CSE: 5382-001: SECURE PROGRAMMING

ASSIGNMENT 1

Tharoon T Thiagarajan

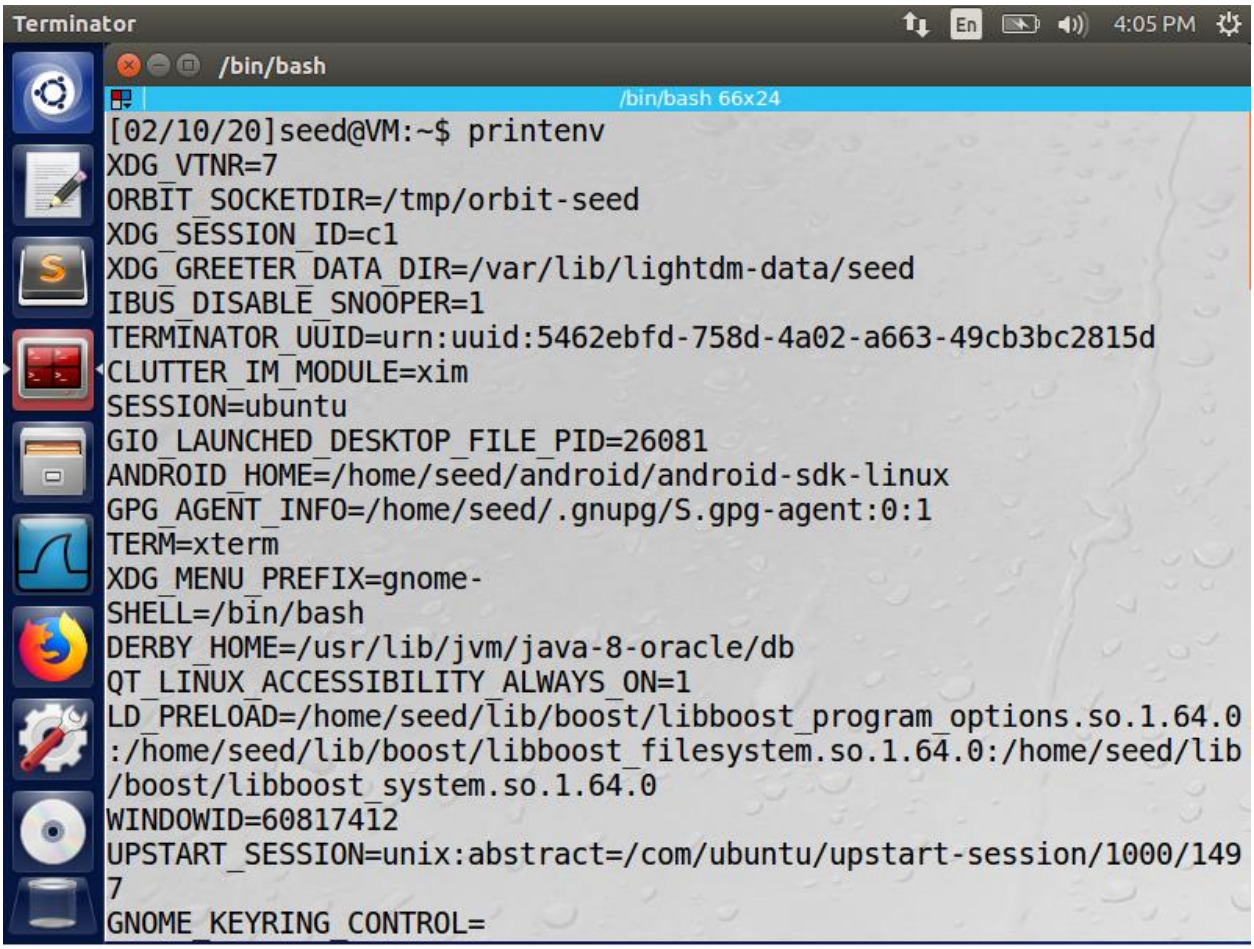
1001704601

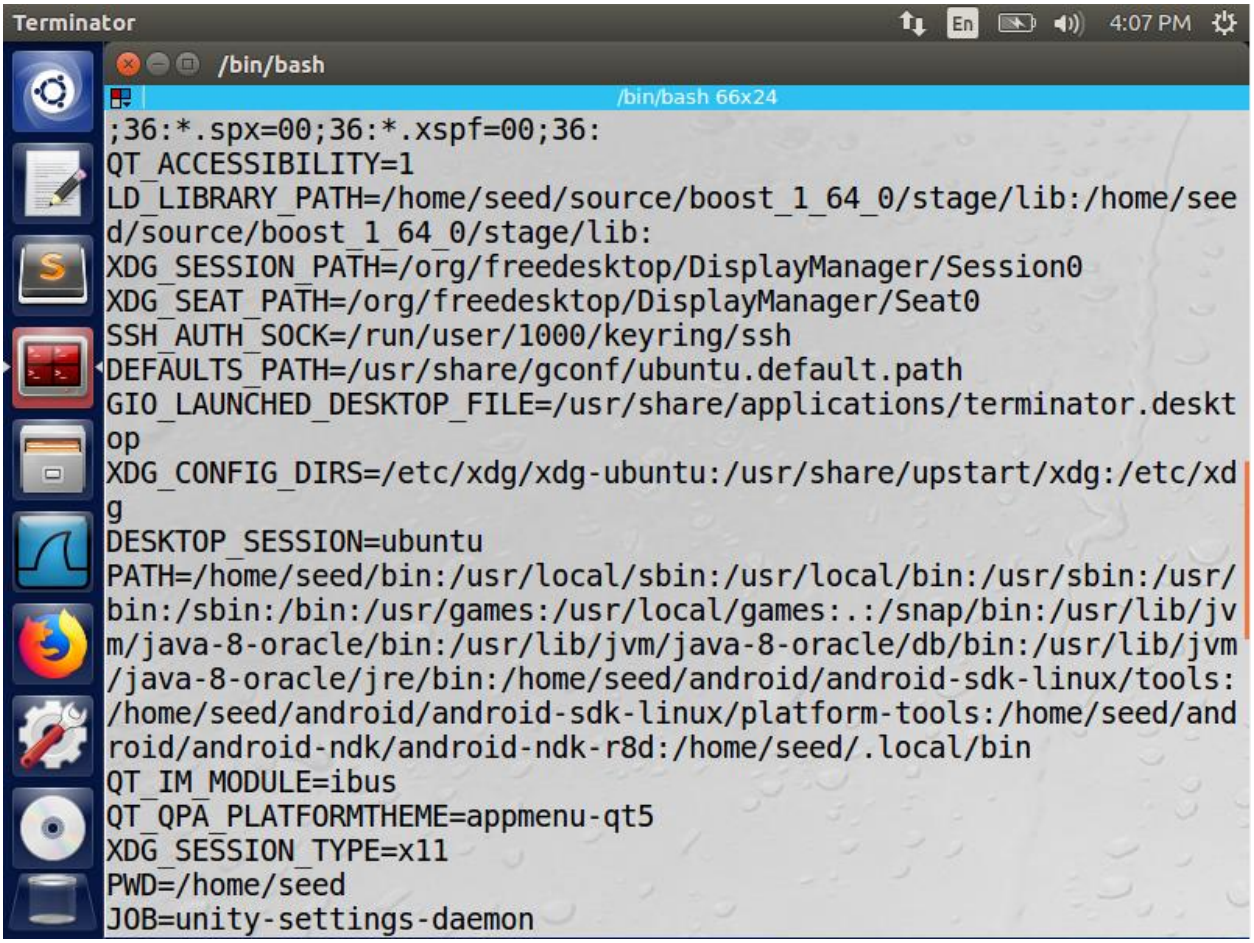
Task 1: Manipulating Environment Variables

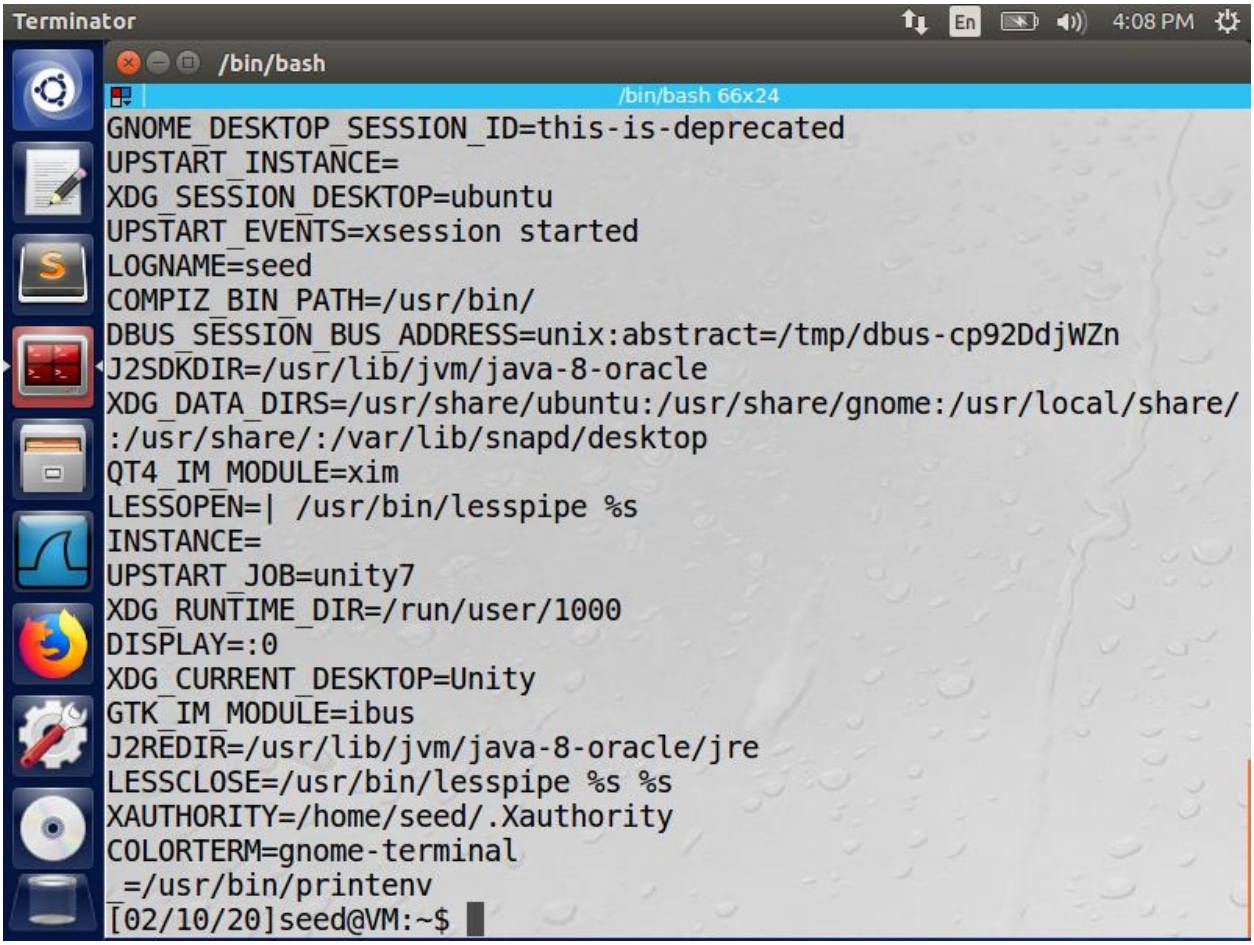
Step 1:

Output:

printenv or env command is used to print all environment variables.

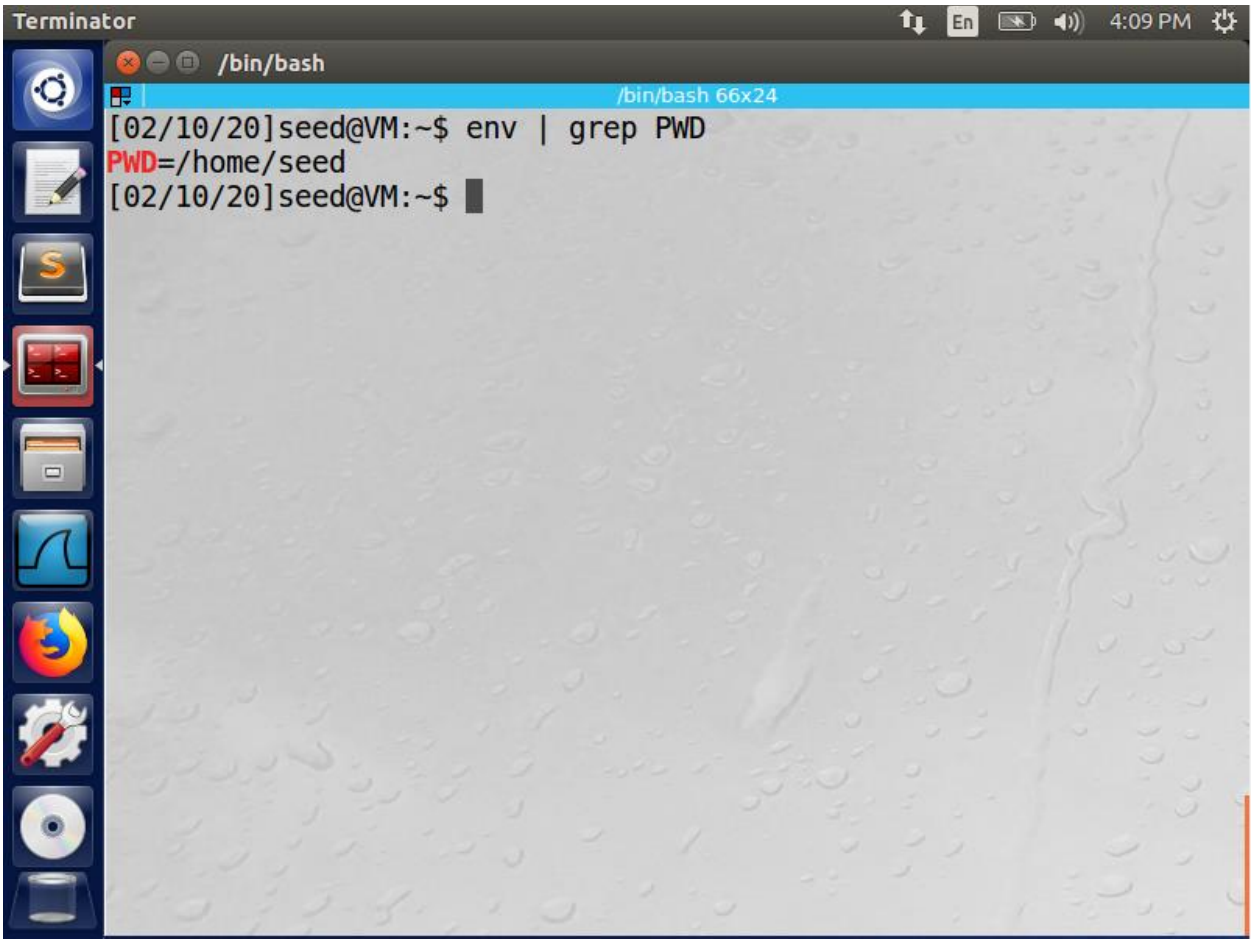



Terminator
En
4:07 PM
/bin/bash
/bin/bash 66x24
;36:*.spx=00;36:*.xspf=00;36:
QT_ACCESSIBILITY=1
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
GIO_LAUNCHED_DESKTOP_FILE=/usr/share/applications/terminator.desktop
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
DESKTOP_SESSION=ubuntu
PATH=/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/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
PWD=/home/seed
JOB=unity-settings-daemon


Terminator
En
4:08 PM
/bin/bash
/bin/bash 66x24
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
UPSTART_INSTANCE=
XDG_SESSION_DESKTOP=ubuntu
UPSTART_EVENTS=xsession started
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-cp92DdjWZn
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/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
COLORTERM=gnome-terminal
_=/usr/bin/printenv
[02/10/20]seed@VM:~$

Using grep:

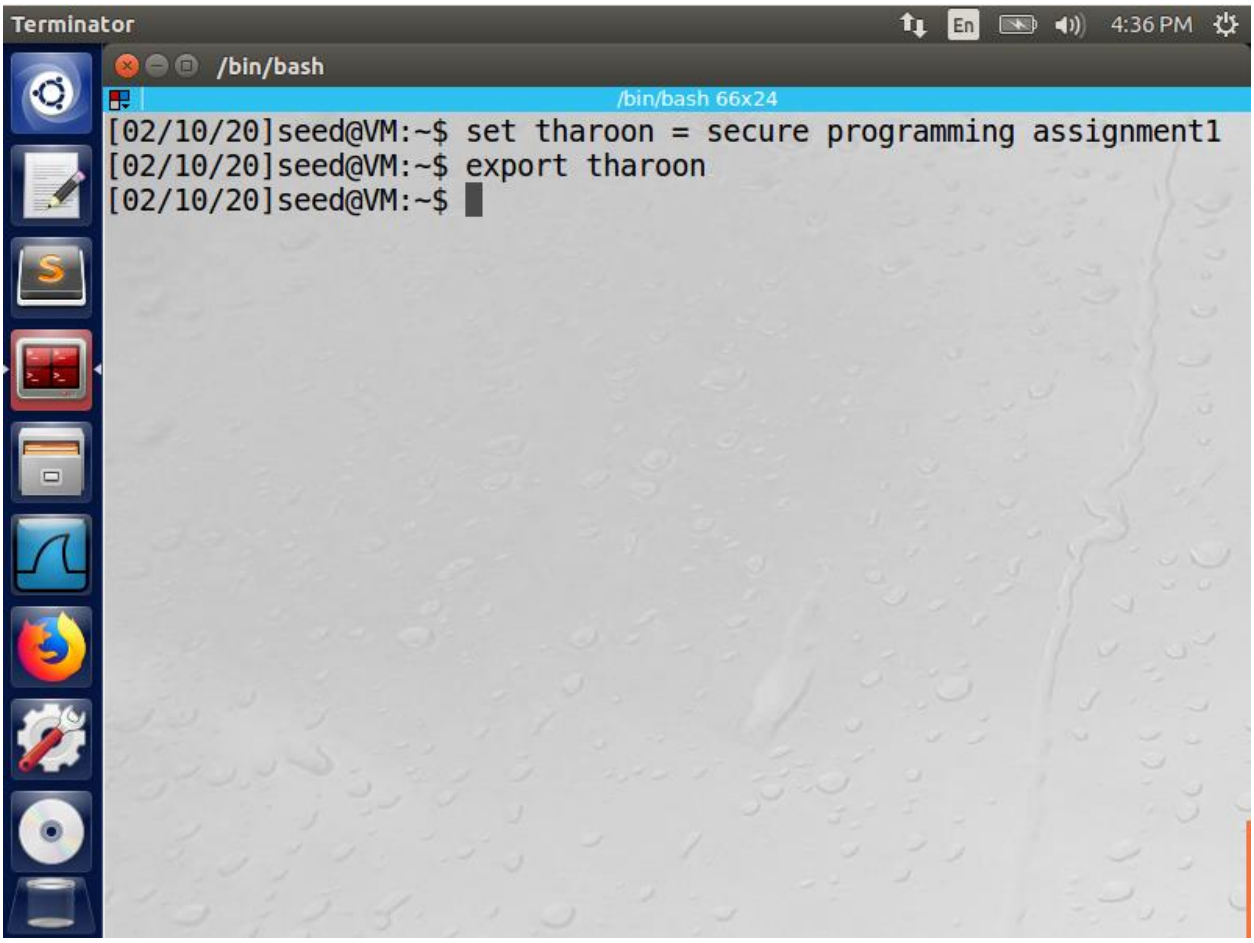
By using grep, I am able to view the particular directory.



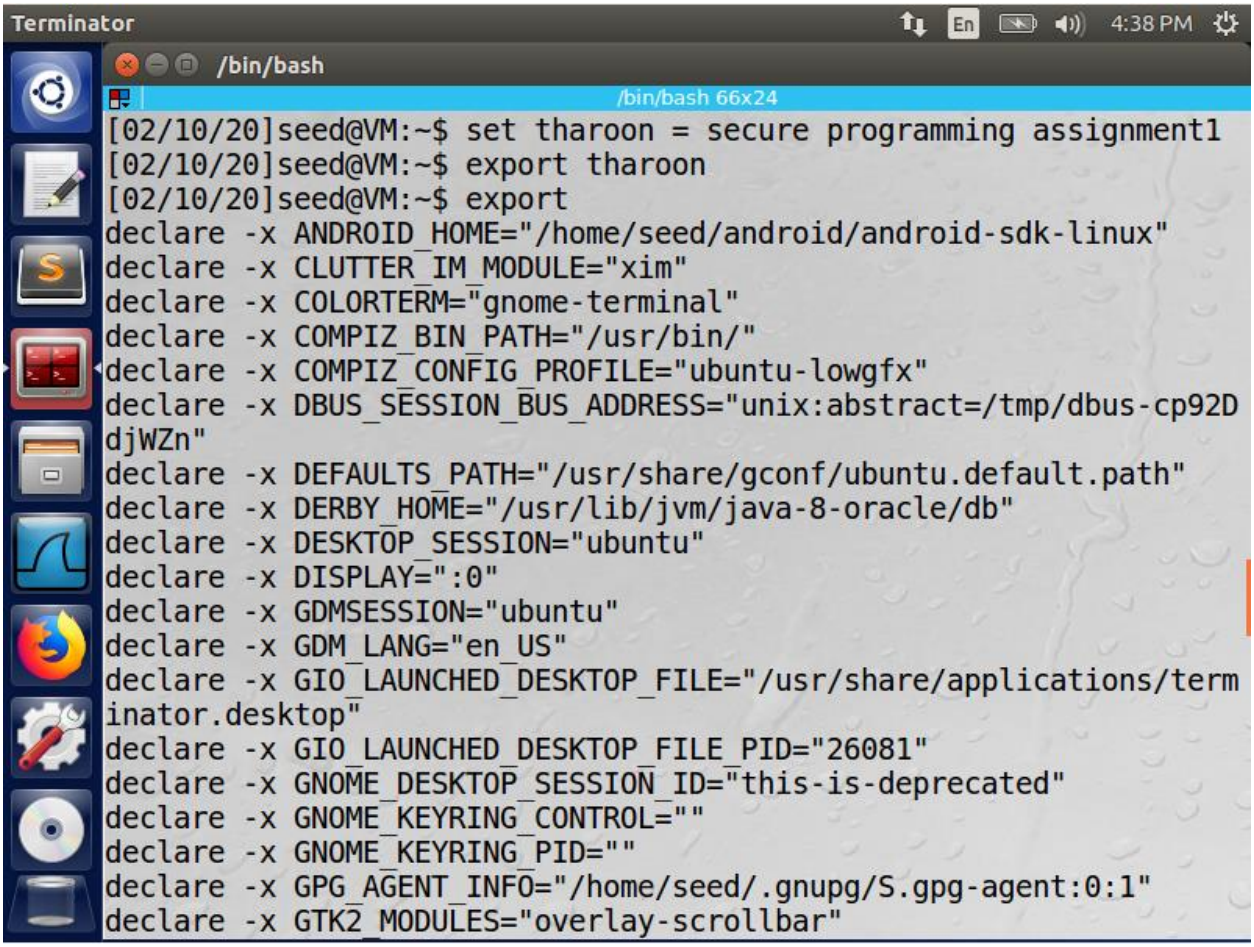
Step 2:

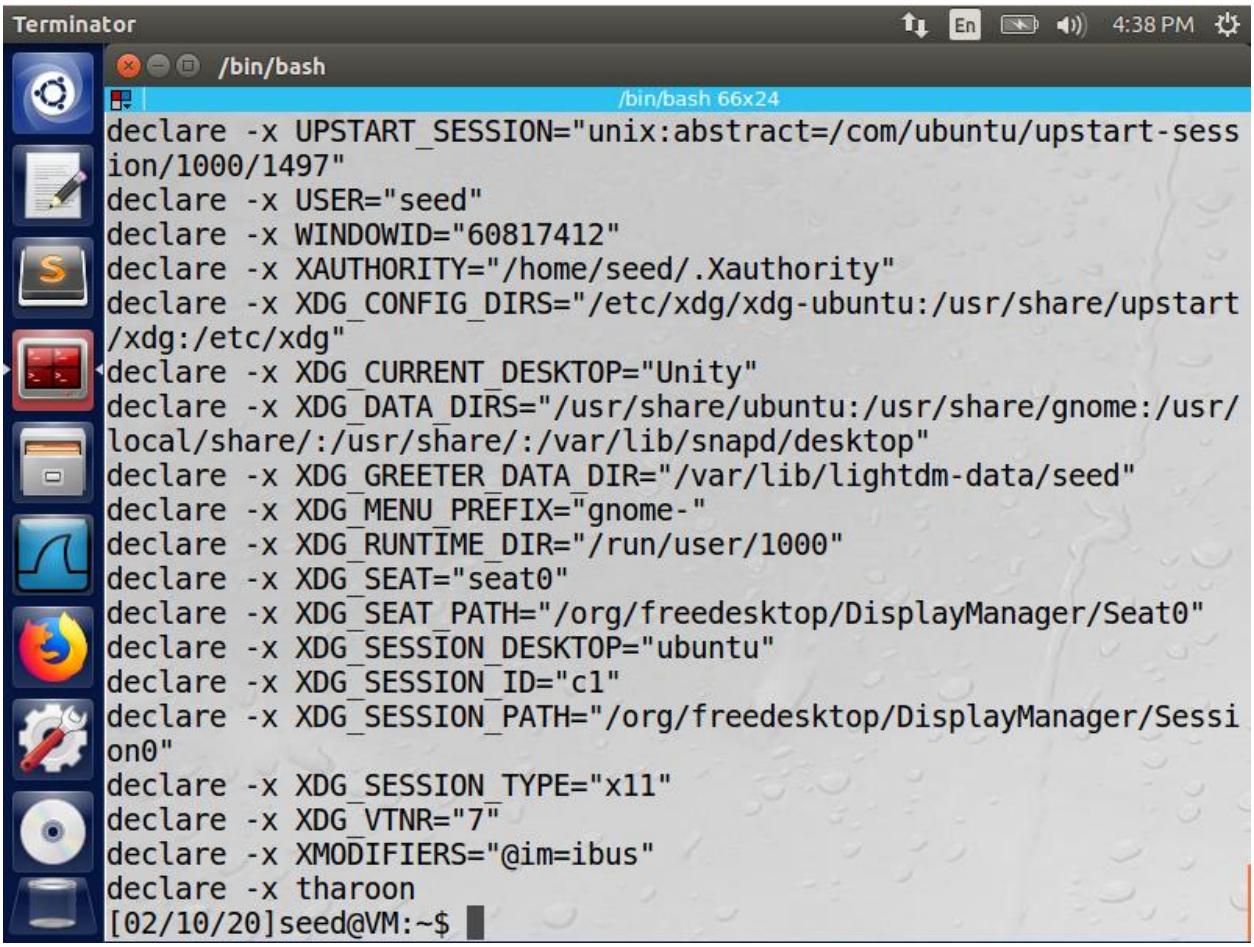
Output:

We use the set command to create an environment variable in the bash. Then, we use export command to set the newly created environment variable.

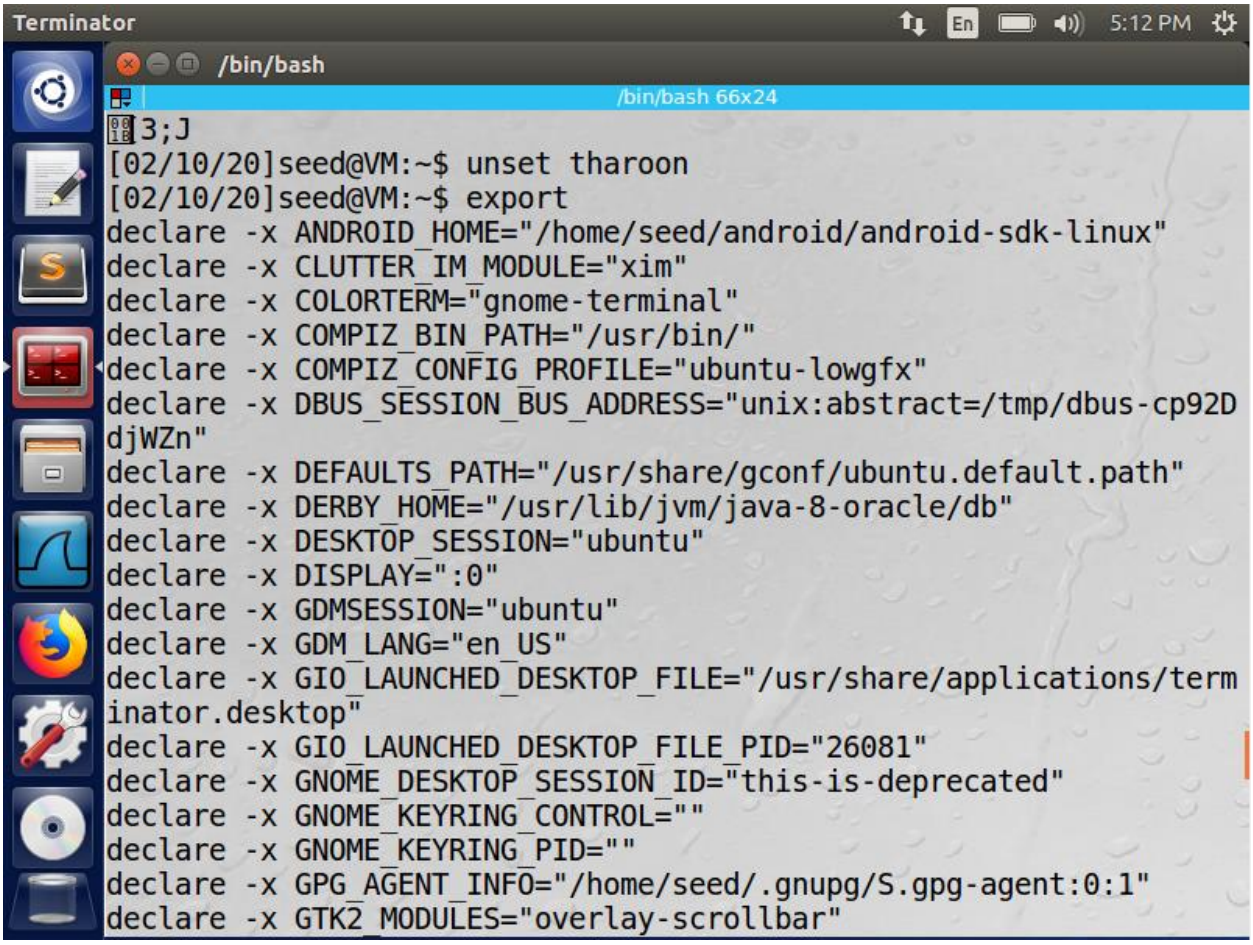


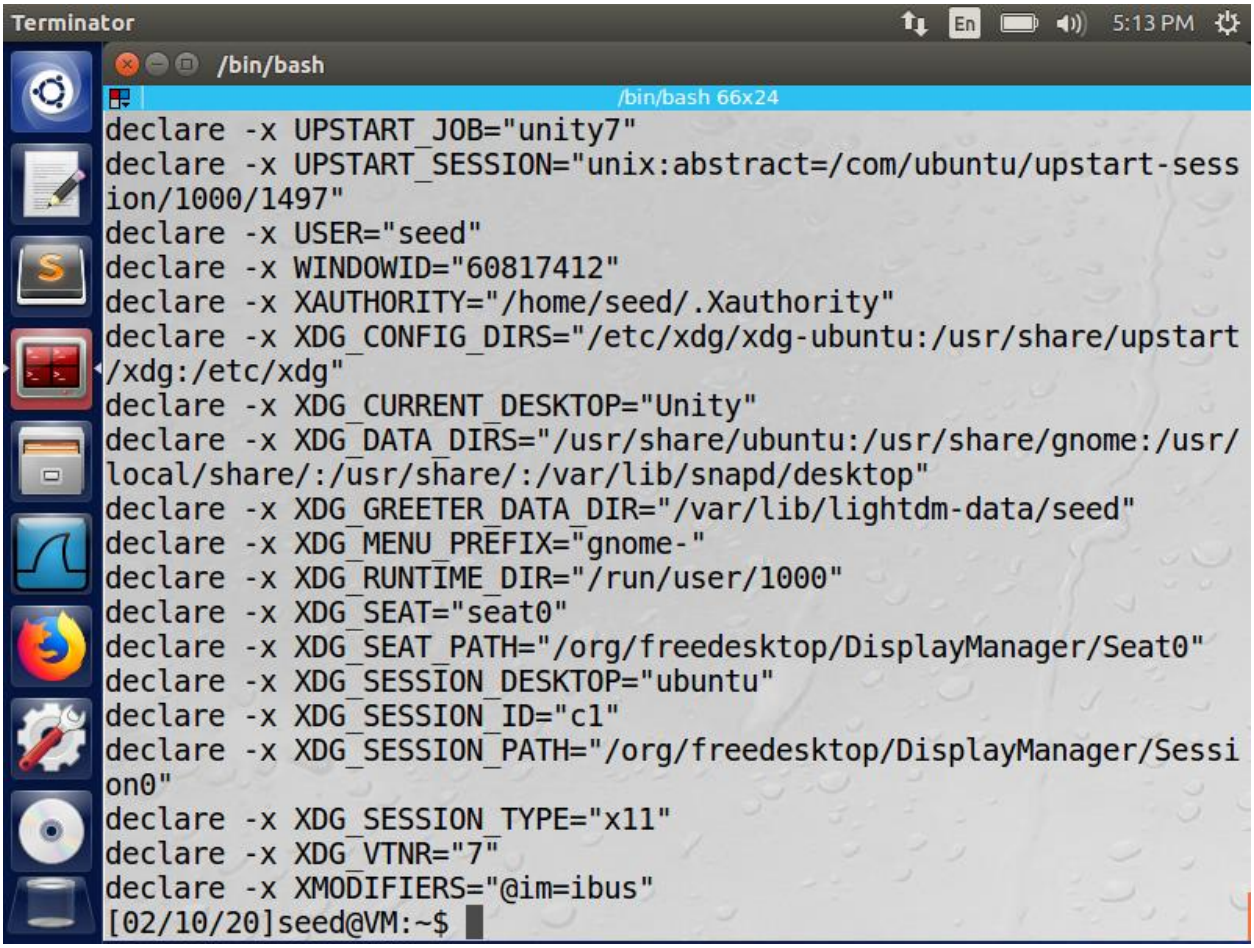
Now we use the export command to check if the newly created environment variable is set to the operating system. We will be able to see the newly created environment variable.




Terminator
En
4:38 PM
/bin/bash
/bin/bash 66x24
declare -x UPSTART_SESSION="unix:abstract=/com/ubuntu/upstart-session/1000/1497"
declare -x USER="seed"
declare -x WINDOWID="60817412"
declare -x XAUTHORITY="/home/seed/.Xauthority"
declare -x XDG_CONFIG_DIRS="/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg"
declare -x XDG_CURRENT_DESKTOP="Unity"
declare -x XDG_DATA_DIRS="/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop"
declare -x XDG_GREETER_DATA_DIR="/var/lib/lightdm-data/seed"
declare -x XDG_MENU_PREFIX="gnome-"
declare -x XDG_RUNTIME_DIR="/run/user/1000"
declare -x XDG_SEAT="seat0"
declare -x XDG_SEAT_PATH="/org/freedesktop/DisplayManager/Seat0"
declare -x XDG_SESSION_DESKTOP="ubuntu"
declare -x XDG_SESSION_ID="c1"
declare -x XDG_SESSION_PATH="/org/freedesktop/DisplayManager/Session0"
declare -x XDG_SESSION_TYPE="x11"
declare -x XDG_VTNR="7"
declare -x XMODIFIERS="@im=ibus"
declare -x tharoon
[02/10/20]seed@VM:~$

Now we will unset the environment variable which we created in the above step, using the unset command. Now if we use export command to look for environment variable, we will not be able to see the created environment variable.



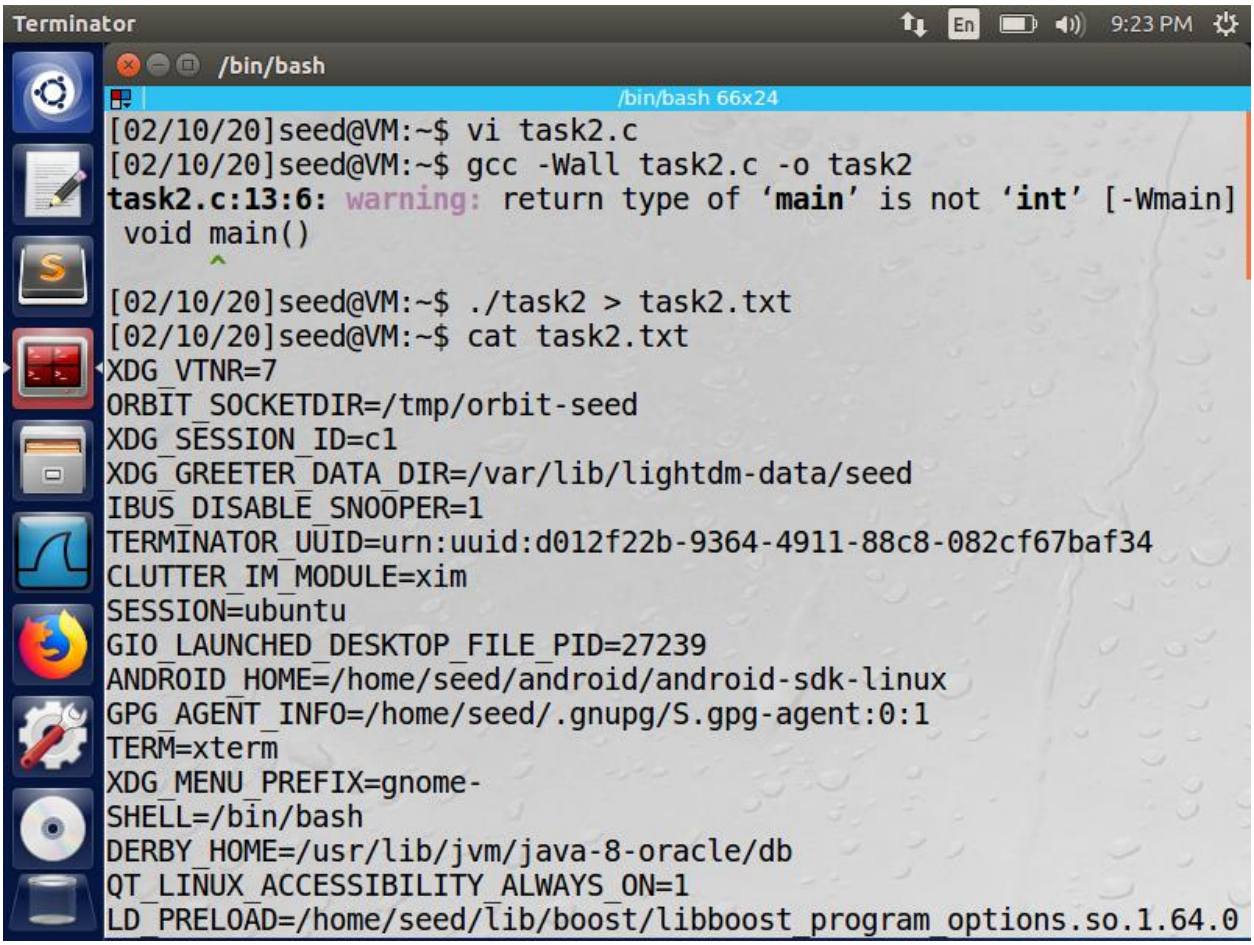


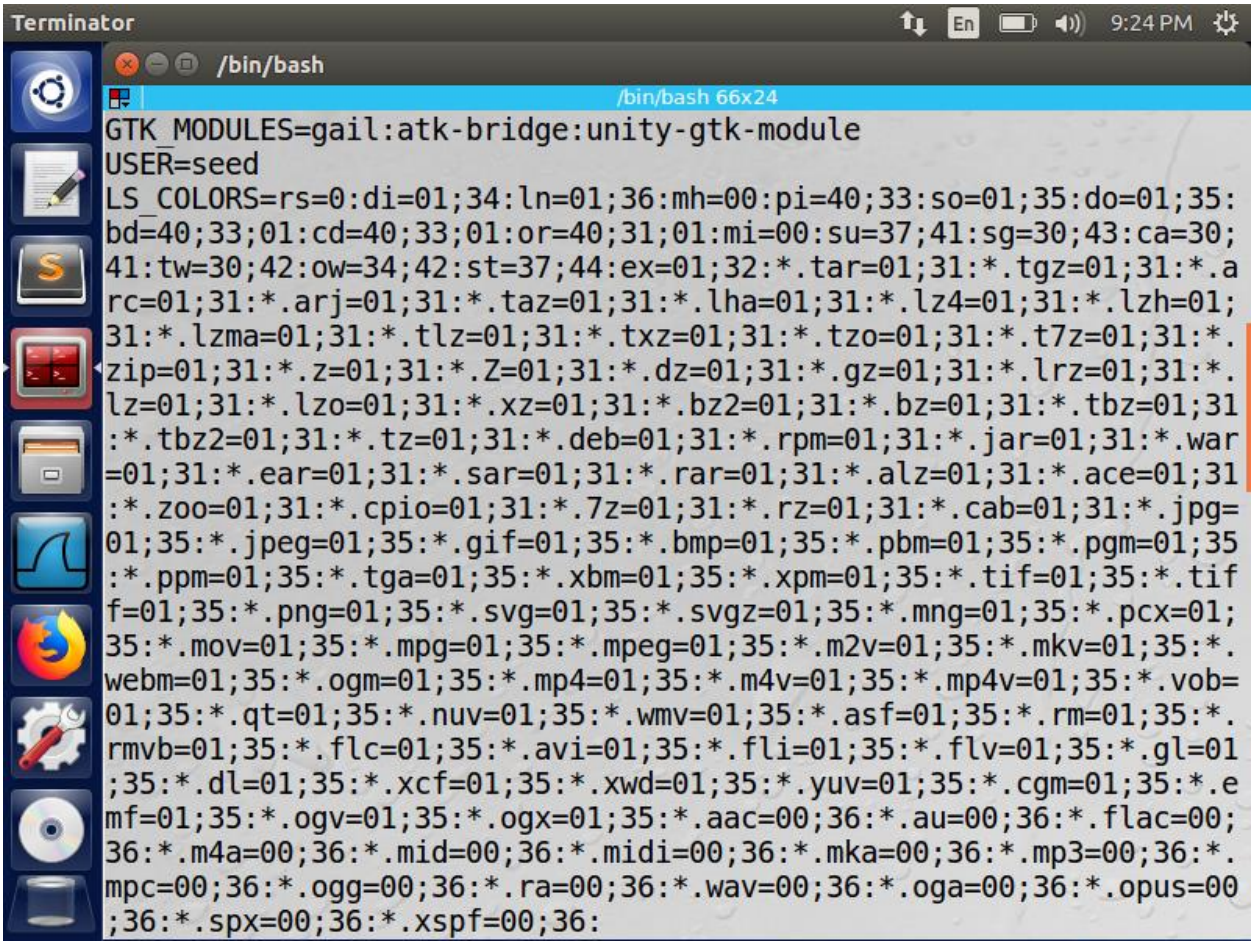
Task 2: Passing Environment Variables from Parent Process to Child Process

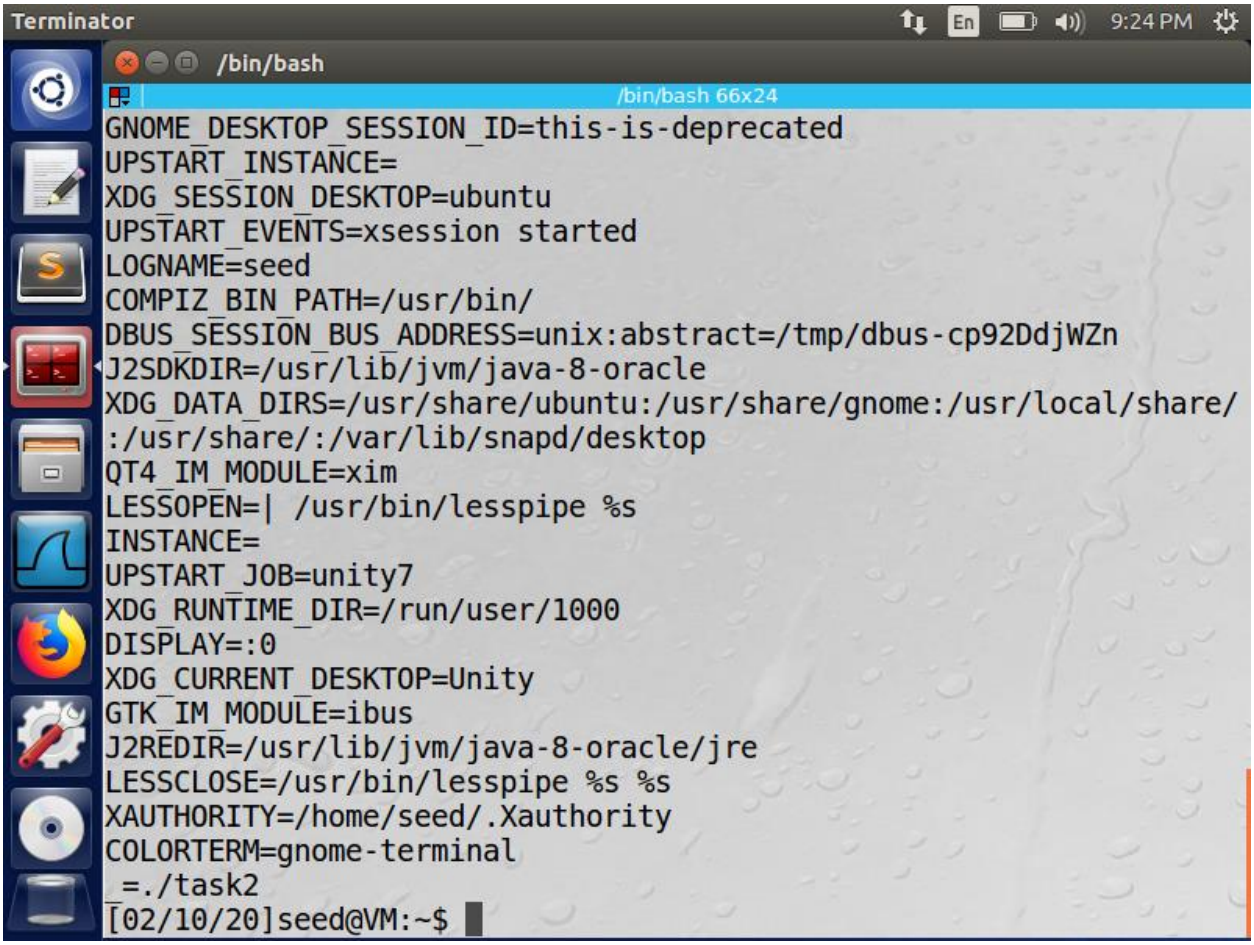
Step 1:

Output:

I run the task2 file which contains the program with uncommented printenv() method in the child process, and the printenv() method in the parent process is commented. Now, I compile and run the task2 file and save it in a text file task2.txt. I use cat command to view the contents of the text file task2.txt. I am able to see list of all environment variables being displayed.


Terminator
En
9:23 PM
/bin/bash
/bin/bash 66x24
[02/10/20]seed@VM:~$ vi task2.c
[02/10/20]seed@VM:~$ gcc -Wall task2.c -o task2
task2.c:13:6: warning: return type of 'main' is not 'int' [-Wmain]
 void main()
[02/10/20]seed@VM:~$ ./task2 > task2.txt
[02/10/20]seed@VM:~$ cat task2.txt
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:d012f22b-9364-4911-88c8-082cf67baf34
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=27239
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
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_options.so.1.64.0

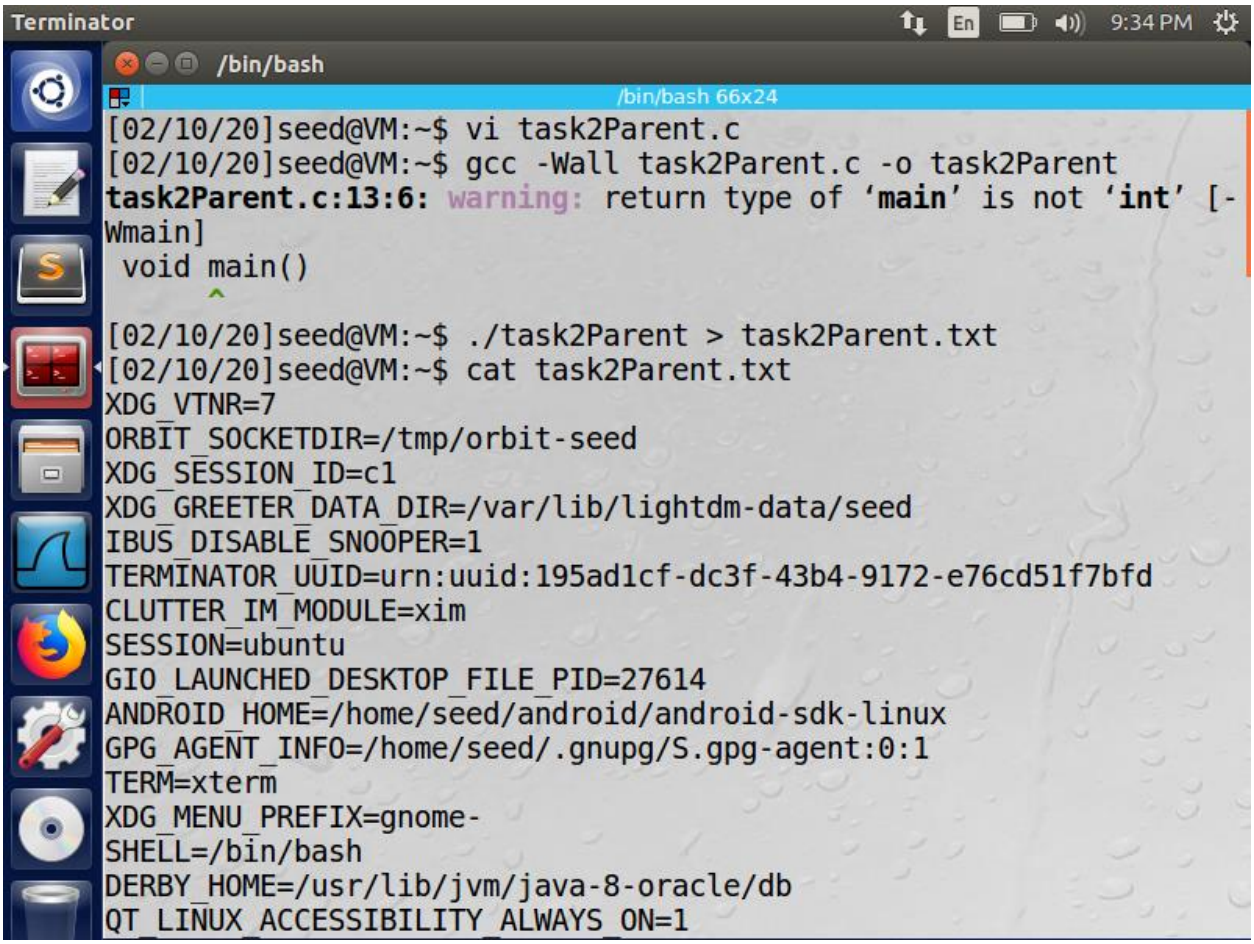

Terminator
En
9:24 PM
/bin/bash
/bin/bash 66x24
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:
bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;
41:tw=30;42:ow=34;42:st=37;44:ex=01;32:*.tar=01;31:*.tgz=01;31:*.a
rc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;
31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*.
zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.
lz=01;31:*.lzo=01;31:*.xz=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31
:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war
=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31
:*.zoo=01;31:*.cpio=01;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.jpg=
01;35:*.jpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pgm=01;35
:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tif
f=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;
35:*.mov=01;35:*.mpg=01;35:*.mpeg=01;35:*.m2v=01;35:*.mkv=01;35:*.
webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=
01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.
rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01
;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.e
mf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;
36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00
;36:*.spx=00;36:*.xspf=00;36:

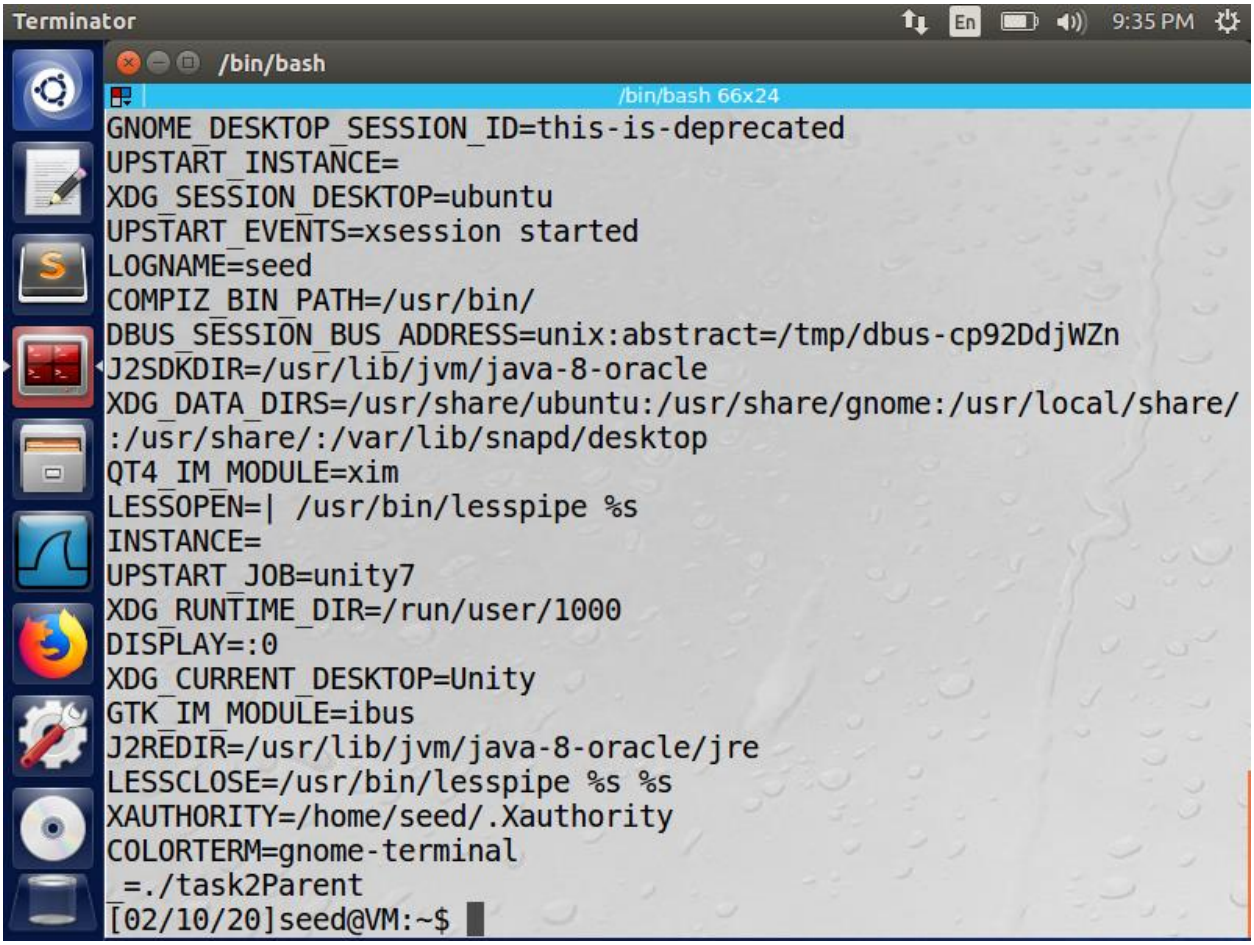


Step 2:

Output:

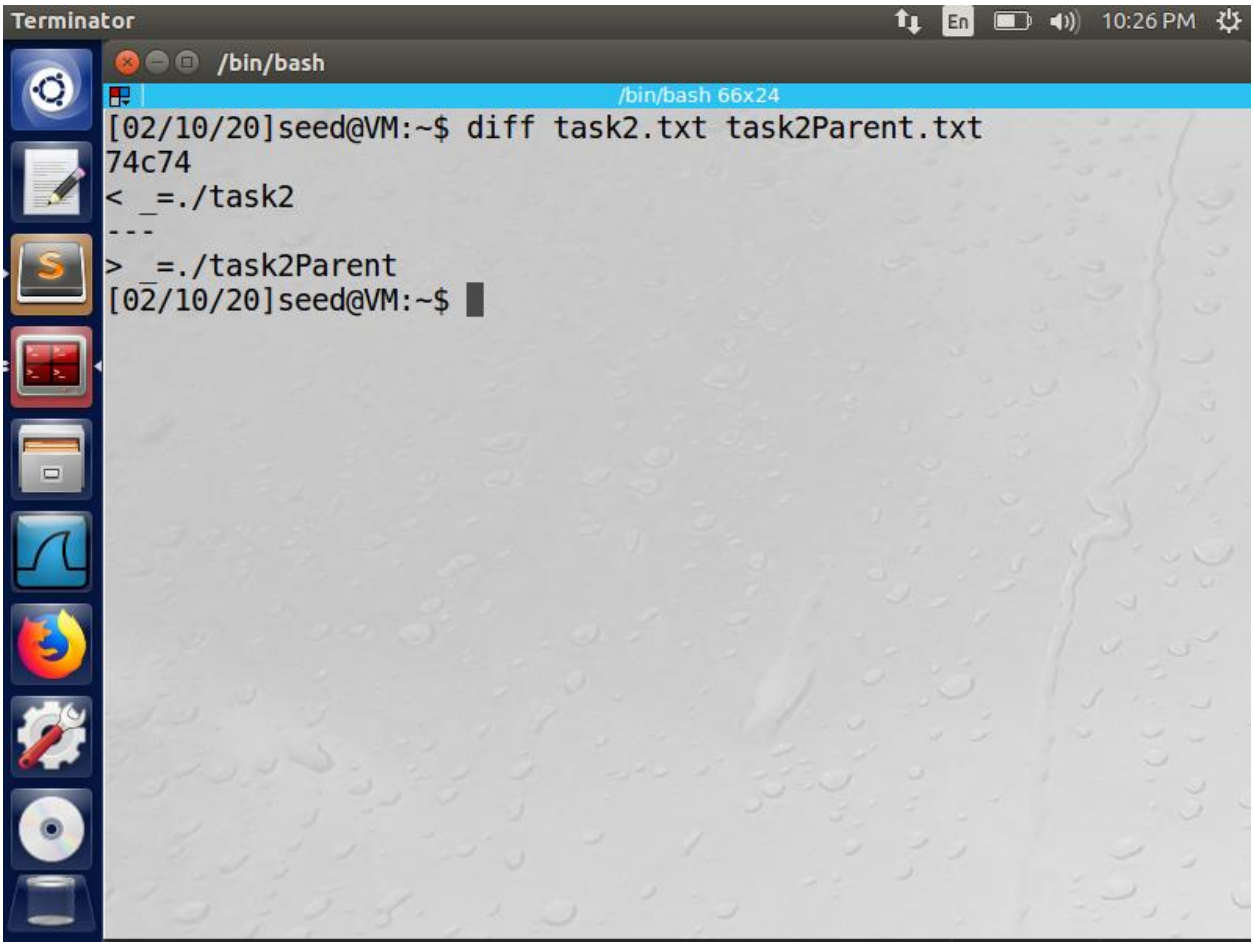
I run the task2Parent file which contains the program with uncommented printenv() method in the parent process, and the printenv() method in the child process is commented. Now, I compile and run the task2Parent file and saved it in a text file task2Parent.txt. I use cat command to view the contents of the text file task2Parent.txt. I am able to see list of all environment variables being displayed.


Terminator
En
9:34 PM
/bin/bash
/bin/bash 66x24
[02/10/20]seed@VM:~$ vi task2Parent.c
[02/10/20]seed@VM:~$ gcc -Wall task2Parent.c -o task2Parent
task2Parent.c:13:6: warning: return type of 'main' is not 'int' [-Wmain]
 void main()
[02/10/20]seed@VM:~$ ./task2Parent > task2Parent.txt
[02/10/20]seed@VM:~$ cat task2Parent.txt
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:195ad1cf-dc3f-43b4-9172-e76cd51f7bfd
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=27614
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1


Terminator
En
9:35 PM
/bin/bash
/bin/bash 66x24
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
UPSTART_INSTANCE=
XDG_SESSION_DESKTOP=ubuntu
UPSTART_EVENTS=xsession started
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-cp92DdjWZn
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/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
COLORTERM=gnome-terminal
_=./task2Parent
[02/10/20]seed@VM:~$

Step 3:

I used the diff command to compare the two text files task2.txt and task2Parent.txt. I am able to see that there is no any differences between the two text files except for the file name which results in 74c74. I conclude that the child process inherits the environment variable of the parent process.

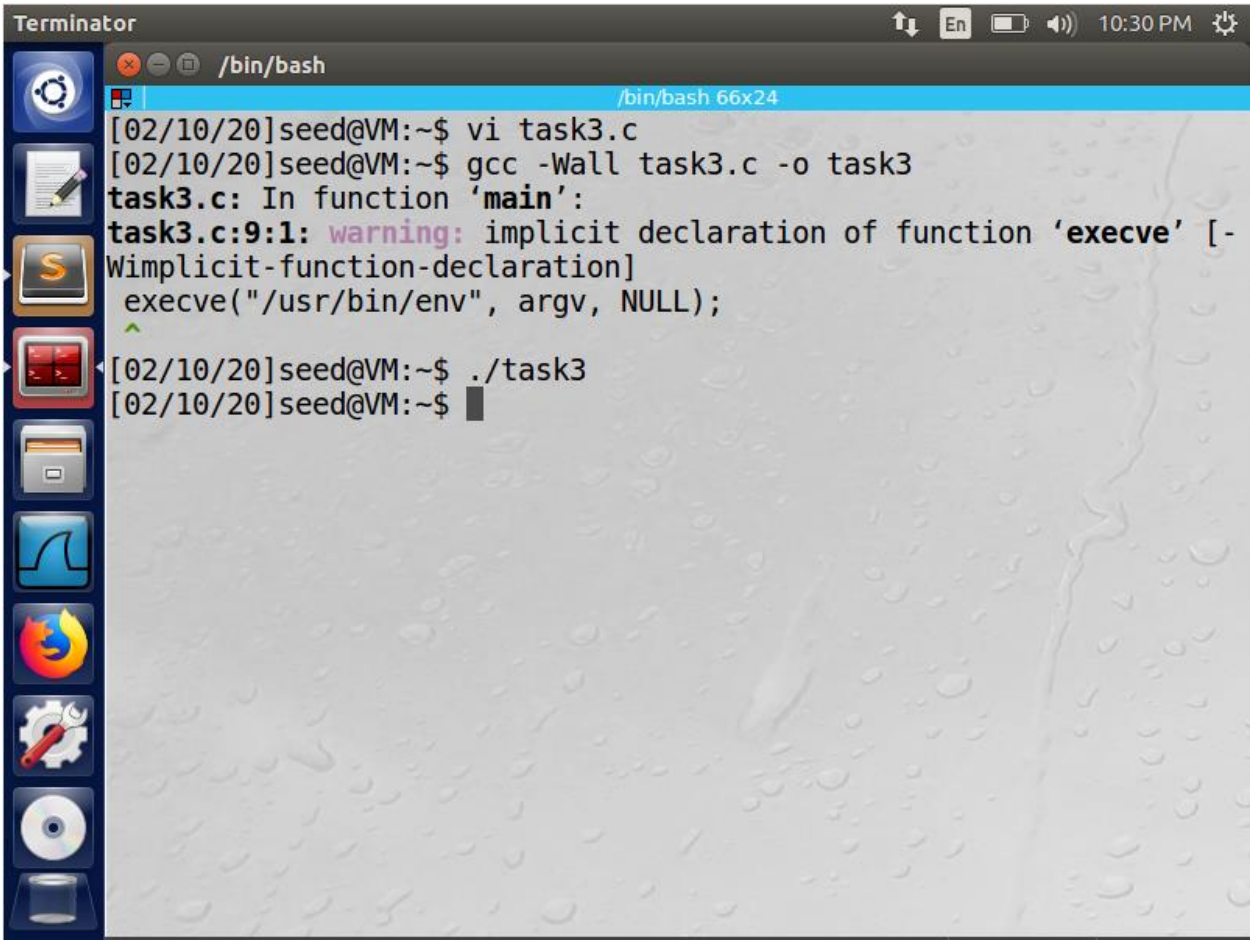


Task 3: Environment Variables and execve()

Step 1:

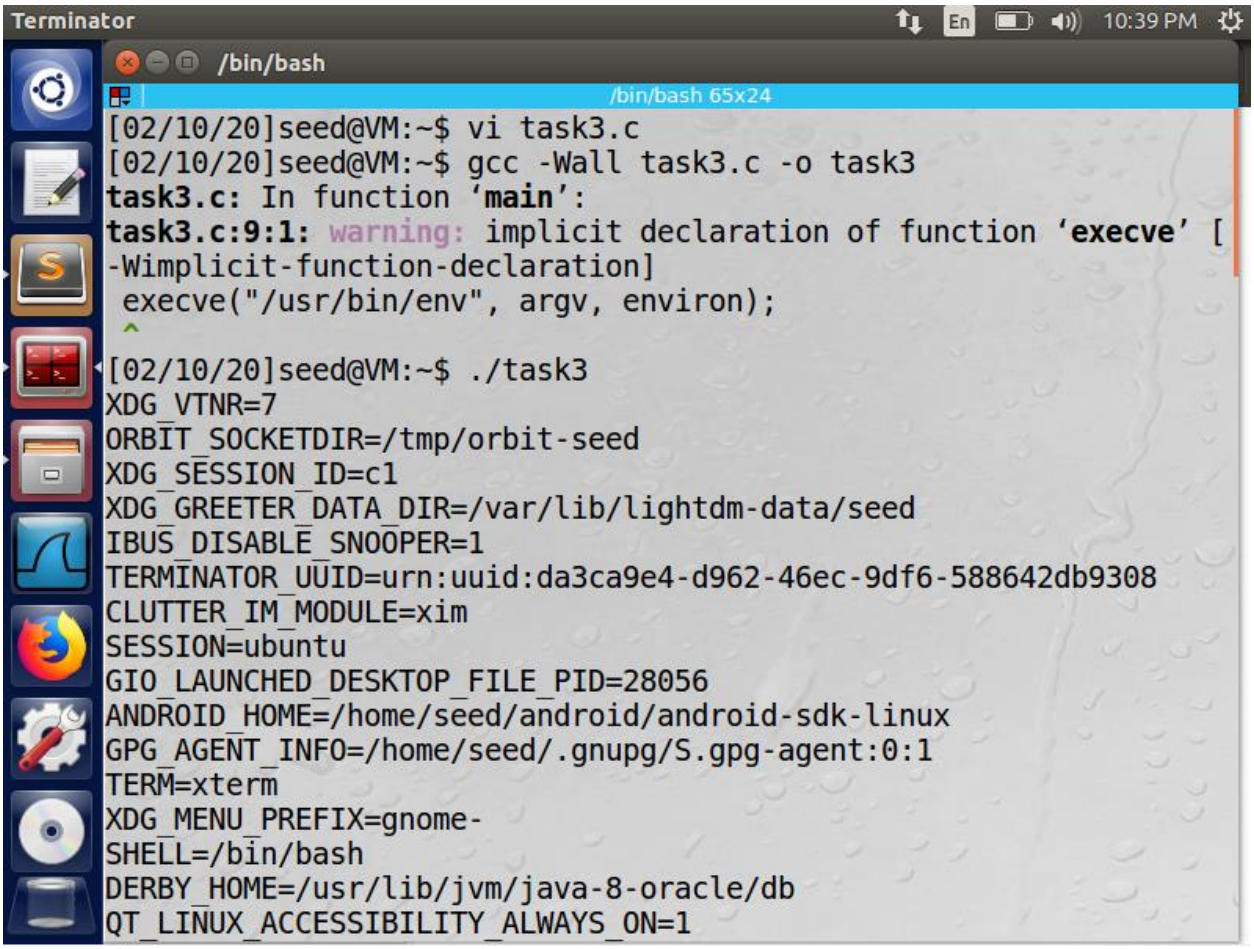
Output:

When I compile and run the given program, I am able to see that nothing gets executed. This is because NULL is passed as the third parameter to the execve() method, which results in empty output. This is because the environment variable is not stored in the memory.



Step 2:

When I pass the third parameter from NULL to environ, after compiling and running the program I am able to see the list of environment variables displayed. This is because the environment variables are loaded into the memory.



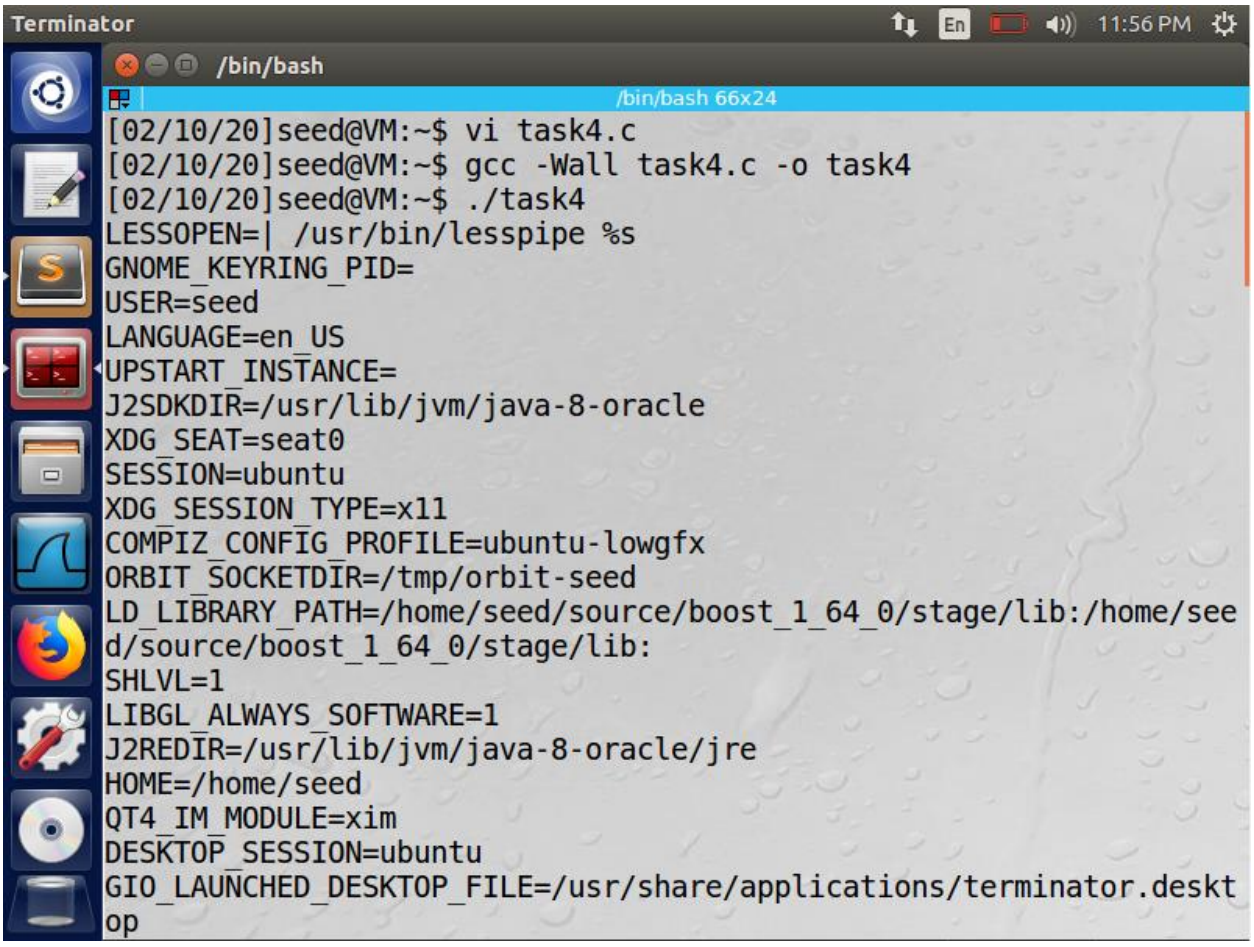
Step 3:

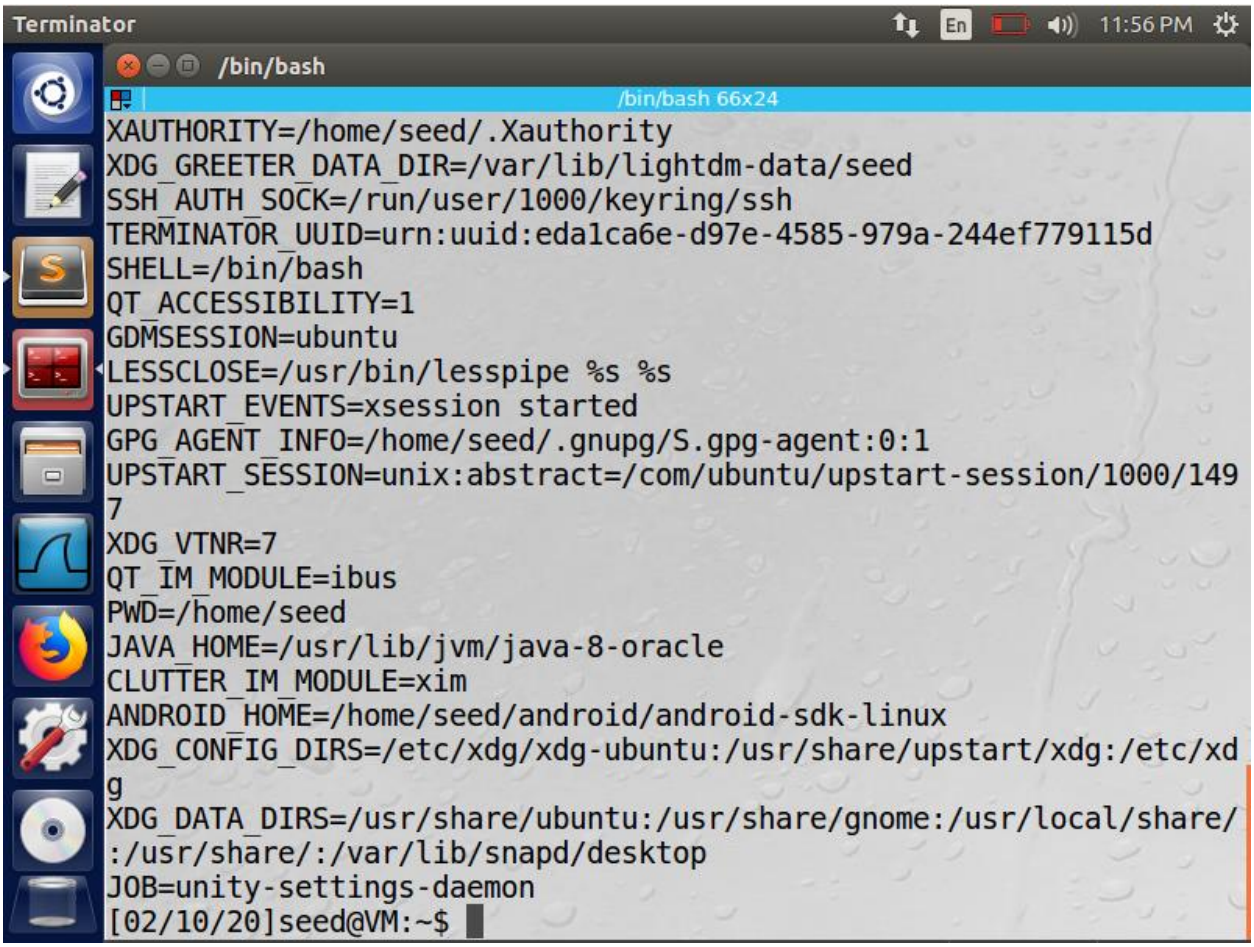
The reason that the new program is able to print the environment variables is because the environ variable is passed as the third parameter to the execve() method which prints the list of environment variables.

Task 4: Environment Variables and system()

Output:

I can see that environmental variable is passed when executed through system("/usr/bin/env"). Basically, system() does not execute the command directly and when I use this system("/usr/bin/env"), environmental variables are passed that are getting displayed and hence verified.

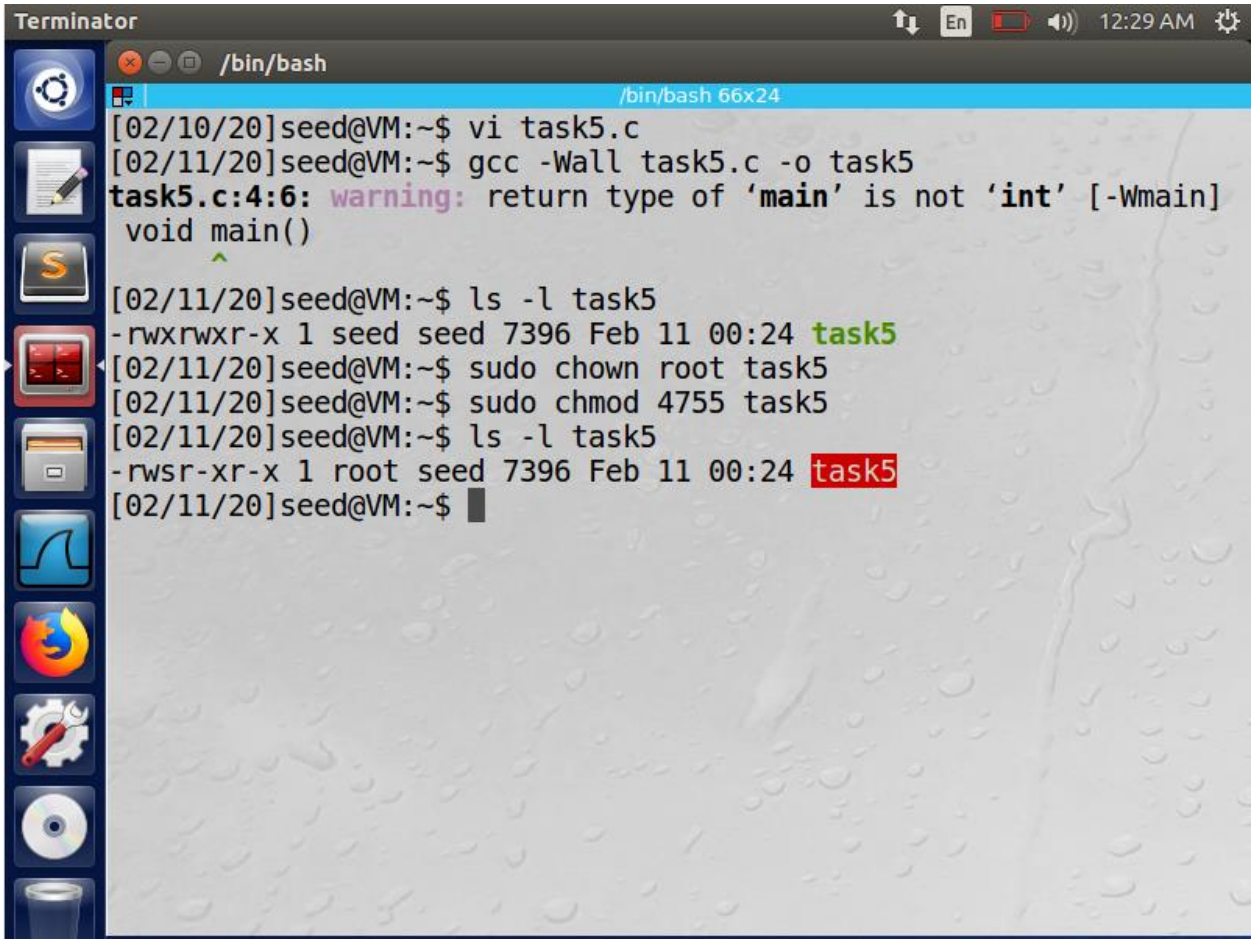



Terminator
En
11:56 PM
/bin/bash
/bin/bash 66x24
XAUTHORITY=/home/seed/.Xauthority
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
TERMINATOR_UUID=urn:uuid:eda1ca6e-d97e-4585-979a-244ef779115d
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
UPSTART_EVENTS=xsession started
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/149
7
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed
JAVA_HOME=/usr/lib/jvm/java-8-oracle
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xd
g
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/
:/usr/share/:/var/lib/snapd/desktop
JOB=unity-settings-daemon
[02/10/20]seed@VM:~$

Task 5: Environment Variable and Set-UID Programs

Step 1 and Step 2:

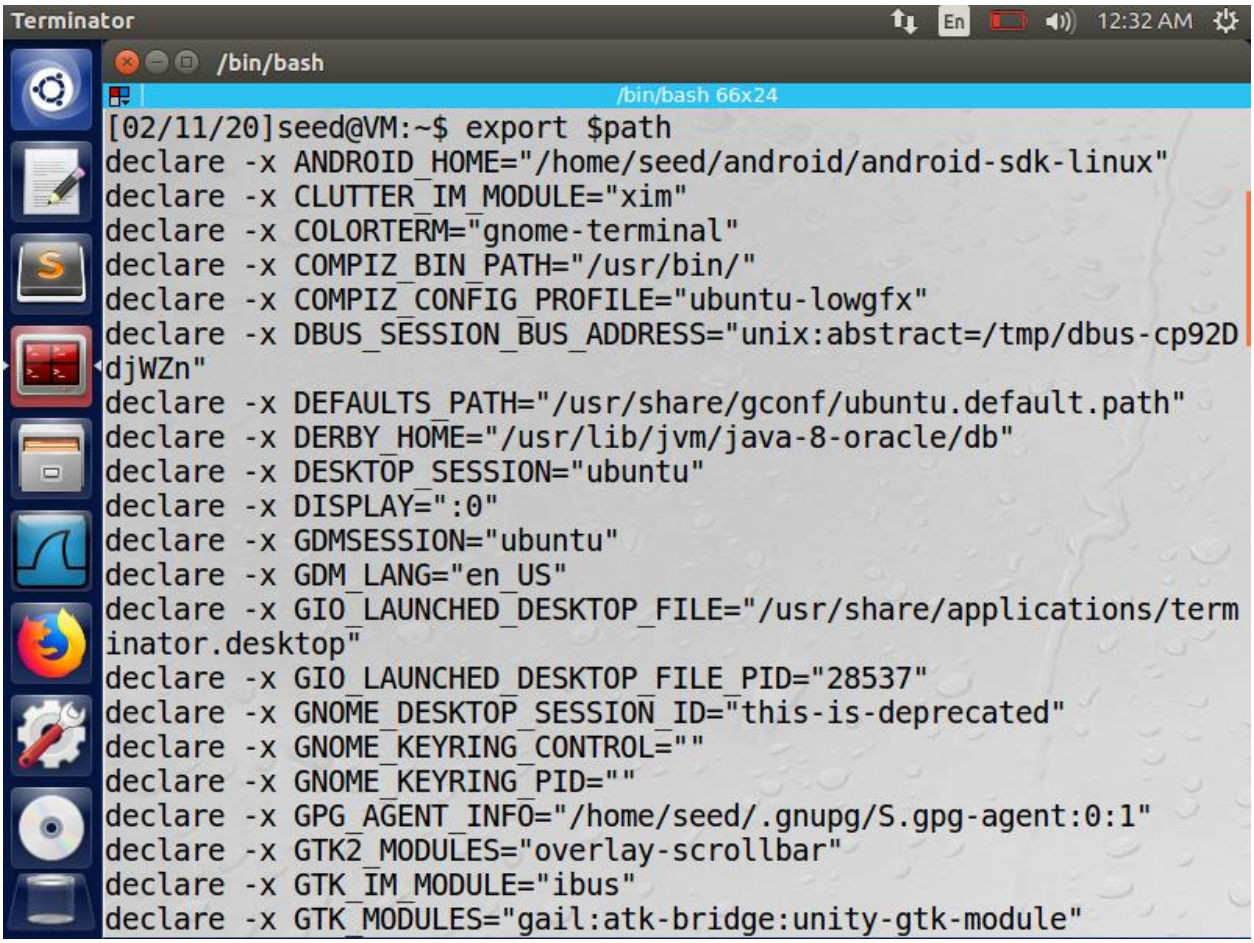
Compiling the given program and changing its owner to root and making it a SET-UID program.



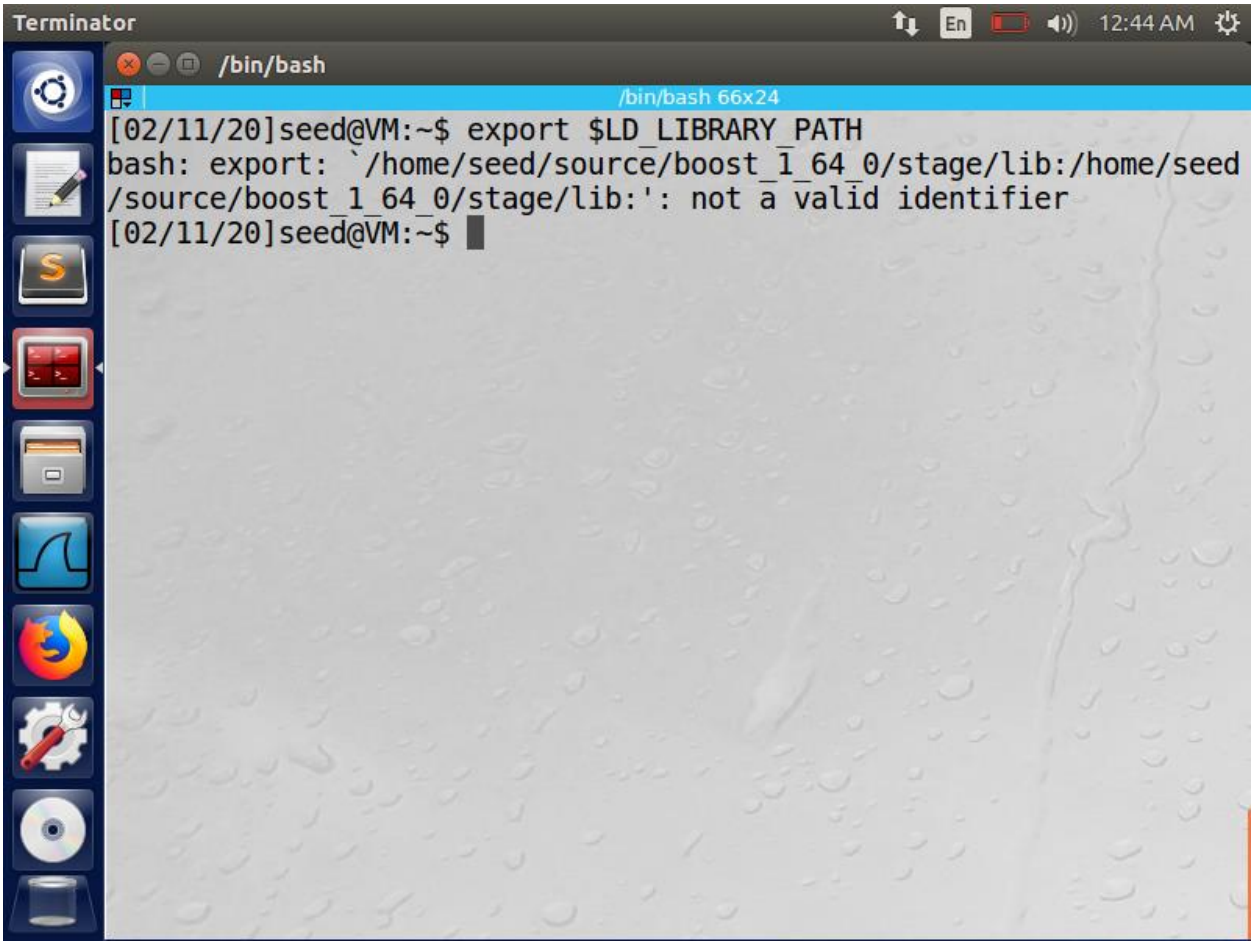
Step 3:

Output:

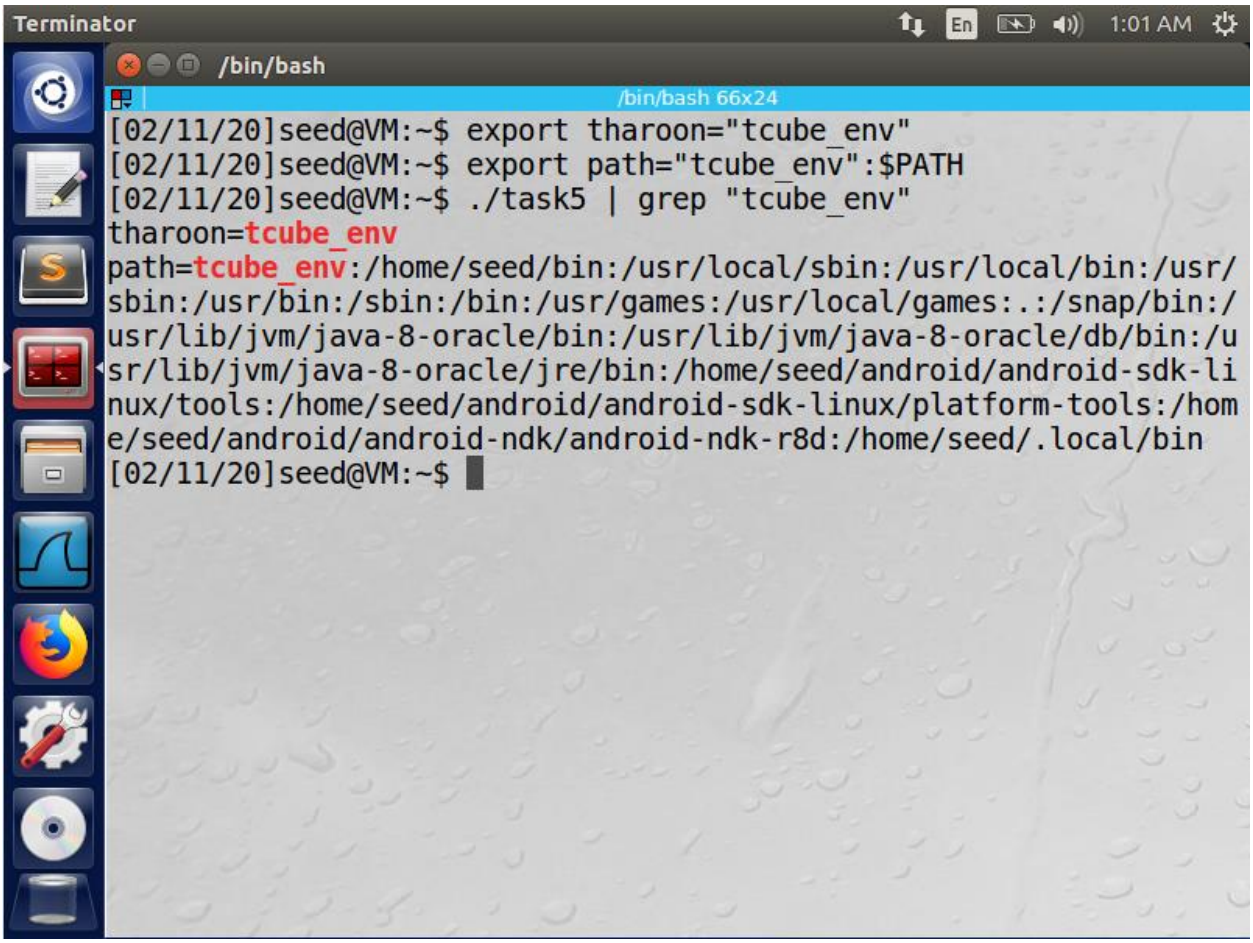
Using the export command to set PATH:



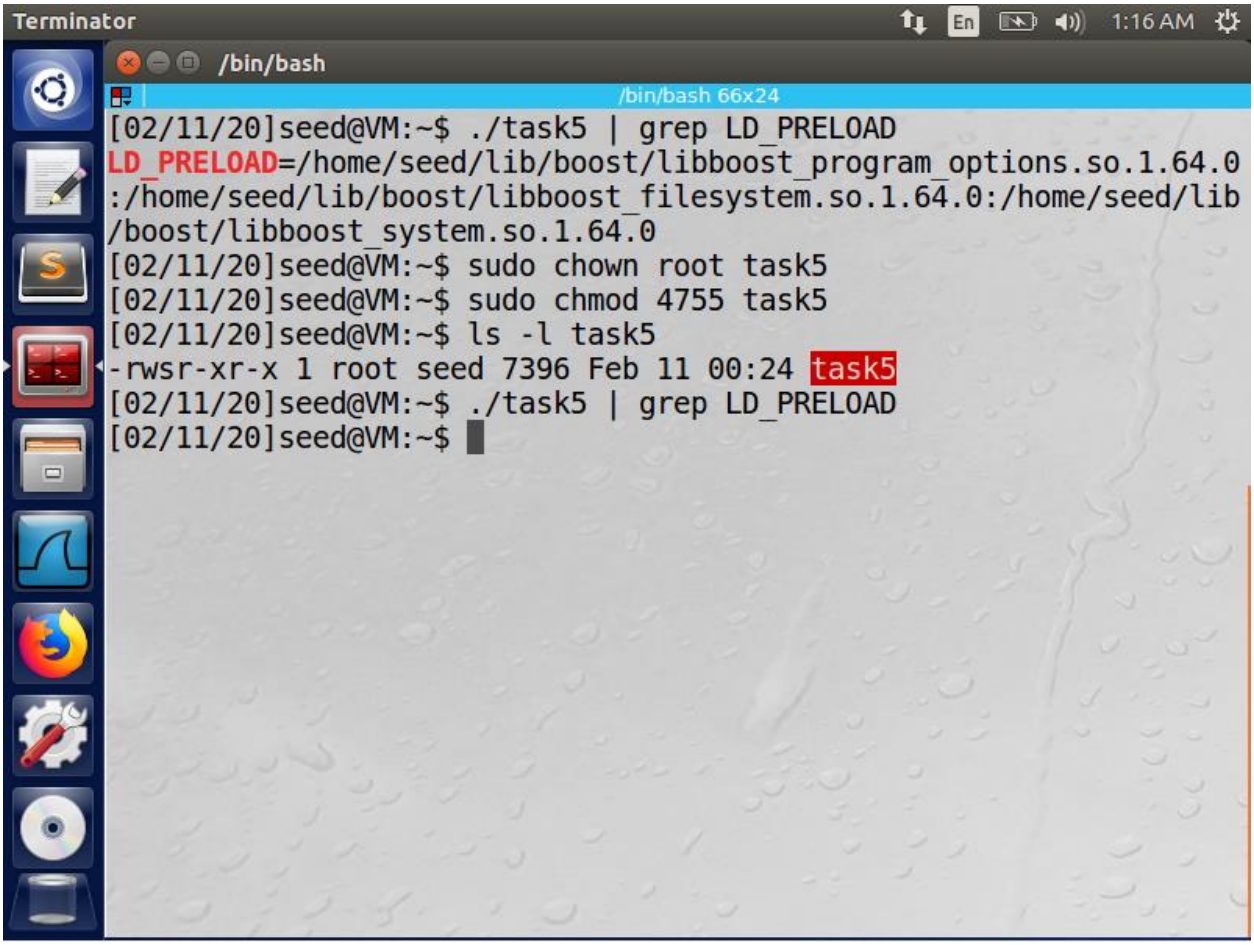
Using the export command to set LD_LIBRARY_PATH:



I now create a new environment variable and set it into the new path using the export command. Export command will allow a child variable to inherit all the variables from the parent. Now, I create a new environment variable and append it to the PATH.

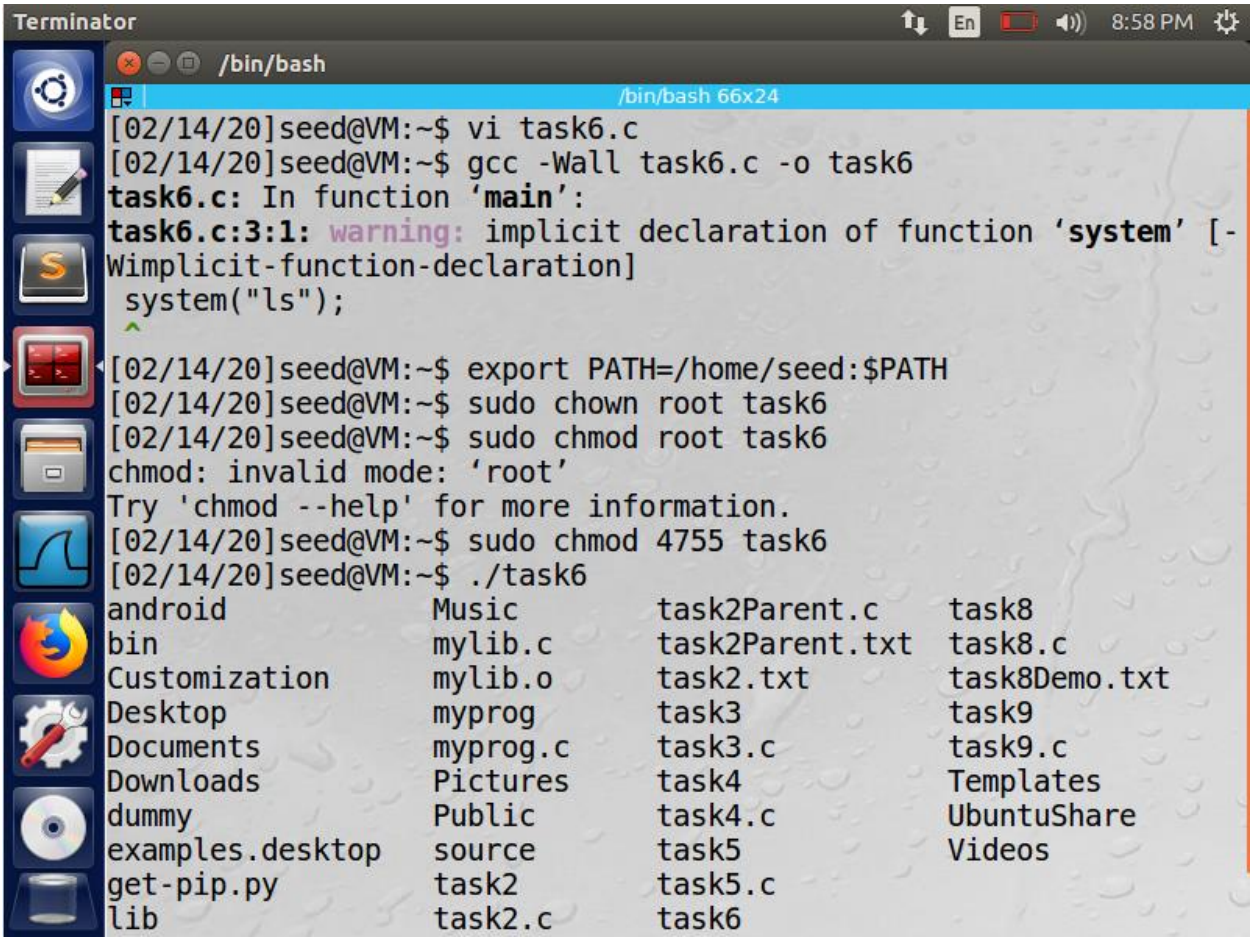


The Surprise thing I noticed is that when I run the task5.file and grep the LD_PRELOAD I am able to see the environment variable. If I change the task5.c file to root and change it as the SET-UID program and grep the LD_PRELOAD I am not able to find the environment variable. This shows that LD_* is not inheritable when run as root.

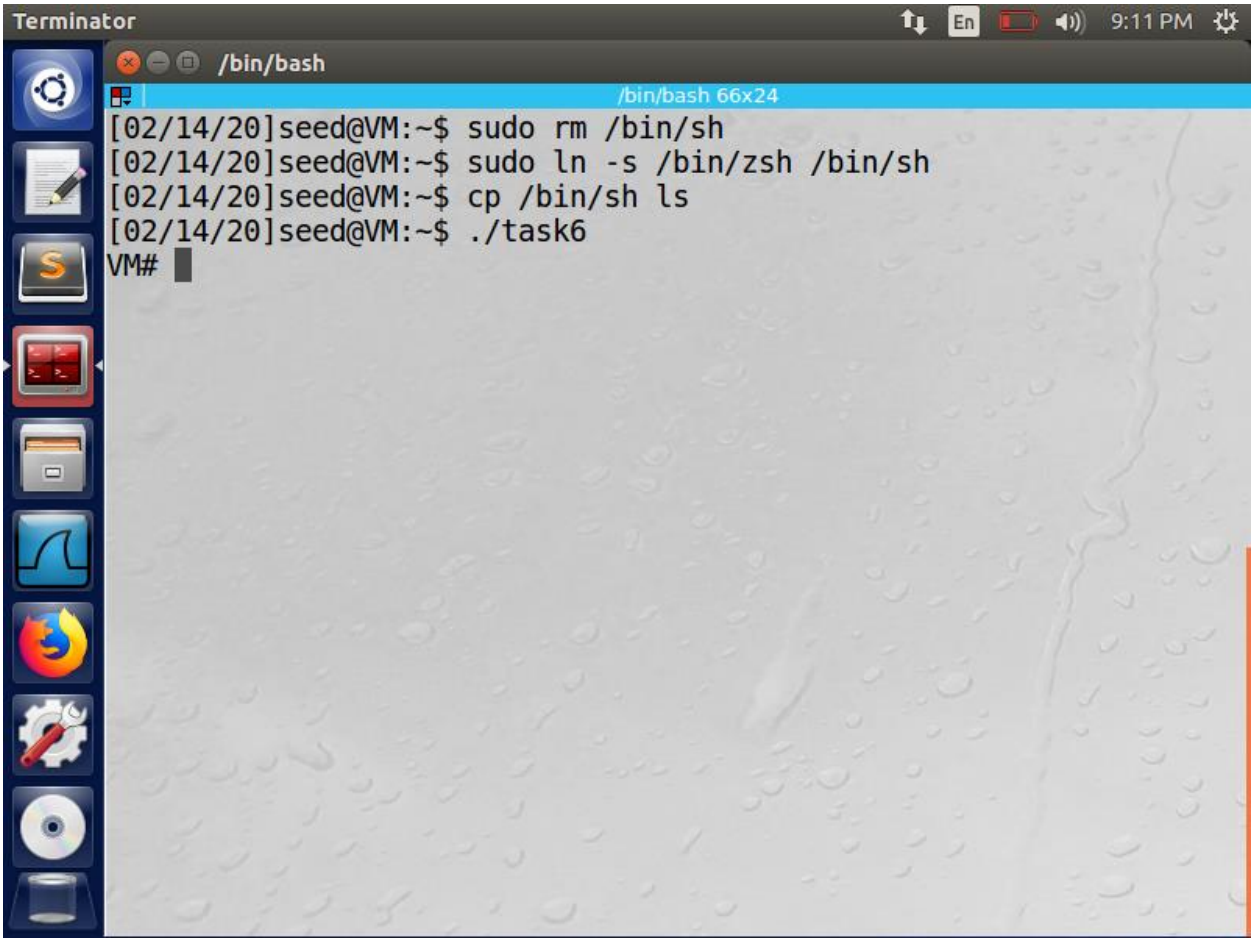


Task 6: The PATH Environment Variable and Set-UID Programs

I compiled the given program and made it as root owned SET-UID program. Then I export the given environment variable export PATH=/home/seed:$PATH. Now when I run the program, I am able to run the ls command and I am able to see the list of all files and file directory. This is because of the countermeasure by the /bin/sh. Since the EUID and the RUID is different.



Now I linked the /bin/sh to another shell that does not have any countermeasure, using the given commands, I point to the shell with no countermeasure. And I have copied the /bin/sh to ls. Now when I run the given program, I am able to gain access to the root. This is because the shell which I changed has no countermeasure and this makes the program to gain the access to root.

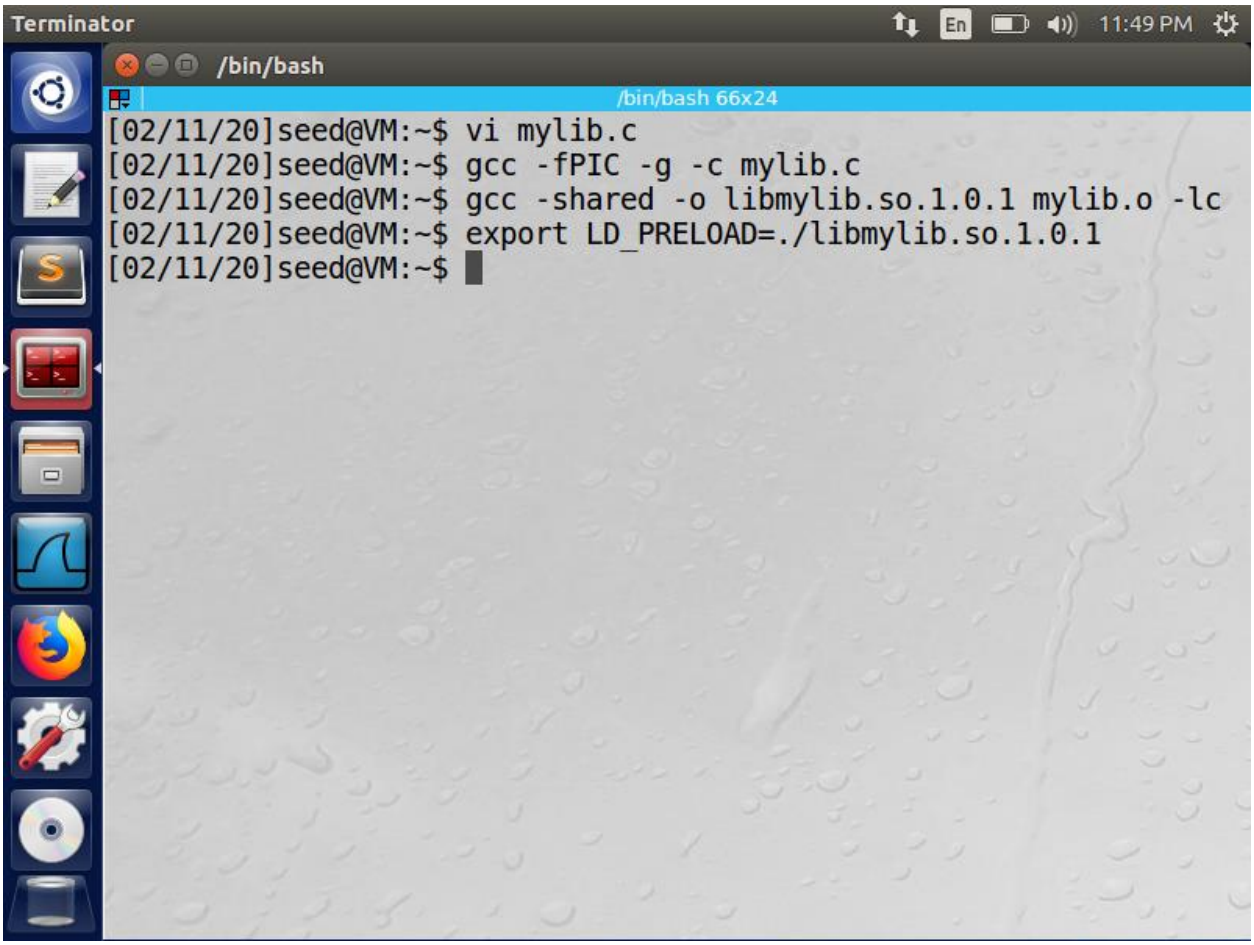


Task 7: The LD_PRELOAD Environment Variable and Set-UID Programs

Step 1:

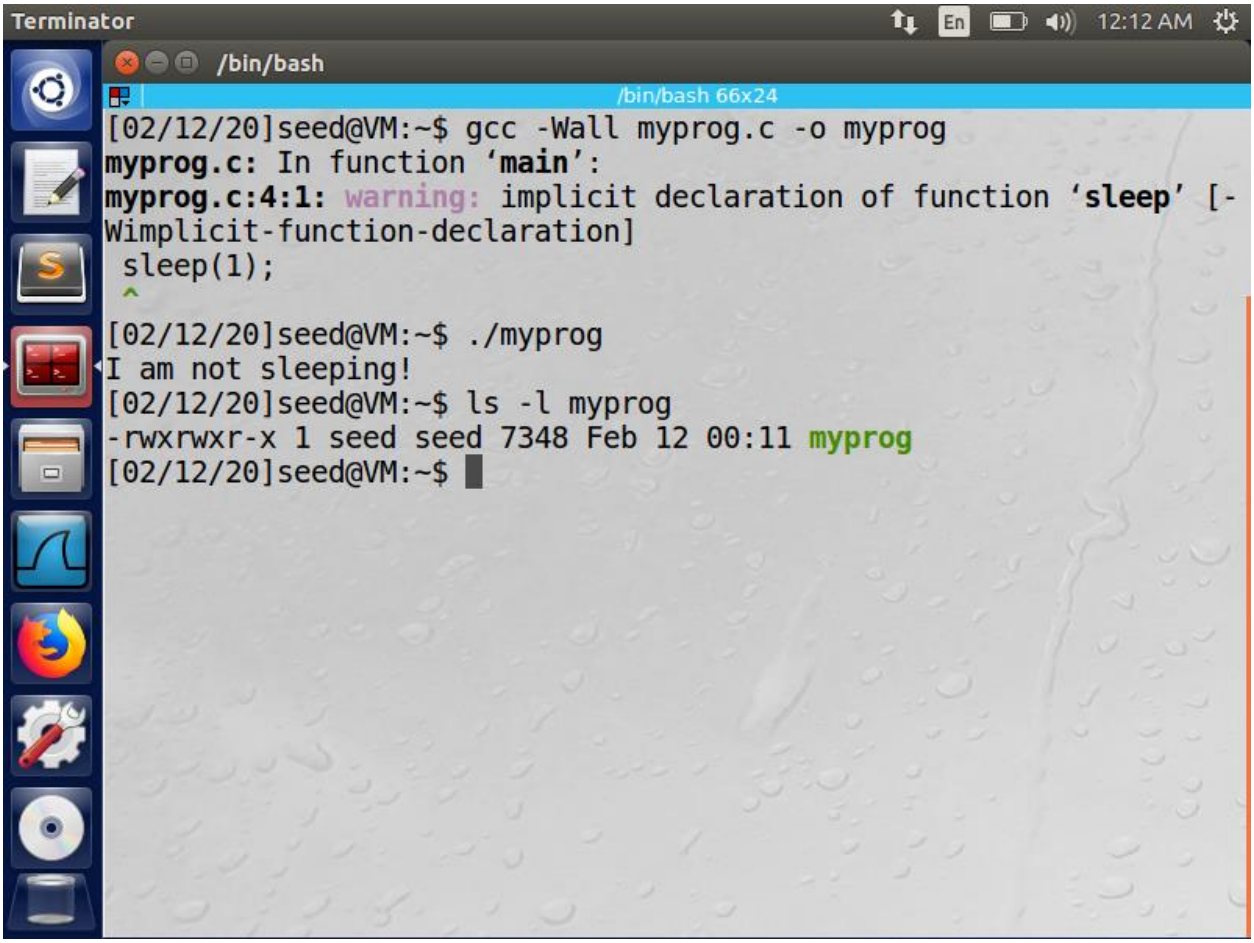
Output:

Compiled and executed as given. Then created the shared path and exported the LD_PRELOAD path.

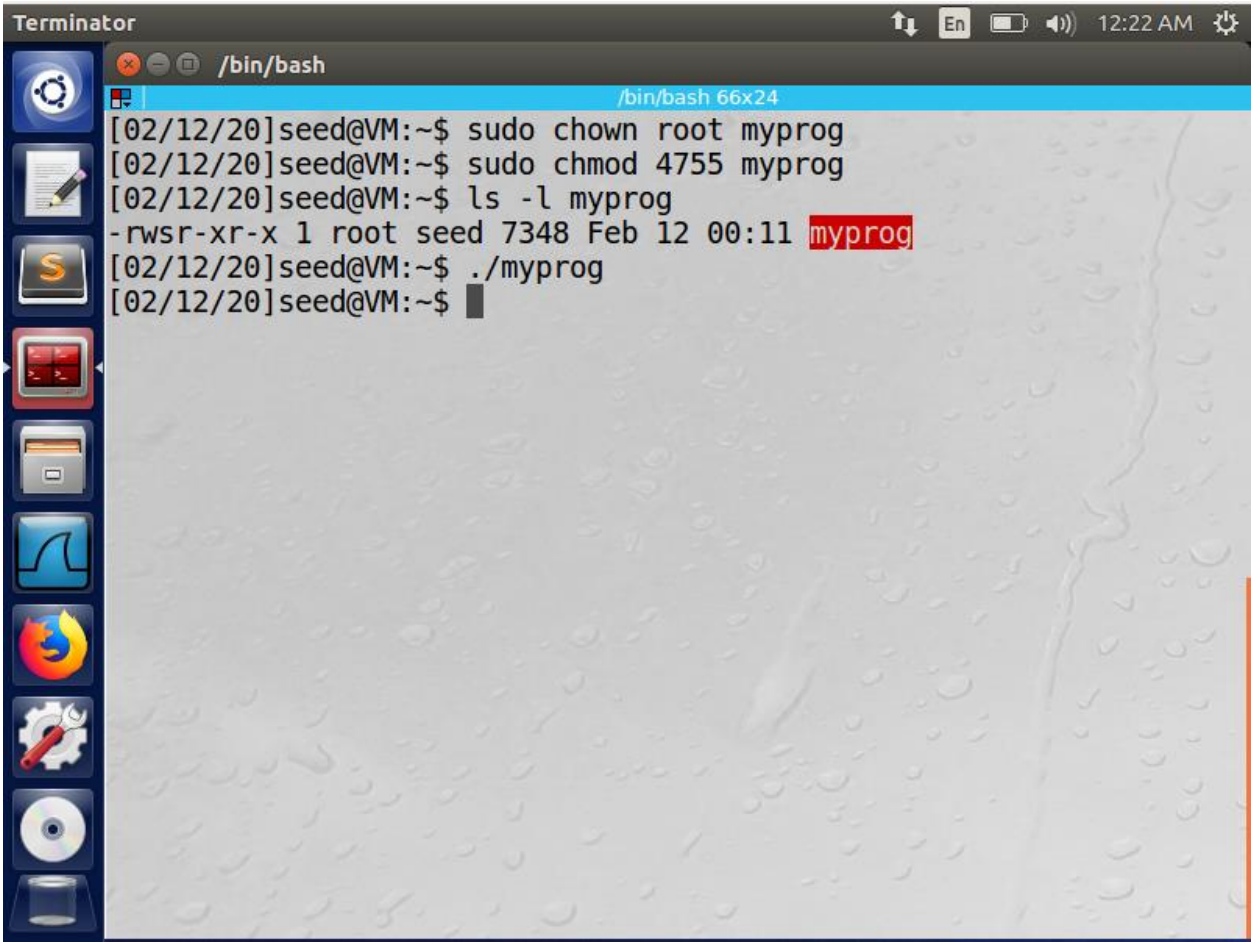


Step 2:

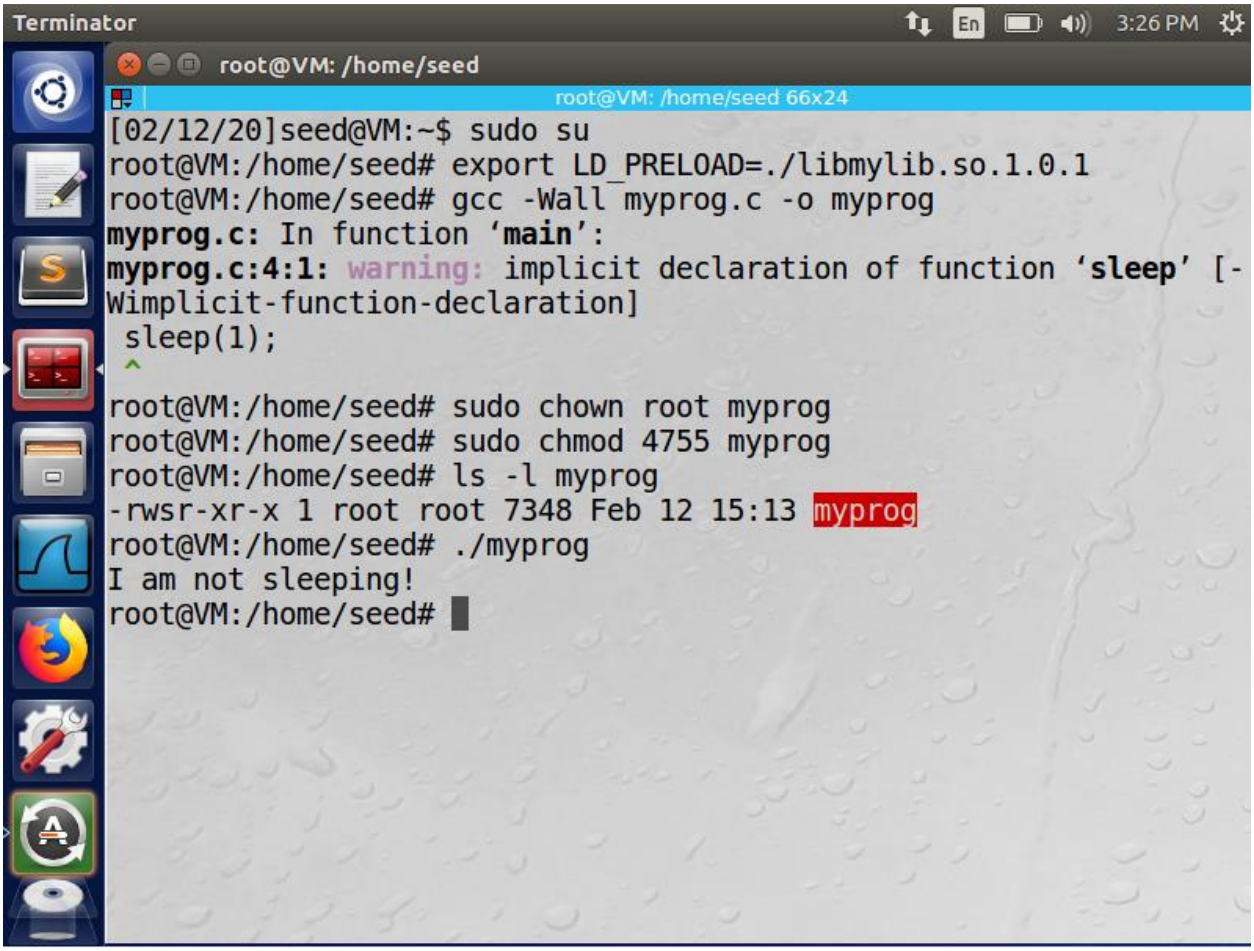
1. Now I made myprog as a regular program and ran it as normal user. I am able to see the output 'I am not sleeping!'. This is due to running the program in the normal user, which has EUID and RUID as same. Here the bash does not perform the counter measure.



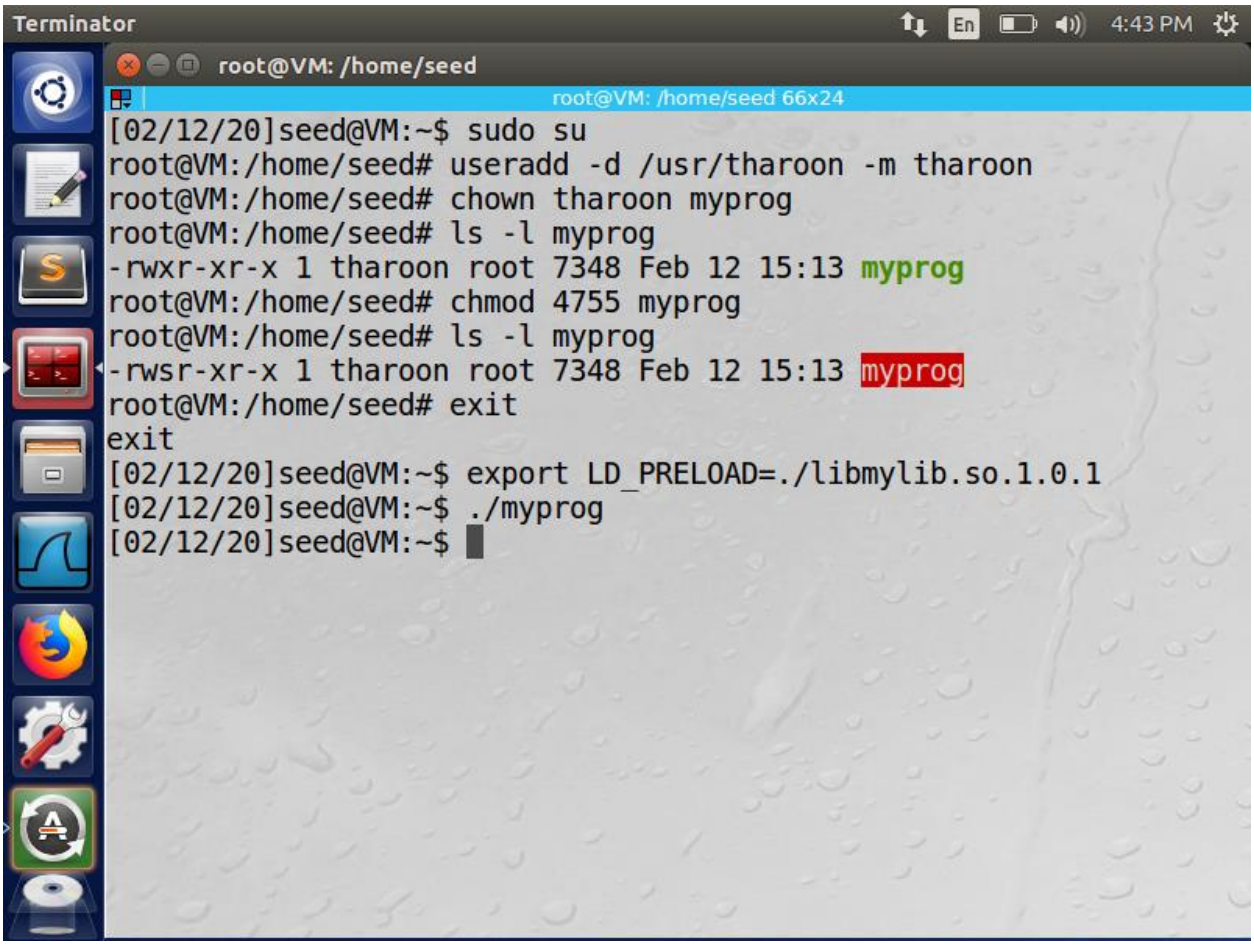
2. I made myprog to a SET-UID root program and I ran it as normal user. I was able to see that when I ran myprog as normal user, the sleep function of myprog is called. This is because of the SET-UID program. The bash takes the countermeasure and sees that EUID and RUID is different and does not execute the mylib program.



3. I now get into the root account and export the LD_ PRELOAD environment variable.Then I compile the myprog.c and change it to SET-UID program and ran the program, I was able to see the output getting printed. This is because both EUID and RUID is same which is the root user.



4. I created a new user called tharoon and made myprog owner as tharoon and made myprog a SET-UID program and I exit from the root. Now I am in the seed user and I export the LD_PRELOAD from the seed user. Now when I run the myprog I am not able to see the output. This is because myprog is now SET-UID program with its ownership as tharoon user.

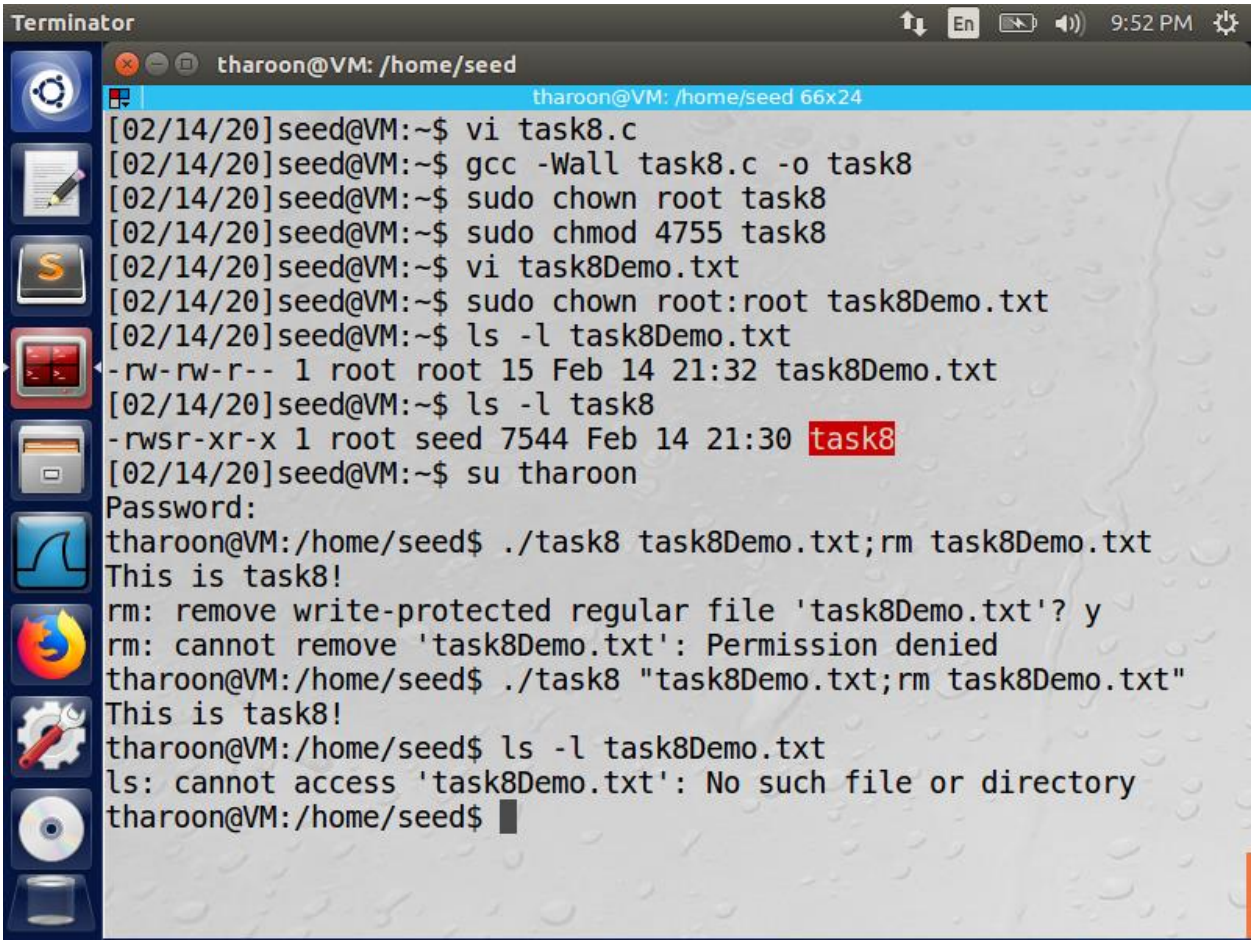

Terminator
En
4:43 PM
root@VM: /home/seed
root@VM: /home/seed 66x24
[02/12/20]seed@VM:~$ sudo su
root@VM:/home/seed# useradd -d /usr/tharoon -m tharoon
root@VM:/home/seed# chown tharoon myprog
root@VM:/home/seed# ls -l myprog
-rwxr-xr-x 1 tharoon root 7348 Feb 12 15:13 myprog
root@VM:/home/seed# chmod 4755 myprog
root@VM:/home/seed# ls -l myprog
-rwsr-xr-x 1 tharoon root 7348 Feb 12 15:13 myprog
root@VM:/home/seed# exit
exit
[02/12/20]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
[02/12/20]seed@VM:~$ ./myprog
[02/12/20]seed@VM:~$

Task 8: Invoking External Programs Using system() versus execve()

Step 1:

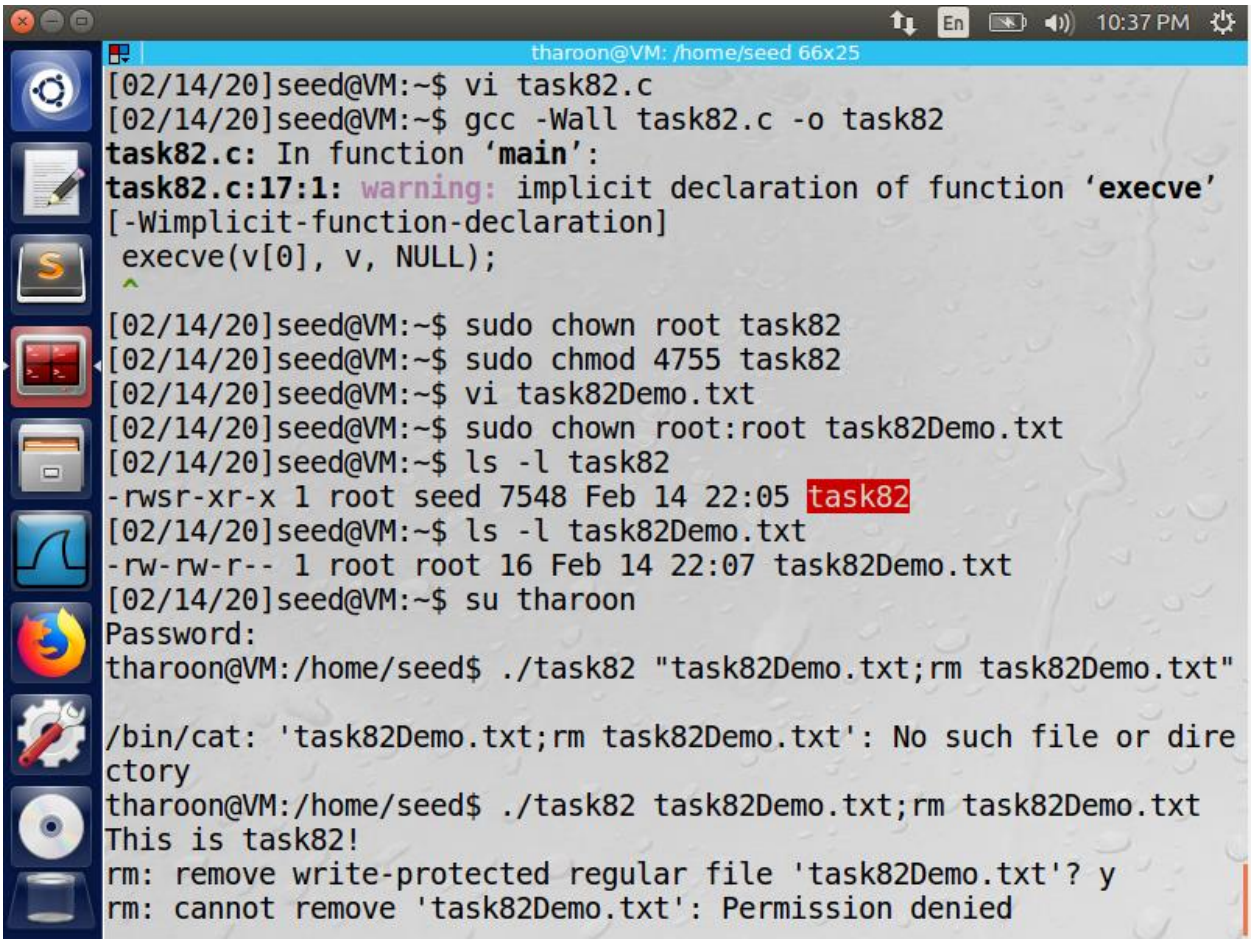
Output:

I compiled the given program and made it a SET-UID program and changed the owner to root. I also created a file called task8Demo.txt. Now, I enter into the user tharoon and run the given program and try removing the created file task8Demo.txt by giving the commands in double quotes. I am able to delete the file. This is because when system() executes, it doesn't execute the command directly. Instead it calls the execl() function, this execl() function eventually calls execve() to run /bin/sh. So, if the program is a Set-UID program, the user will have temporary root privileges and can remove any file with root privileges. And I also show that when removing the file without giving it as an argument I am not able to delete the file.



Step2:

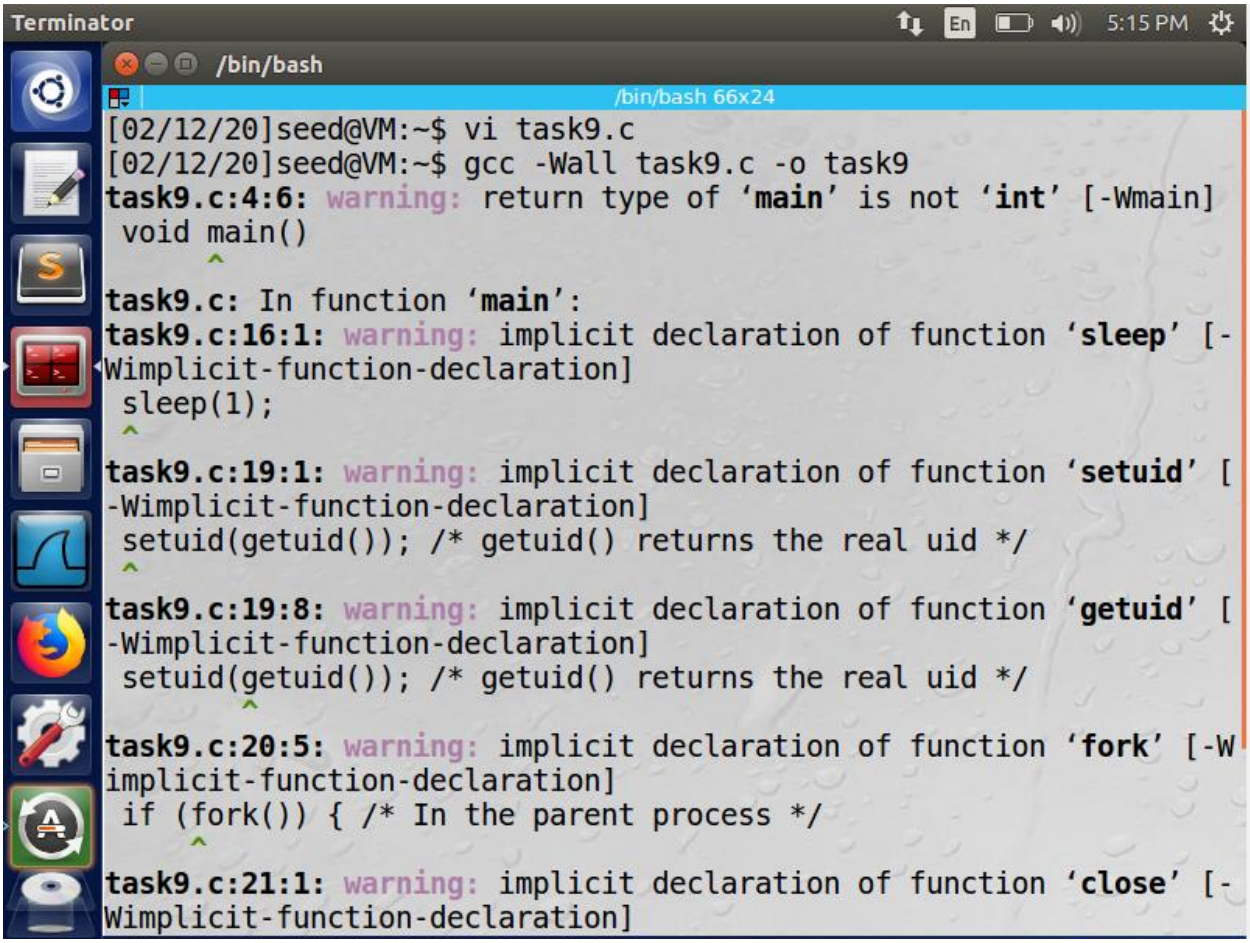
Now I commented out the system() function and uncommented the execve() function in the given program. I also created a file called task82Demo.txt. Now, I enter into the user tharoon and run the given program and try removing the created file task82Demo.txt by giving the commands in double quotes. I am not able to remove the file. This is because of the execve() command, which directly runs the commands.

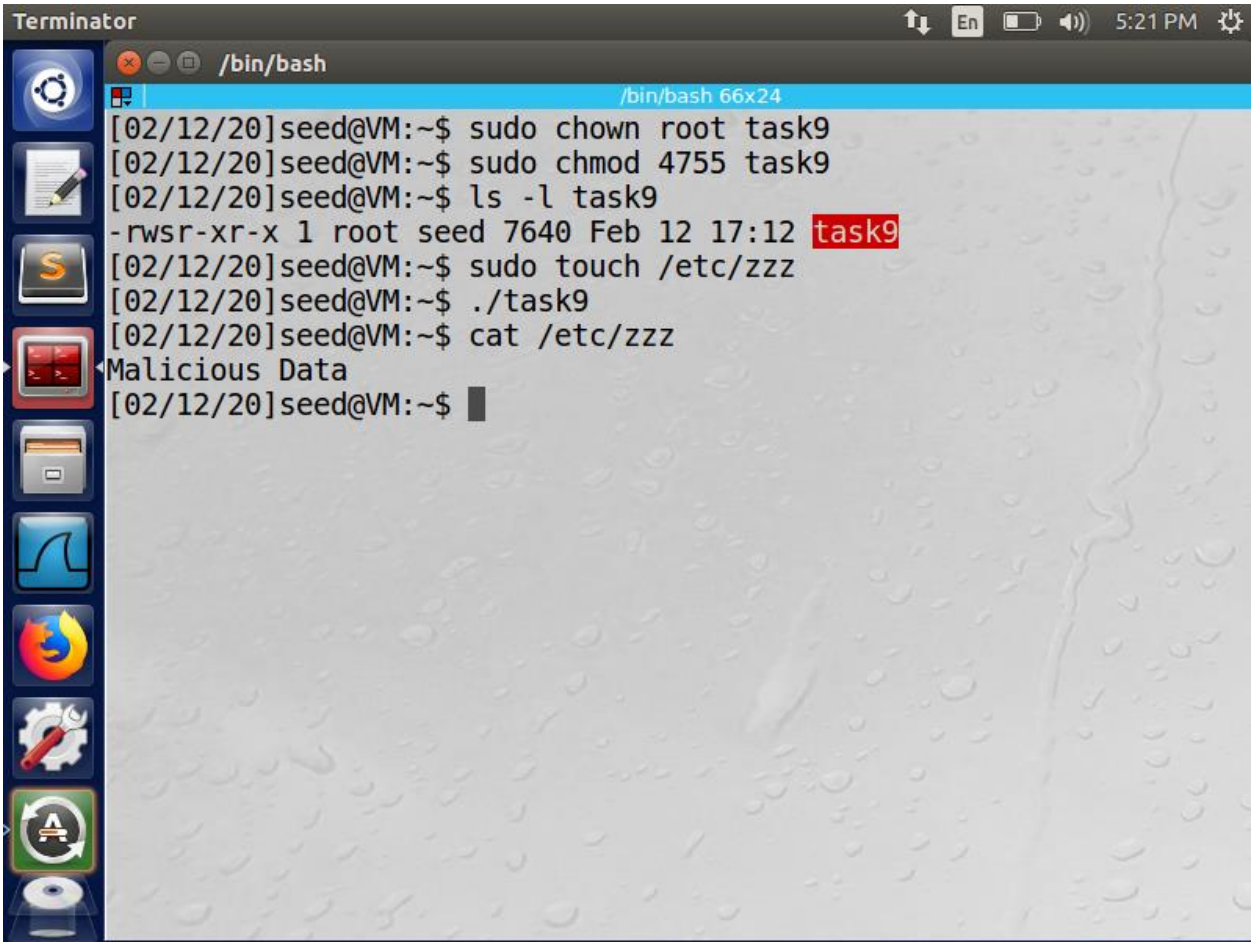


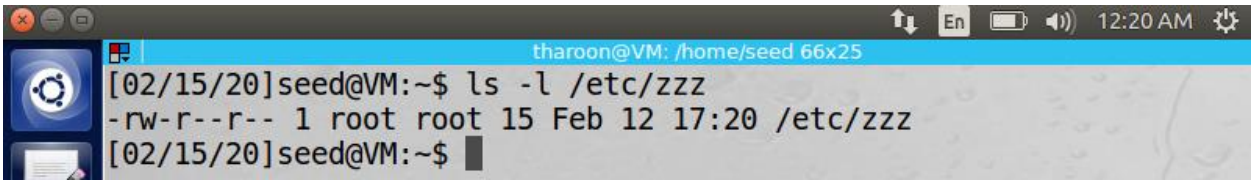
Task 9: Capability Leaking

Observation:

I have compiled the given program and made the compiled a SET-UID program and changed its owner to root. I have also created a file /etc/zzz using the sudo touch command which makes the owner as root. Now when I run the program, I am able to see the output getting printed. This is because the child process refers to the file descriptor of the parent process. So, the privileges of the parent did not downgrade. Downgrading of privileges happens using the setuid(getuid()) in the program. Since the privileges were not downgraded, the child process is able to access the created file. Also, the reason is file descriptor is not closed before the fork process.




Terminator
En
5:21 PM
/bin/bash
/bin/bash 66x24
[02/12/20]seed@VM:~$ sudo chown root task9
[02/12/20]seed@VM:~$ sudo chmod 4755 task9
[02/12/20]seed@VM:~$ ls -l task9
-rwsr-xr-x 1 root seed 7640 Feb 12 17:12 task9
[02/12/20]seed@VM:~$ sudo touch /etc/zzz
[02/12/20]seed@VM:~$ ./task9
[02/12/20]seed@VM:~$ cat /etc/zzz
Malicious Data
[02/12/20]seed@VM:~$


En
12:20 AM
tharoon@VM: /home/seed 66x25
[02/15/20]seed@VM:~$ ls -l /etc/zzz
-rw-r--r-- 1 root root 15 Feb 12 17:20 /etc/zzz
[02/15/20]seed@VM:~$