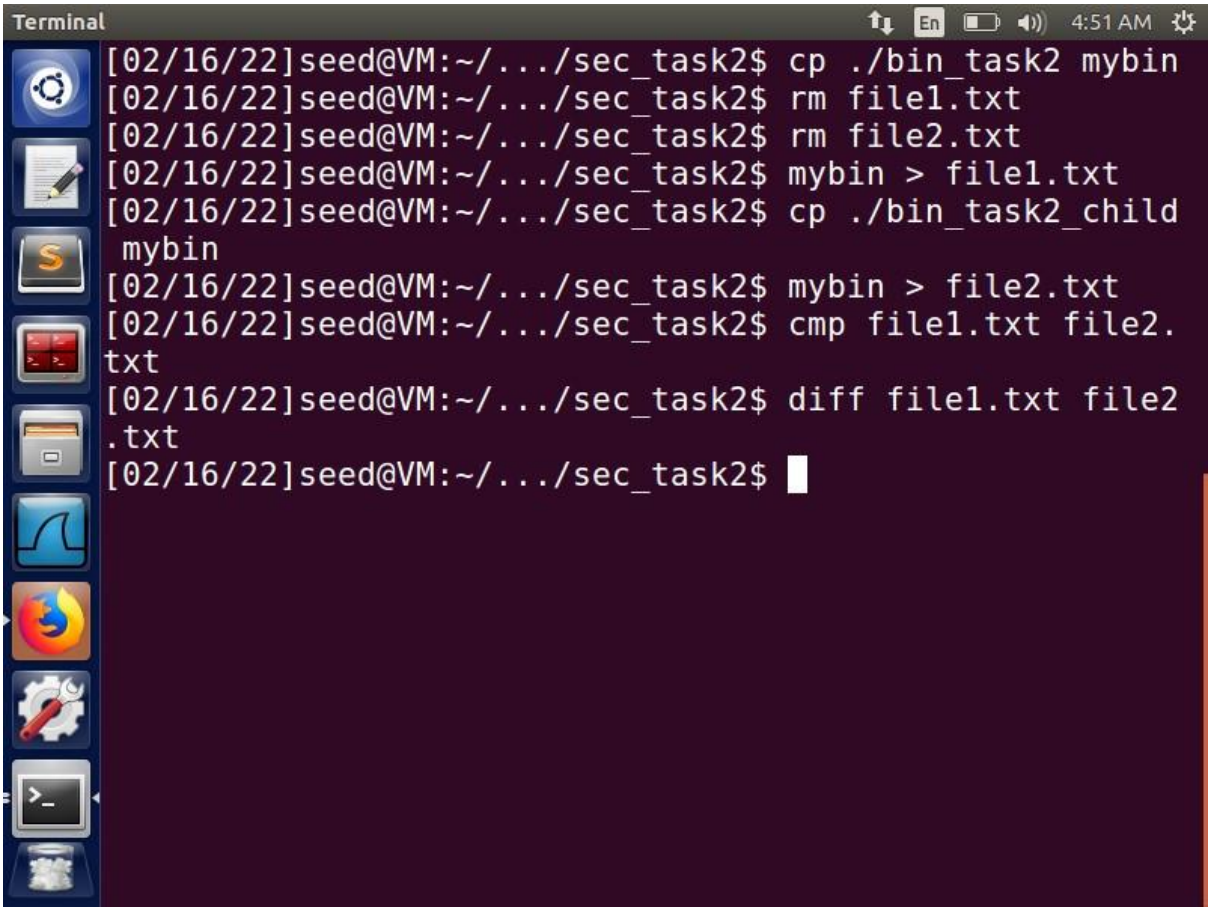
TASK - 02:



```
lab   lab.c  task3  task5  task7
lab1  task1  task4  task6
[02/16/22]seed@VM:~/.../sec_task2$ vim lab.c
[02/16/22]seed@VM:~/.../sec_task2$ gcc lab.c -o bin_tas
k2
[02/16/22]seed@VM:~/.../sec_task2$ ./bin_task2
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
FOO=TEST
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
```

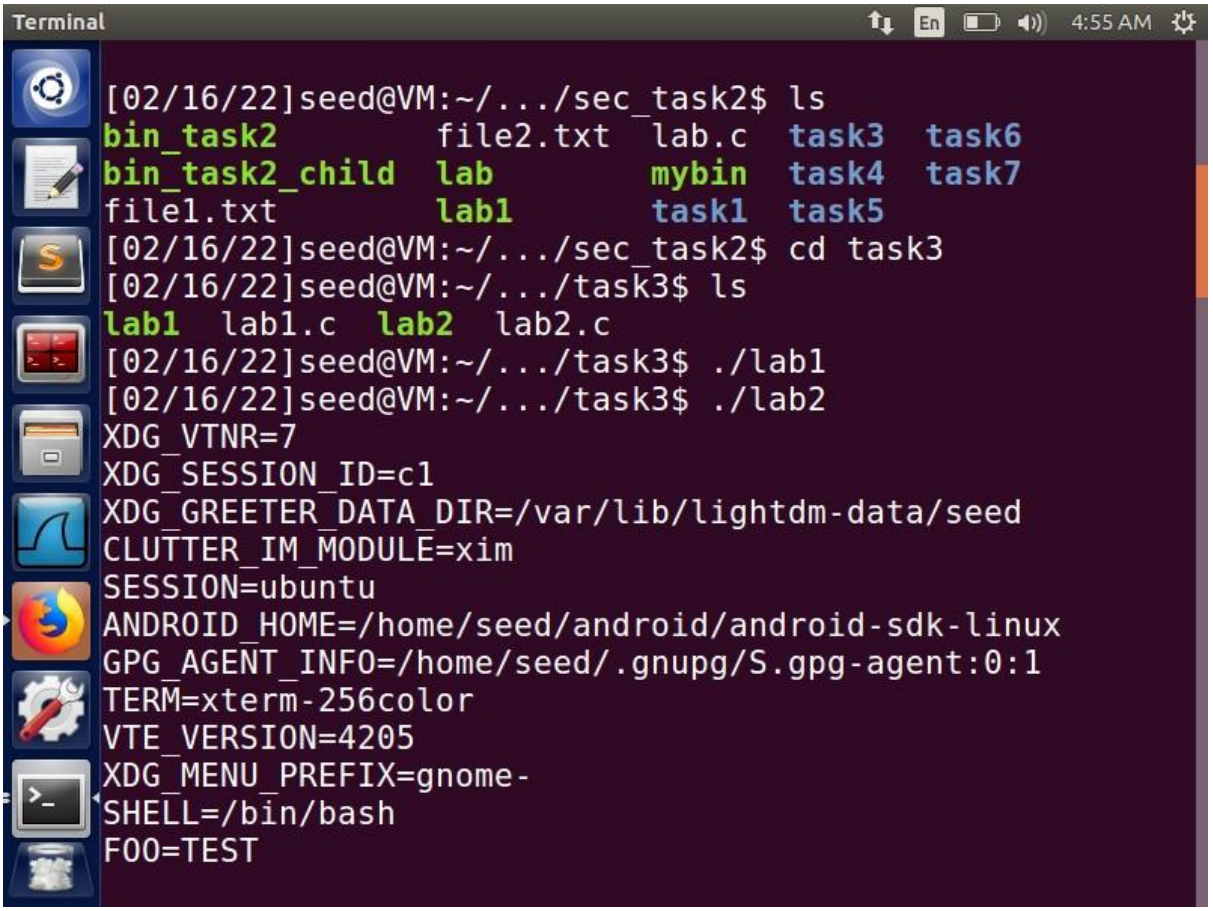
```
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
_=./bin_task2
[02/16/22]seed@VM:~/.../sec_task2$ vim lab.c
[02/16/22]seed@VM:~/.../sec_task2$ gcc lab.c -o bin_tas
k2_child
[02/16/22]seed@VM:~/.../sec_task2$ ./bin_task2_child
XDG_VTNR=7
XDG_SESSION_ID=c1
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
FOO=TEST
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
```

```
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/l
ocal/share/:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
UPSTART_JOB=unity7
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
XDG_CURRENT_DESKTOP=Unity
GTK_IM_MODULE=ibus
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
LESSCLOSE=/usr/bin/lesspipe %s %s
XAUTHORITY=/home/seed/.Xauthority
_=./bin_task2_child
[02/16/22]seed@VM:~/.../sec_task2$ ./bin_task2>file1.tx
t
[02/16/22]seed@VM:~/.../sec_task2$ ./bin_task2_child>fi
le2.txt
[02/16/22]seed@VM:~/.../sec_task2$ cmp file1.txt file2.
txt
file1.txt file2.txt differ: byte 4102, line 73
[02/16/22]seed@VM:~/.../sec_task2$
```



./bin task2 prints the environmental variables in the parent process, while./bin task2 child prints the environmental variables in the child process.
File1.txt and file2.txt contain the results of the child and parent process environment variables. Then use cmp or diff to compare the results.
In the end, nothing is written, indicating that the environment variables of the child and parent processes are the same. Both are the same.

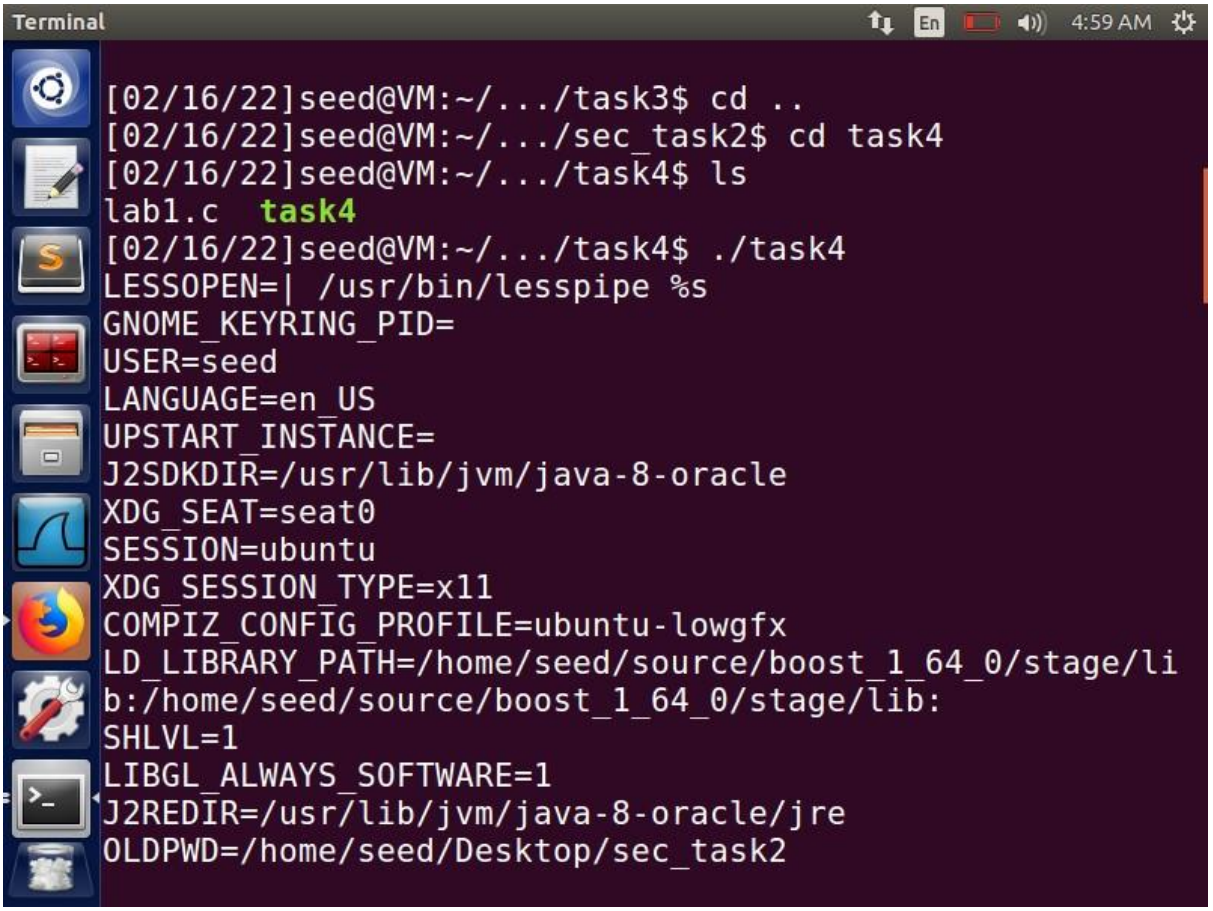
TASK - 03:



execve(): The execve function is used to overlay a process image formed by a call to the fork function. is the name of the file that contains the new process's executable image.

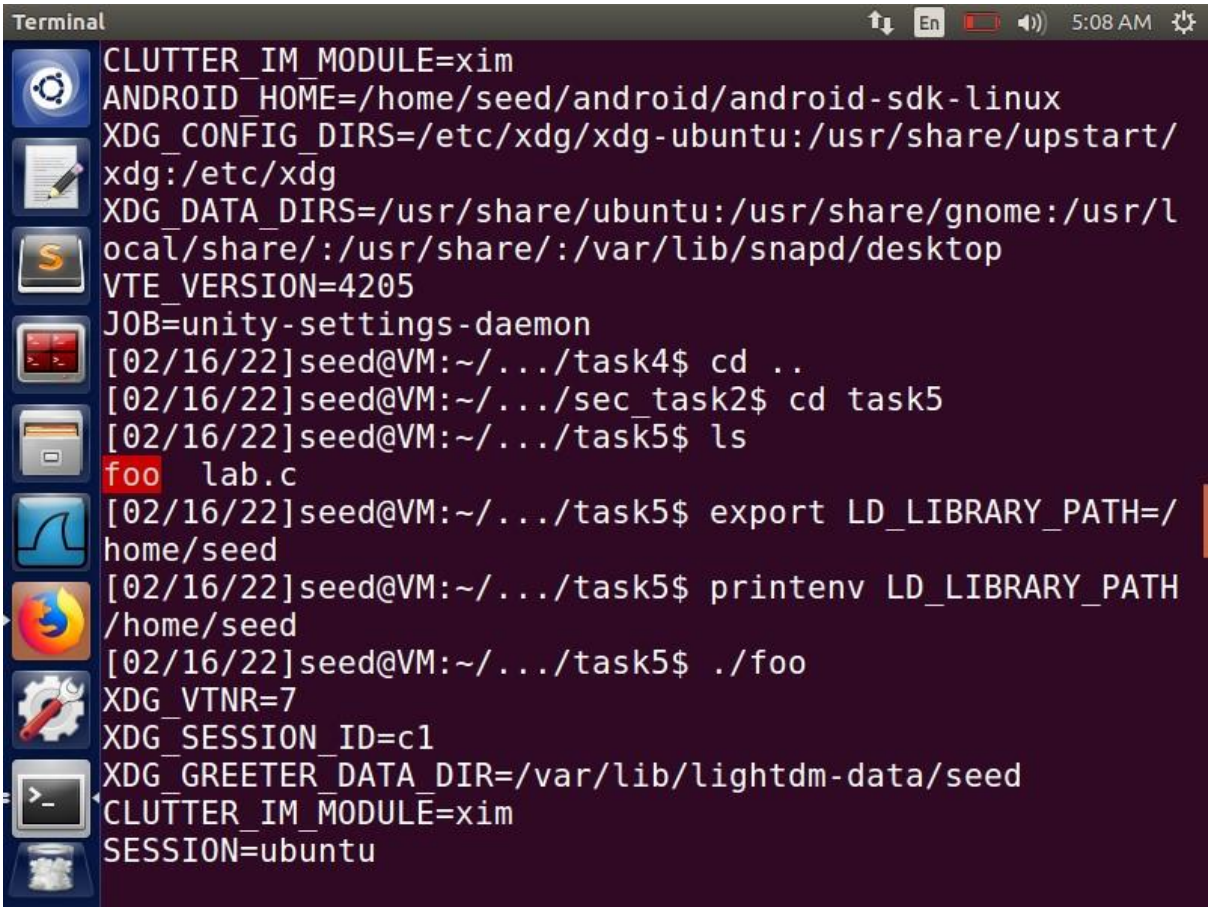
The third argument in the first step will be none, so the program will not print anything.
The third input in the second step is environ, which prints the environment variables, thus we need to manage what to provide on the shell command in execve. It's a secure method for a developer to employ, as it prevents shell syntax from accessing all of the environment variables.

TASK - 04:



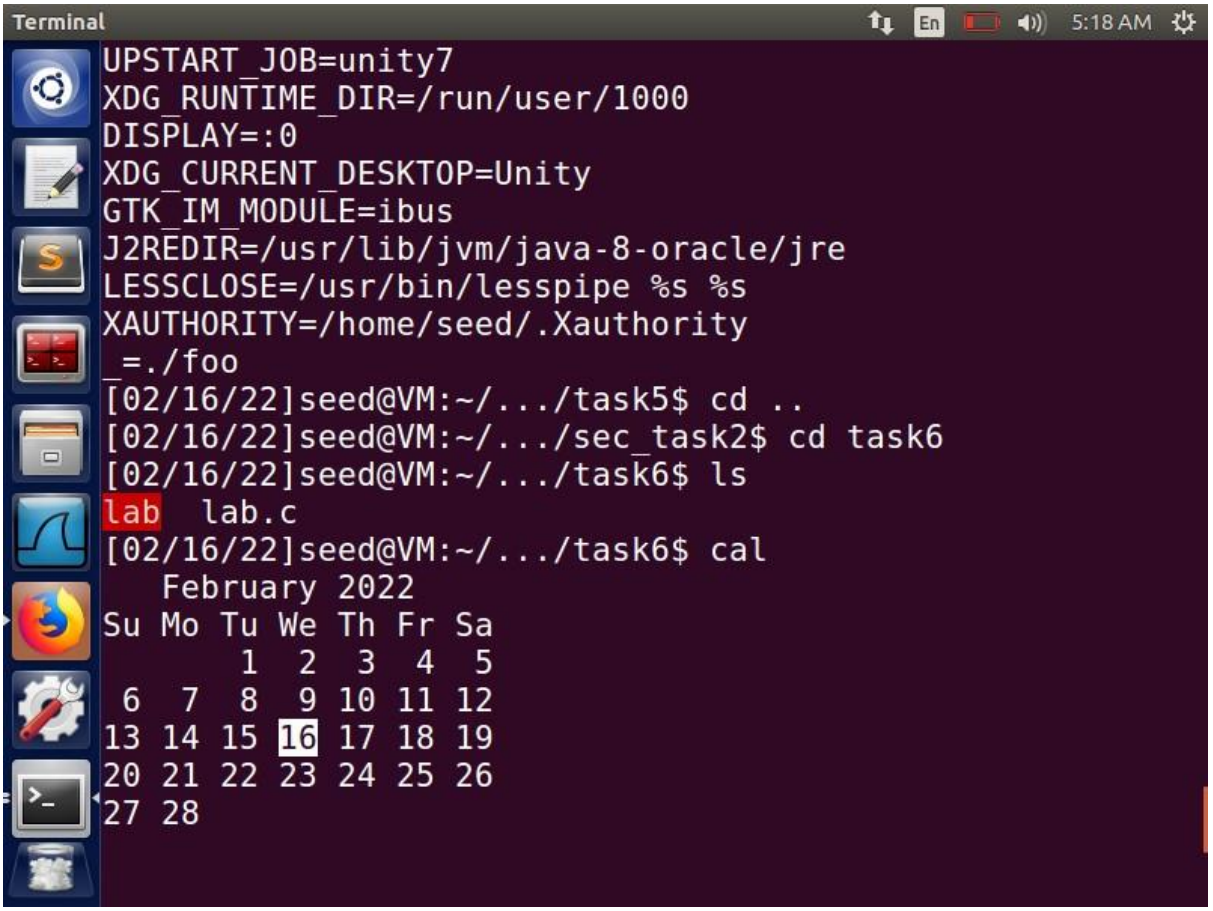
The variables in the environment are all printed out. If a developer does not wish to list all of the environmental variables, the exceve syntax is a safer option.

TASK - 05:

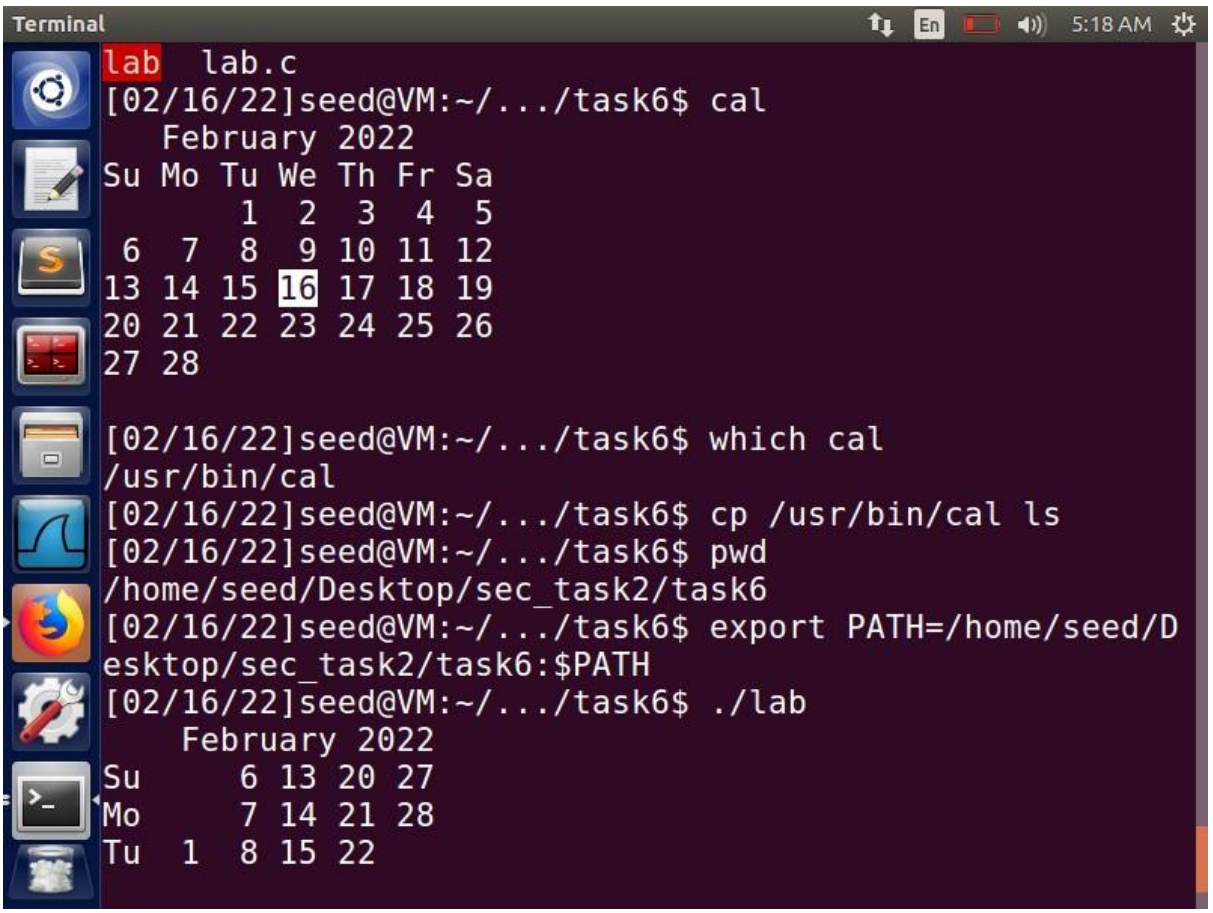


I converted a supplied program to setuid and added the env variable. LD LIBRARY PATH=/home/seed is exported to the entire env list.
The default library path is LD LIBRARY PATH, which is used to look for dynamic and shared libraries.

TASK - 06:



Created a setuid program from a provided C program. The cal command displays the current calendar in the terminal.



which cal gives it's path transferring the current path to ls path Instead of listing, it will run the cal program. If it runs with root access, it can change the default commands to malicious commands, as illustrated.

TASK - 07:

```
Terminal                                                    2:09 PM
[02/16/22]seed@VM:~/.../task3$ cd ..
[02/16/22]seed@VM:~/.../sec_task2$ cd task7
[02/16/22]seed@VM:~/.../task7$ gcc -fPIC -g -c mylib.c
[02/16/22]seed@VM:~/.../task7$ ls mylib*
mylib.c  mylib.o
[02/16/22]seed@VM:~/.../task7$ gcc -shared -o libmylib.
so.1.0.1 mylib.o -lc
[02/16/22]seed@VM:~/.../task7$ export LD_PRELOAD=./libm
ylib.so.1.0.1
[02/16/22]seed@VM:~/.../task7$ printenv LD_PRELOAD
./libmylib.so.1.0.1
```

```
Terminal                                                    2:12 PM
[02/16/22]seed@VM:~/.../task7$ sudo chown root mylib.o
[02/16/22]seed@VM:~/.../task7$ sudo chmod u+s mylib.o
[02/16/22]seed@VM:~/.../task7$ ls -la mylib.o
-rwSrw-r-- 1 root seed 2600 Feb 16 14:07 mylib.o
```

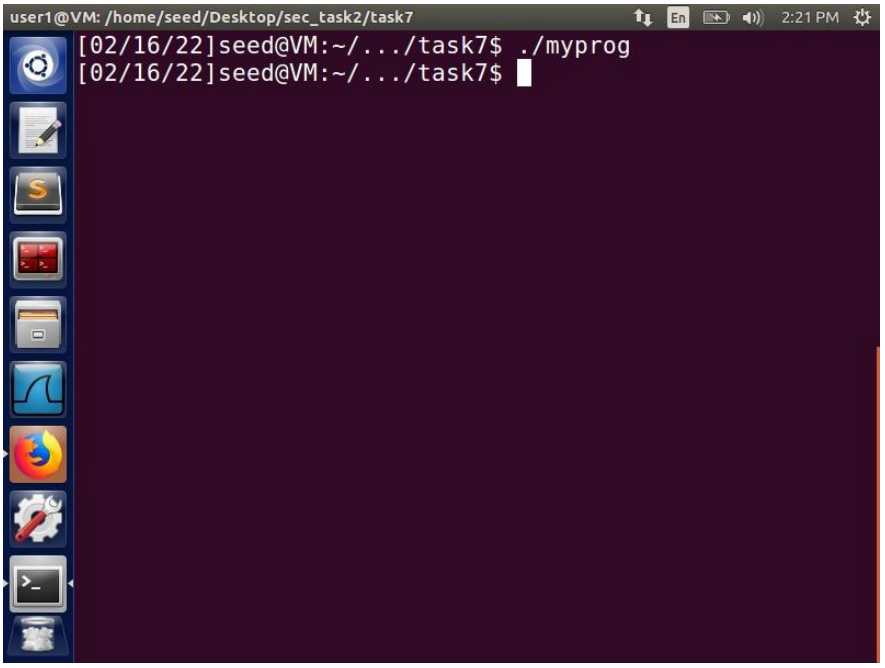
user1@VM: /home/seed/Desktop/sec_task2/task7

```
[02/16/22]seed@VM:~/.../task7$ sudo adduser user1
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
        Full Name []: user1
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
[02/16/22]seed@VM:~/.../task7$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),
24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128
(sambashare)
[02/16/22]seed@VM:~/.../task7$ sudo adduser user1 sudo
```

```
user1@VM: /home/seed/Desktop/sec_task2/task7                    2:17 PM
        Full Name []: user1
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n] Y
[02/16/22]seed@VM:~/.../task7$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),
24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128
(sambashare)
[02/16/22]seed@VM:~/.../task7$ sudo adduser user1 sudo
Adding user `user1' to group `sudo' ...
Adding user user1 to group sudo
Done.
[02/16/22]seed@VM:~/.../task7$ sudo newgrp
[02/16/22]root@VM:~/.../task7# sudo su seed
[02/16/22]seed@VM:~/.../task7$ sudo su user1
To run a command as administrator (user "root"), use "s
udo <command>".
See "man sudo_root" for details.

user1@VM:/home/seed/Desktop/sec_task2/task7$
```

```
user1@VM: /home/seed/Desktop/sec_task2/task7                    2:20 PM
[02/16/22]seed@VM:~/.../task7$ gcc -o myprog mylib.c
/usr/lib/gcc/i686-linux-gnu/5/../../../i386-linux-gnu/c
rt1.o: In function `_start':
(.text+0x18): undefined reference to `main'
collect2: error: ld returned 1 exit status
[02/16/22]seed@VM:~/.../task7$ gcc -o myprog myprog.c
myprog.c: In function ‘main’:
myprog.c:5:1: warning: implicit declaration of function
 ‘sleep’ [-Wimplicit-function-declaration]
 sleep(1);
 ^
[02/16/22]seed@VM:~/.../task7$ sudo chown user1 myprog
[02/16/22]seed@VM:~/.../task7$ sudo chmod u+s myprog
[02/16/22]seed@VM:~/.../task7$ ls -la myprog
-rwsrwxr-x 1 user1 seed 7348 Feb 16 14:18 myprog
```



The four scenarios are as follows:

1. Run a normal program by a normal user - The sleep() method is overridden when a normal user creates, compiles, and runs a program. "I am not asleep" is the message we receive.
2. Normal user runs set-uid to root program - myprog.c is compiled by root and the executable is set-uid, but the sleep() function is not overwritten. After a one-second pause, the prompt appears again.
3. Root user runs set-uid root program - myprog.c was compiled by root and the executable is set-uid, therefore the sleep() method isn't overwritten. After a one-second pause, the prompt appears again. However, if the root user additionally exports LD PRELOAD, the sleep() method is rewritten, resulting in the message "I am not sleeping."
4. When a non-root user calls a set-uid user program, the sleep() method is not overridden because myprog.c was compiled by a non-root user and the executable is set-uid.

   There is a one-second pause before prompt reappears.

As a result, if the real uid of the user id differs from the same user's effective uid, the linker ignores the LD PRELOAD environment setting. This serves as a safeguard against set-uid assaults.