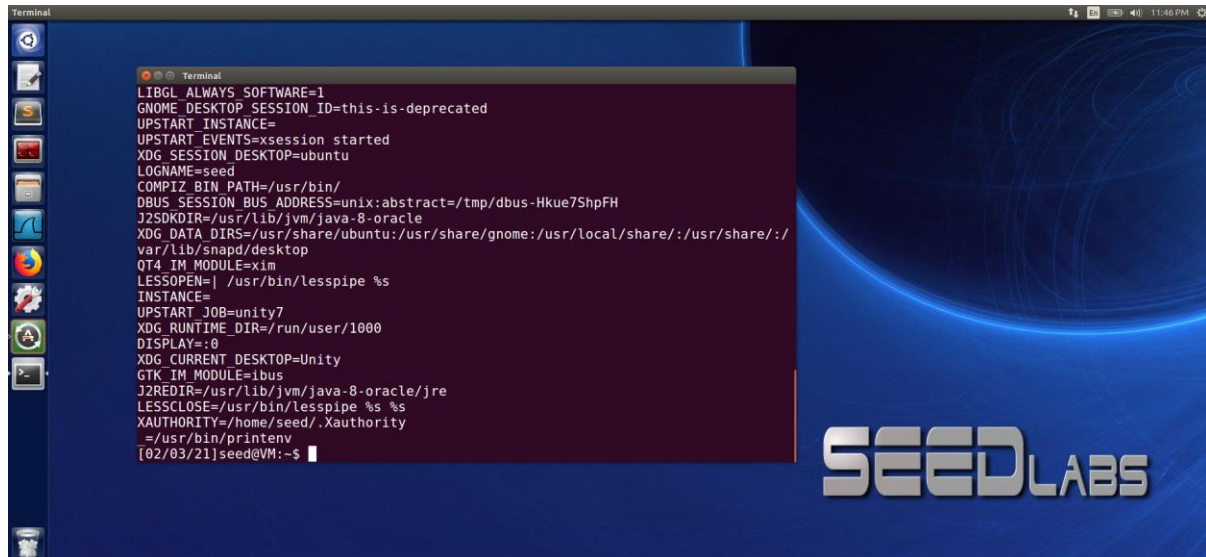


Name: Varun Pande

UTA-ID: 1001722538

## 2.1 Task 1: Manipulating Environment Variables:

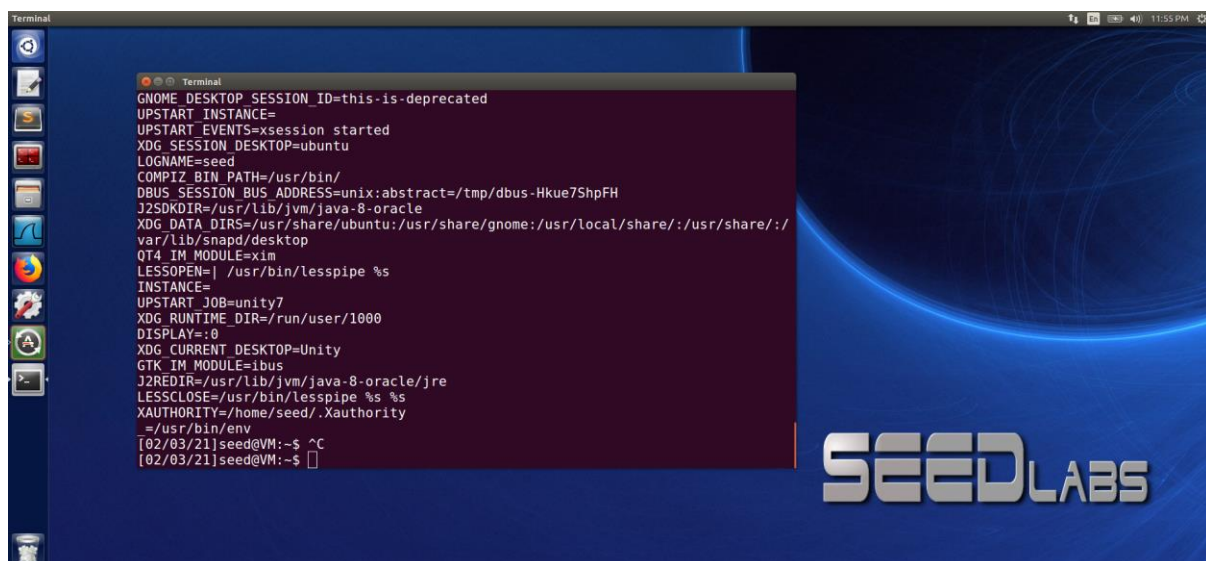
Cmd : printenv

A terminal window with a dark blue background and a 'SEEDLABS' logo on the right. The terminal displays the output of the 'printenv' command, listing various environment variables such as LIBGL\_ALWAYS\_SOFTWARE, GNOME\_DESKTOP\_SESSION\_ID, UPSTART\_INSTANCE, UPSTART\_EVENTS, XDG\_SESSION\_DESKTOP, LOGNAME, COMPIZ\_BIN\_PATH, DBUS\_SESSION\_BUS\_ADDRESS, J2SDKDIR, XDG\_DATA\_DIRS, QT4\_IM\_MODULE, LESSOPEN, INSTANCE, UPSTART\_JOB, XDG\_RUNTIME\_DIR, DISPLAY, XDG\_CURRENT\_DESKTOP, GTK\_IM\_MODULE, J2REDIR, LESSCLOSE, XAUTHORITY, and the command path. The prompt is '[02/03/21]seed@VM:~\$'.

### Observation:

On executing the “printenv” command we can see that all the set environment variables get out putted to shell. Special ENV Variables like PATH are used to point the path of common executables, so that they can directly made to run simply by typing the name of program.

Cmd : env

A terminal window with a dark blue background and a 'SEEDLABS' logo on the right. The terminal displays the output of the 'env' command, showing a subset of the environment variables listed in the previous screenshot, including GNOME\_DESKTOP\_SESSION\_ID, UPSTART\_INSTANCE, UPSTART\_EVENTS, XDG\_SESSION\_DESKTOP, LOGNAME, COMPIZ\_BIN\_PATH, DBUS\_SESSION\_BUS\_ADDRESS, J2SDKDIR, XDG\_DATA\_DIRS, QT4\_IM\_MODULE, LESSOPEN, INSTANCE, UPSTART\_JOB, XDG\_RUNTIME\_DIR, DISPLAY, XDG\_CURRENT\_DESKTOP, GTK\_IM\_MODULE, J2REDIR, LESSCLOSE, XAUTHORITY, and the command path. The prompt is '[02/03/21]seed@VM:~\$'.

### Observation:

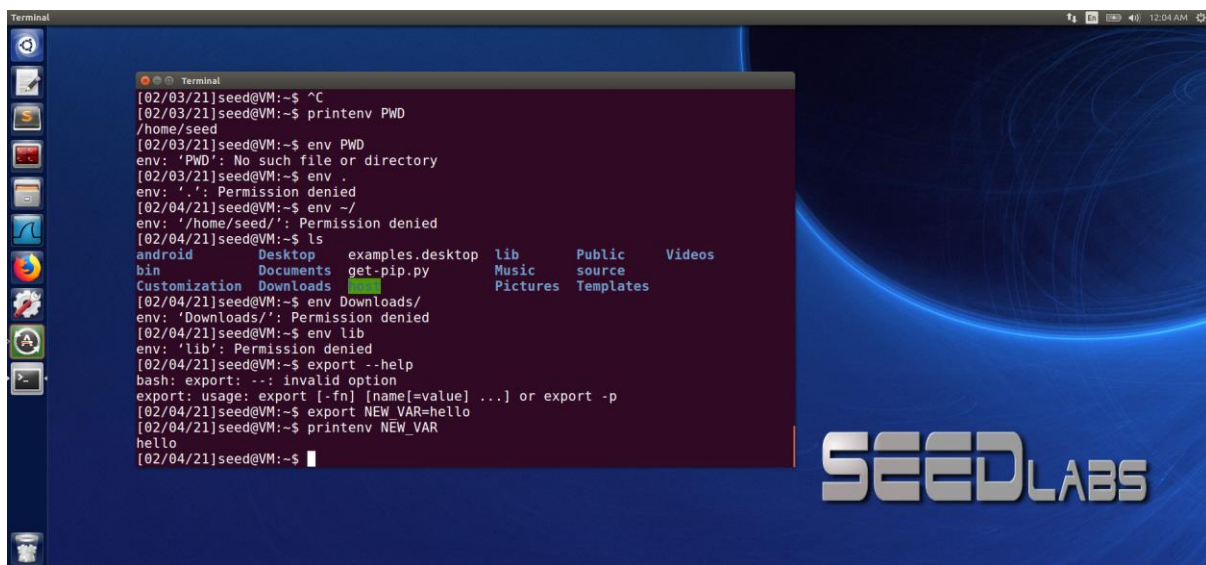
The same variable can be observed using the env command as well, but on careful observation we can see that at the end, we can figure out which command was used to display the variables.

**Cmd : printenv PWD or env | grep PWD**

**Observation:**

To filter the long list of ENV variables we can use printenv <name-of-variable> or env | grep <name-of-variable>. Note we cannot simply type “env PWD” it will result into an error.

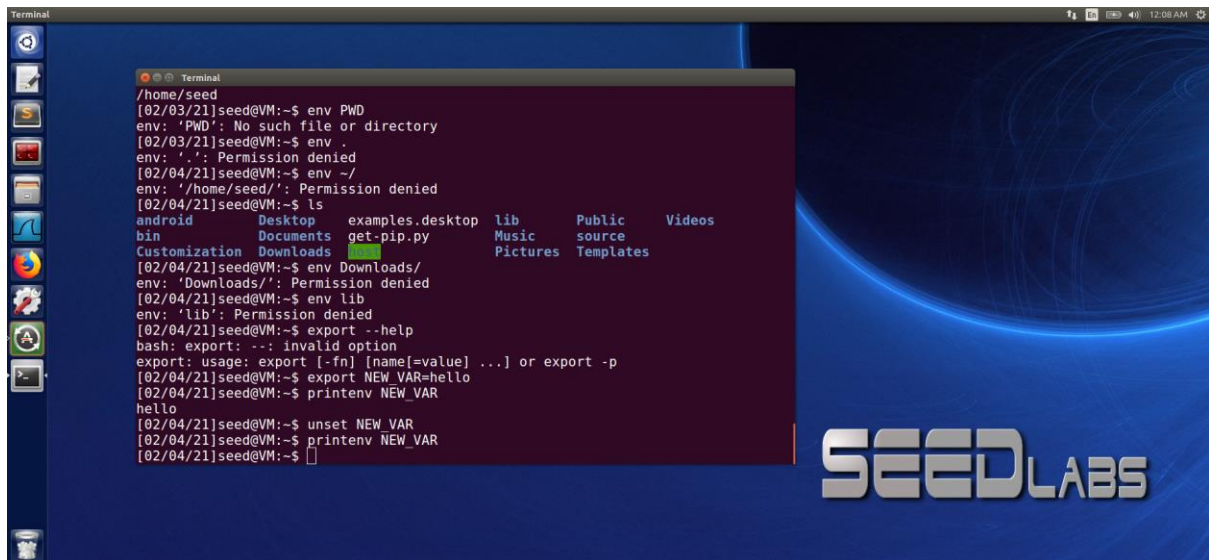
**Cmd : export <VAR\_Name>=<Value>**

A screenshot of a Linux terminal window with a dark blue background and a 'SEEDLABS' logo in the bottom right. The terminal shows a series of commands and their outputs. The user starts with 'printenv PWD', which returns '/home/seed'. Then they try 'env PWD', which returns an error: 'env: 'PWD': No such file or directory'. Next, they try 'env .', which returns 'env: '.': Permission denied'. Then they try 'env ~/', which returns 'env: '/home/seed/': Permission denied'. They then run 'ls', which lists the contents of the current directory: 'android', 'bin', 'Customization', 'Desktop', 'Downloads', 'examples.desktop', 'get-pip.py', 'lib', 'Music', 'Pictures', 'Public', 'source', 'Videos', and 'Templates'. They then try 'env Downloads/', which returns 'env: 'Downloads/': Permission denied'. Next, they try 'env lib', which returns 'env: 'lib': Permission denied'. They then run 'export --help', which returns 'bash: export: --: invalid option' and 'export: usage: export [-fn] [name[=value] ...] or export -p'. Finally, they run 'export NEW\_VAR=hello' and 'printenv NEW\_VAR', which returns 'hello'.

**Observation:**

We can use export to create new ENV variables or set values of the one that already exist, and then run printenv to check if the variables are set. Sometimes we need to restart the bash terminal in order for the change to take place.

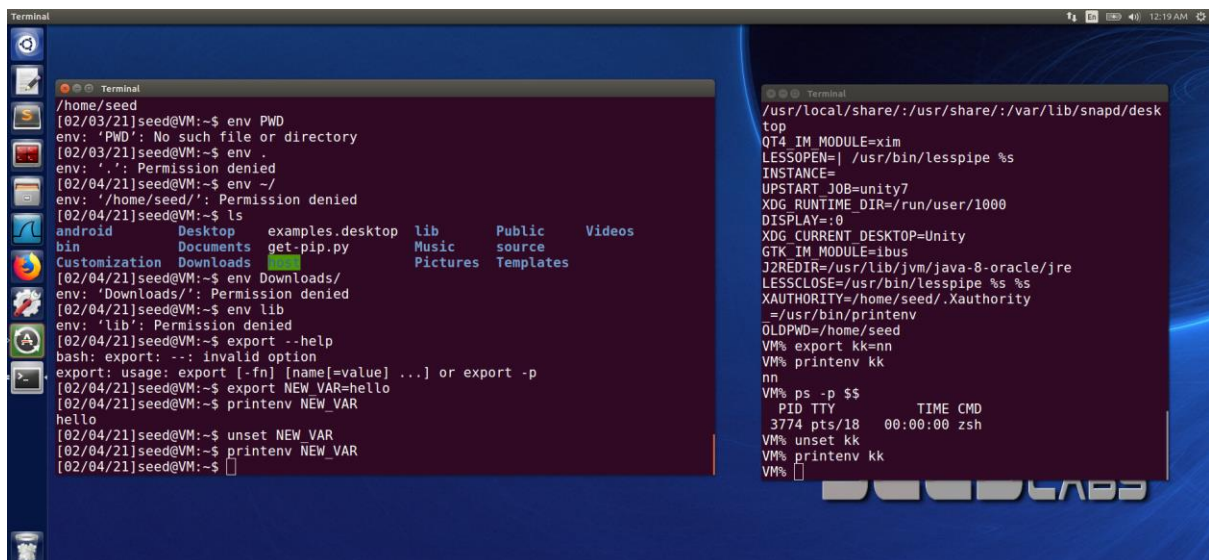
**Cmd : unset <ENV\_VAR\_name>**



```
/home/seed
[02/03/21]seed@VM:~$ env PWD
env: 'PWD': No such file or directory
[02/03/21]seed@VM:~$ env .
env: '.': Permission denied
[02/04/21]seed@VM:~$ env ~
env: '/home/seed/': Permission denied
[02/04/21]seed@VM:~$ ls
android      Desktop      examples.desktop  lib      Public      Videos
bin          Documents    get-pip.py        Music     source
Customization Downloads     [redacted]         Pictures  Templates
[02/04/21]seed@VM:~$ env Downloads/
env: 'Downloads/': Permission denied
[02/04/21]seed@VM:~$ env lib
env: 'lib': Permission denied
[02/04/21]seed@VM:~$ export --help
bash: export: --: invalid option
export: usage: export [-fn] [name[=value] ...] or export -p
[02/04/21]seed@VM:~$ export NEW_VAR=hello
[02/04/21]seed@VM:~$ printenv NEW_VAR
hello
[02/04/21]seed@VM:~$ unset NEW_VAR
[02/04/21]seed@VM:~$ printenv NEW_VAR
[02/04/21]seed@VM:~$
```

### Observation:

The unset command is used to delete the entry of ENV variable. On trying the above commands on “zsh” they seem to work just fine:



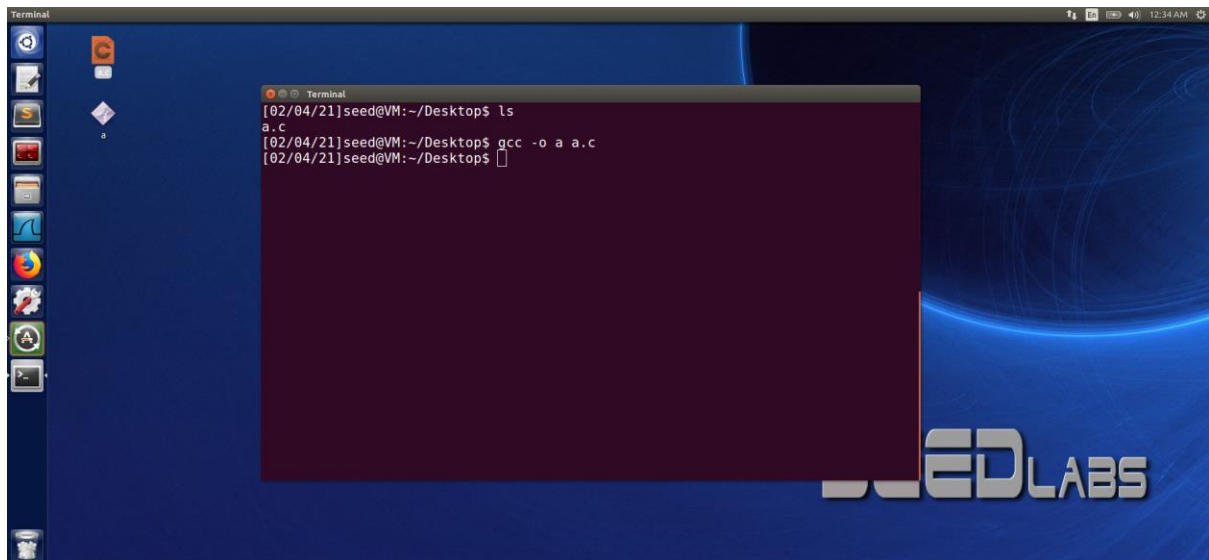
```
/home/seed
[02/03/21]seed@VM:~$ env PWD
env: 'PWD': No such file or directory
[02/03/21]seed@VM:~$ env .
env: '.': Permission denied
[02/04/21]seed@VM:~$ env ~
env: '/home/seed/': Permission denied
[02/04/21]seed@VM:~$ ls
android      Desktop      examples.desktop  lib      Public      Videos
bin          Documents    get-pip.py        Music     source
Customization Downloads     [redacted]         Pictures  Templates
[02/04/21]seed@VM:~$ env Downloads/
env: 'Downloads/': Permission denied
[02/04/21]seed@VM:~$ env lib
env: 'lib': Permission denied
[02/04/21]seed@VM:~$ export --help
bash: export: --: invalid option
export: usage: export [-fn] [name[=value] ...] or export -p
[02/04/21]seed@VM:~$ export NEW_VAR=hello
[02/04/21]seed@VM:~$ printenv NEW_VAR
hello
[02/04/21]seed@VM:~$ unset NEW_VAR
[02/04/21]seed@VM:~$ printenv NEW_VAR
[02/04/21]seed@VM:~$
```

```
/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
=/usr/bin/printenv
OLDPWD=/home/seed
VM% export kk=nn
VM% printenv kk
nn
VM% ps -p $$
PID TTY          TIME CMD
3774 pts/18    00:00:00 zsh
VM% unset kk
VM% printenv kk
VM%
```

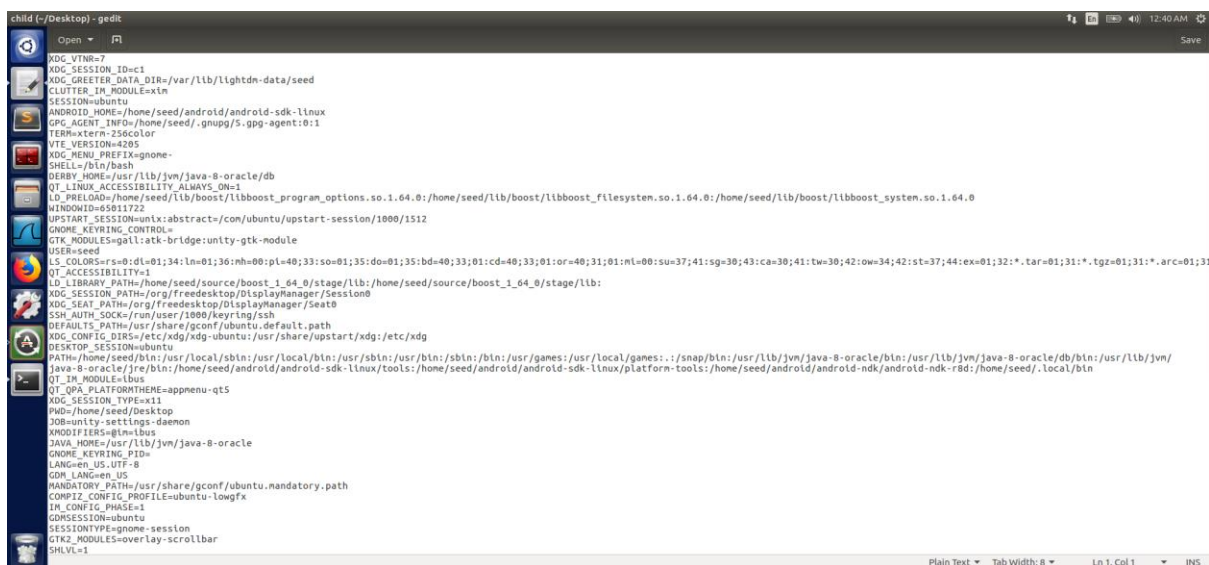
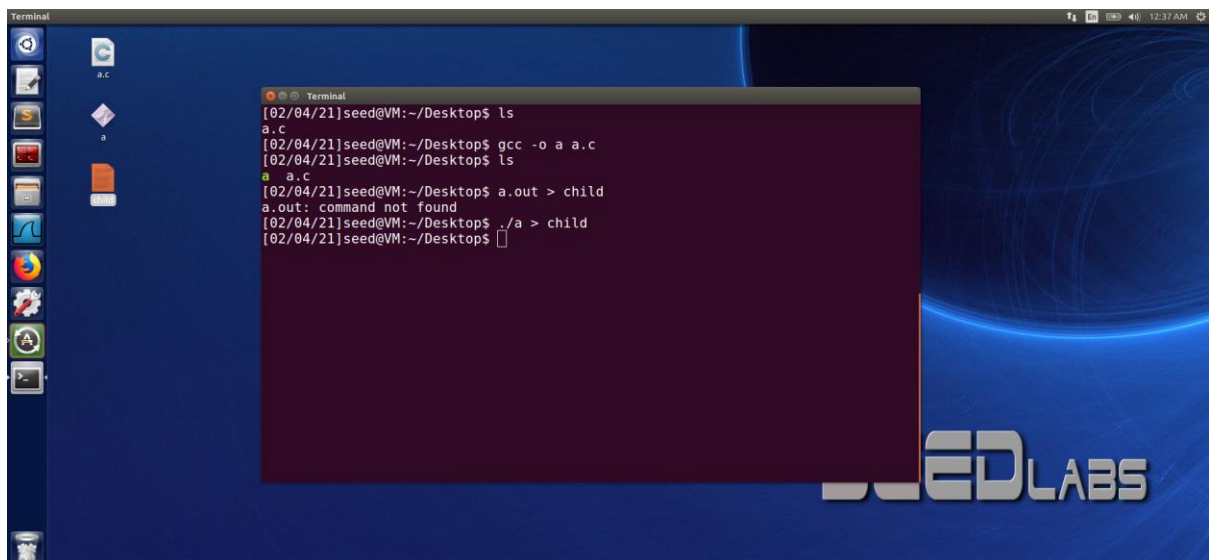
## 2.2 Task 2: Passing Environment Variables from Parent Process to Child Process

### Step 1:

Created file “a.c” using vim on Desktop, compiled it using “gcc -o a a.c”



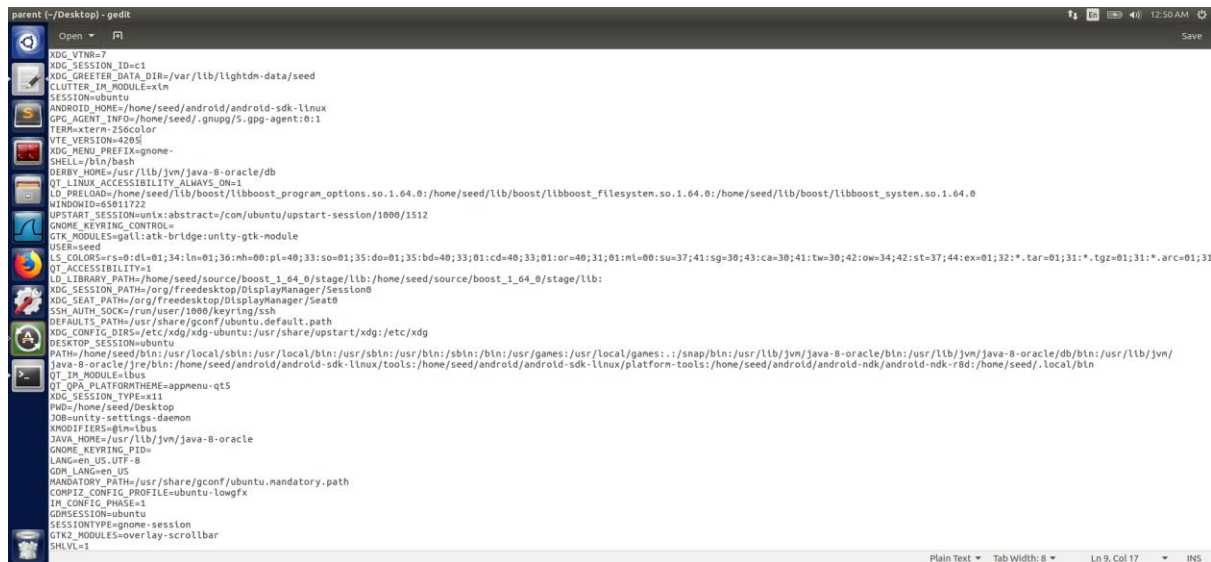
## Observation:





On executing the above program all the environment variables of the forked child process got saved to child.txt file.

## Step 2:

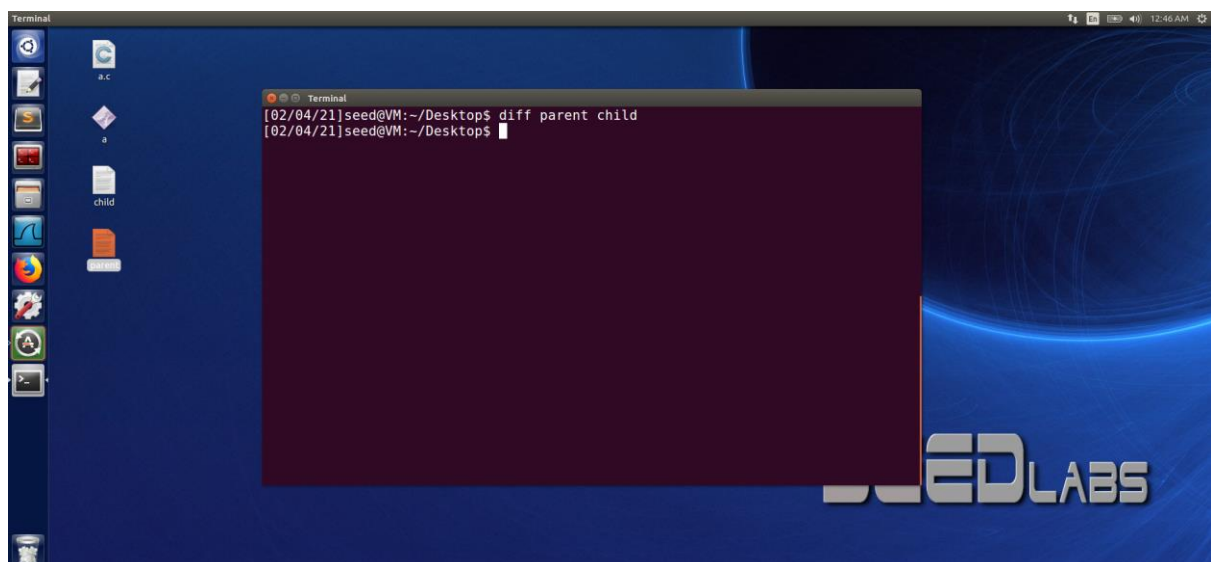


```
parent (~/Desktop) - gedit
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
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_options.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=65811722
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1512
GNOME_KEYRING_CONTROL=
GTK_MODULES=gall:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pl=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:ml=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:*.arc=01;31
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
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_OPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
PWD=/home/seed/Desktop
JOB=unity-settings-daemon
XMODIFIERS=@fn=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-lowfx
IM_CONFIG_PHASE=1
COROSION=ubuntu
SESSIONTYPE=gnome-session
GTK2_MODULES=overlay-scrollbar
XIMVL=1
```

By commenting and uncommenting the required lines, now on executing the program the ENV variables of the parent process was saved in the parent.txt file.

## Step 3:

On performing a diff on the files no difference was observed, because the fork command creates a duplicate of parent process.

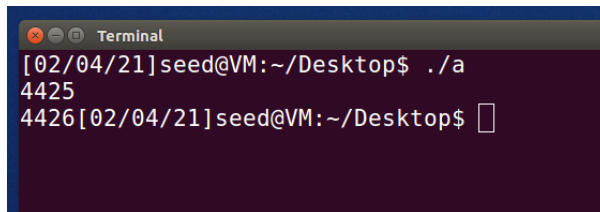


By Using the below command I have tried to print the process ID of the child and parent process:

```
printf("%d\n", getpid());
```

```
printf("%d",getpid());
```

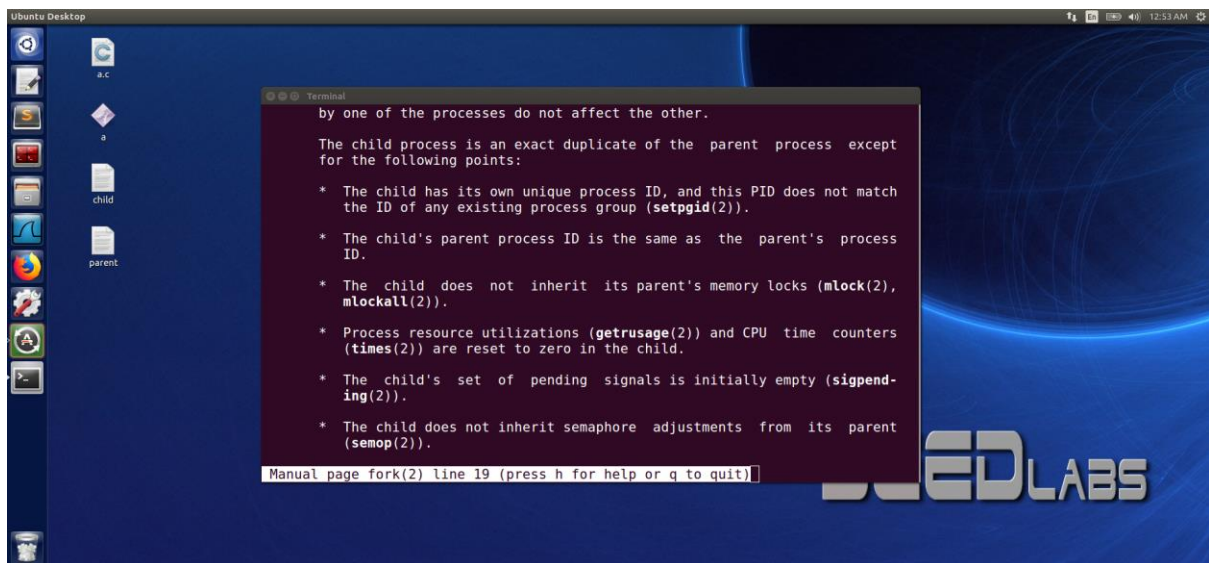
following is the output:



```
[02/04/21]seed@VM:~/Desktop$ ./a
4425
4426[02/04/21]seed@VM:~/Desktop$
```

It can be clearly seen from here that the parent and child have different PID's

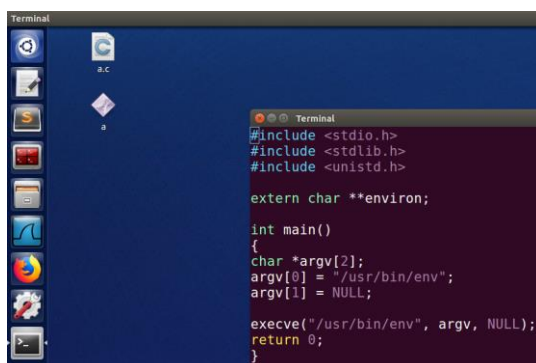
These are some of the differences that are highlighted on the man page of fork:



### 2.3 Task 3: Environment Variables and execve():

Step 1:

Compile and run the given code:



```
[02/04/21]seed@VM:~/Desktop$ cat a.c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

extern char **environ;

int main()
{
    char *argv[2];
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;

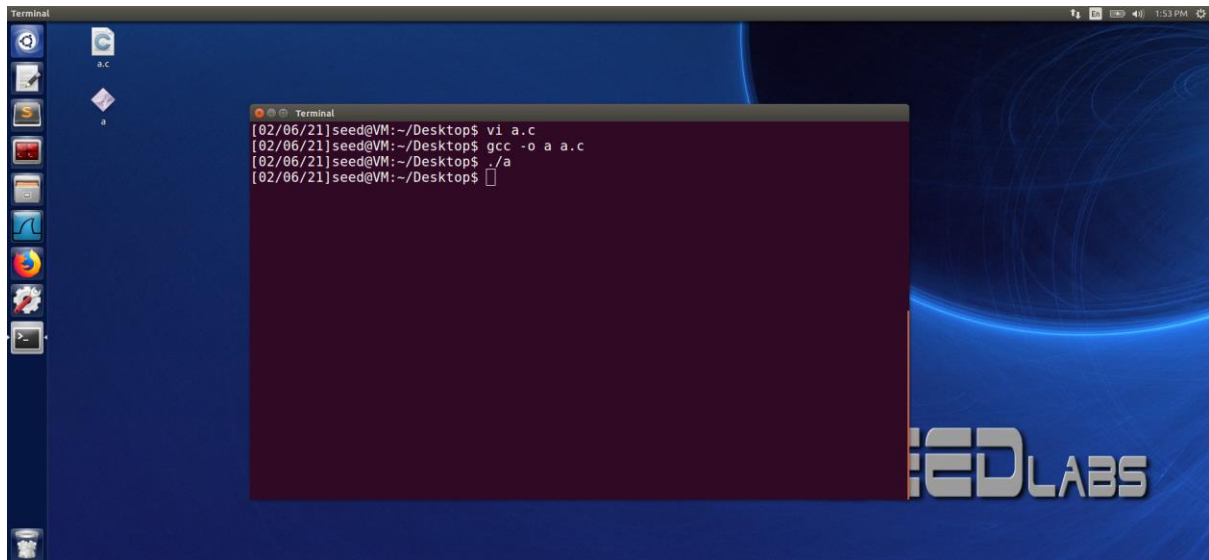
    execve("/usr/bin/env", argv, NULL);
    return 0;
}

[02/04/21]seed@VM:~/Desktop$ gcc a.c -o a
[02/04/21]seed@VM:~/Desktop$ ./a
[02/04/21]seed@VM:~/Desktop$
```

On executing the below code:

```
argv[0] = "/usr/bin/env";  
argv[1] = NULL;  
execve("/usr/bin/env", argv, NULL);
```

There was no output at all i.e. nothing was returned from the function.

A terminal window on a Linux desktop with a blue background and 'SEED LABS' logo. The terminal shows the following commands and output:

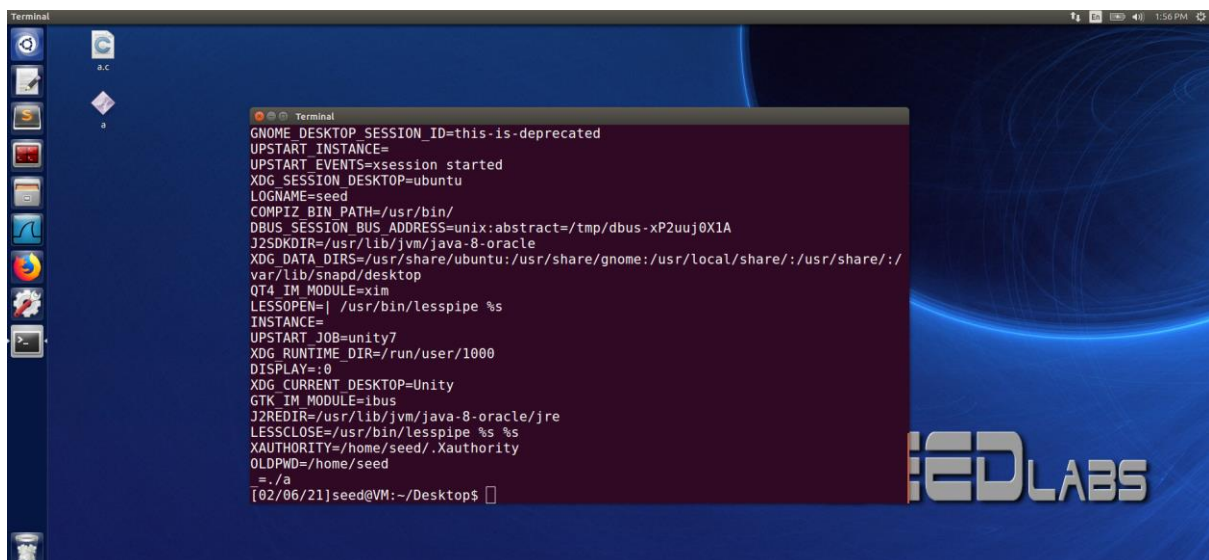
```
[02/06/21]seed@VM:~/Desktop$ vi a.c  
[02/06/21]seed@VM:~/Desktop$ gcc -o a a.c  
[02/06/21]seed@VM:~/Desktop$ ./a  
[02/06/21]seed@VM:~/Desktop$
```

Step 2:

After changing to following code:

```
execve("/usr/bin/env", argv, environ);
```

Now we see the environment variables getting printed (also we can see from the last line that this function was called by the written code):

A terminal window on a Linux desktop with a blue background and 'SEED LABS' logo. The terminal shows the following output after running the program:

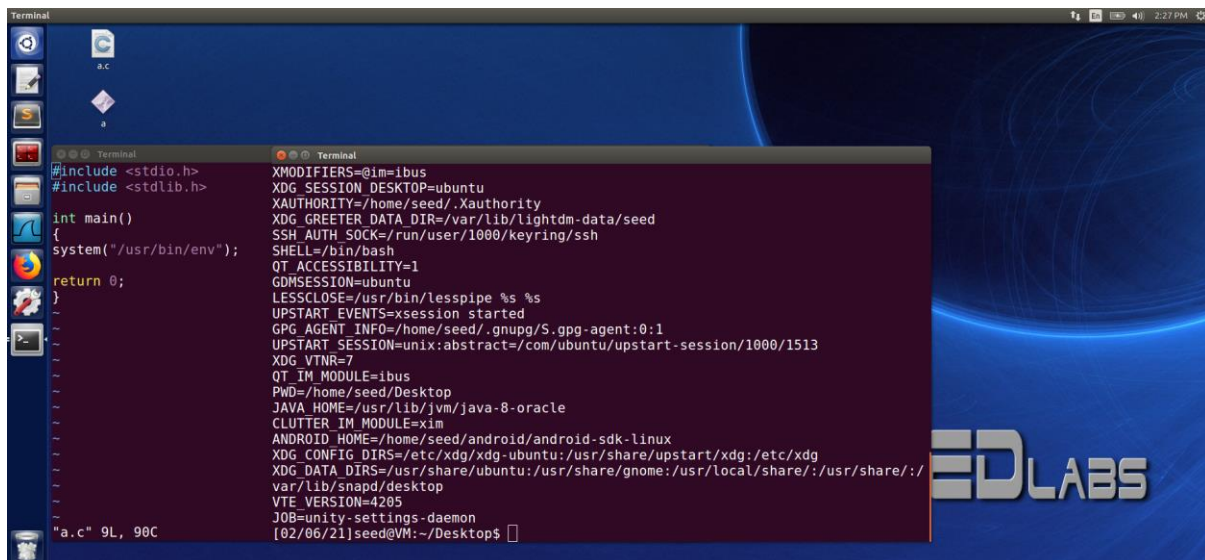
```
GNOME_DESKTOP_SESSION_ID=this-is-deprecated  
UPSTART_INSTANCE=  
UPSTART_EVENTS=xsession started  
XDG_SESSION_DESKTOP=ubuntu  
LOGNAME=seed  
COMPIZ_BIN_PATH=/usr/bin/  
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-xP2uuj0X1A  
J2SDKDIR=/usr/lib/jvm/java-8-oracle  
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share:/usr/share:/  
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  
OLDPWD=/home/seed  
./a  
[02/06/21]seed@VM:~/Desktop$
```

Step 3:

The third argument to `execve()` is the array of strings of environment variables. In the given code below lines were used to get the list of environment variables and pass it to the `execve` function.

```
extern char **environ;  
  
execve("/usr/bin/env", argv, environ);
```

## 2.4 Task 4: Environment Variables and `system()`



```
Terminal  
#include <stdio.h>  
#include <stdlib.h>  
  
int main()  
{  
    system("/usr/bin/env");  
    return 0;  
}  
"a.c" 9L, 90C  
XMODIFIERS=@im=ibus  
XDG_SESSION_DESKTOP=ubuntu  
XAUTHORITY=/home/seed/.Xauthority  
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed  
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh  
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/1000/1513  
XDG_VTNR=7  
QT_IM_MODULE=ibus  
PWD=/home/seed/Desktop  
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/xdg  
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share:/usr/share/:/  
var/lib/snapd/desktop  
VTE_VERSION=4205  
JOB=unity-settings-daemon  
[02/06/21]seed@VM:~/Desktop$
```

### Observation:

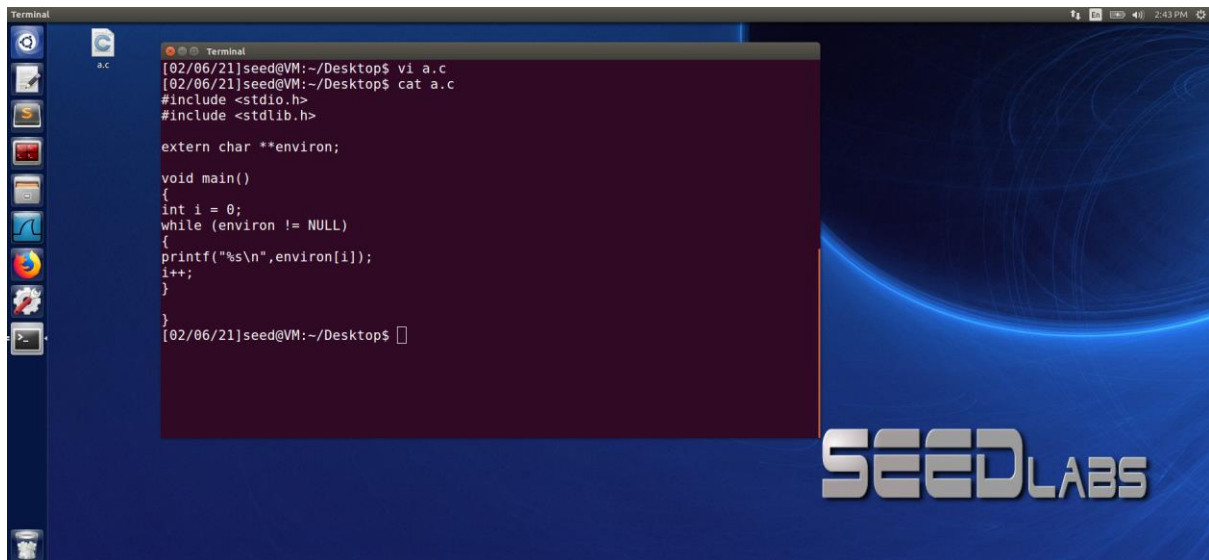
From the above screenshot we don't need to pass in the environment variable and also we can see that using `system` function internally calls a separate function to run "execve" as the last line does not show any "`_=./xx`" being used to execute the "execve" cmd unlike that in task 3.

## 2.5 Task 5: Environment Variable and Set-UID Programs

Step 1:

Writing program.





```
Terminal
[02/06/21]seed@VM:~/Desktop$ vi a.c
[02/06/21]seed@VM:~/Desktop$ cat a.c
#include <stdio.h>
#include <stdlib.h>

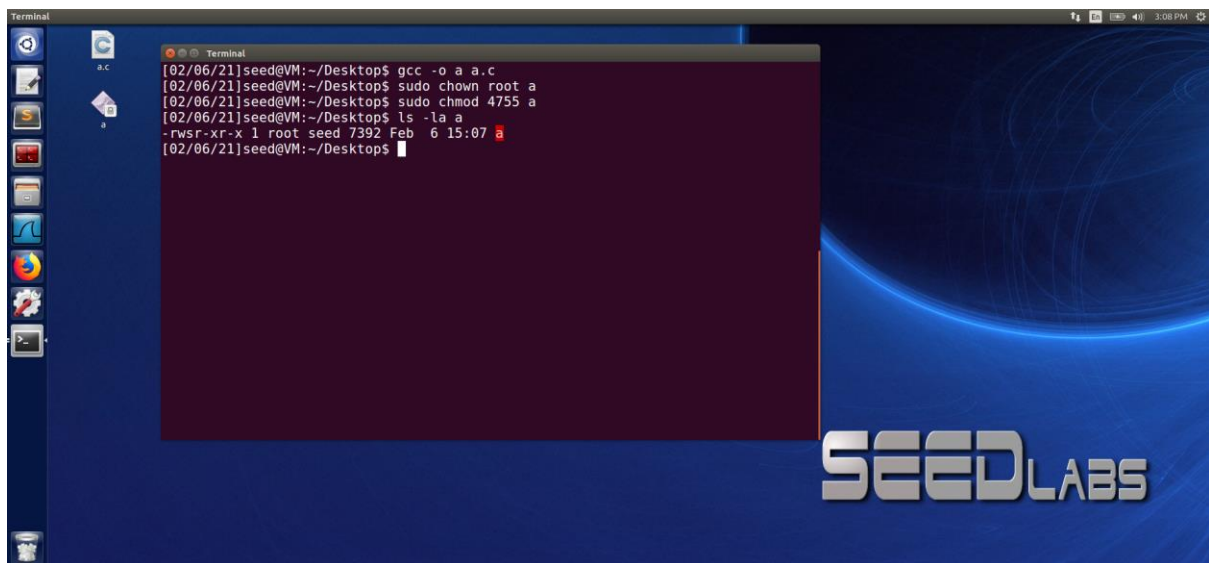
extern char **environ;

void main()
{
    int i = 0;
    while (environ != NULL)
    {
        printf("%s\n", environ[i]);
        i++;
    }
}
[02/06/21]seed@VM:~/Desktop$
```

SEEDLABS

Step 2:

Compile, change its ownership to root, and make it a Set-UID program:



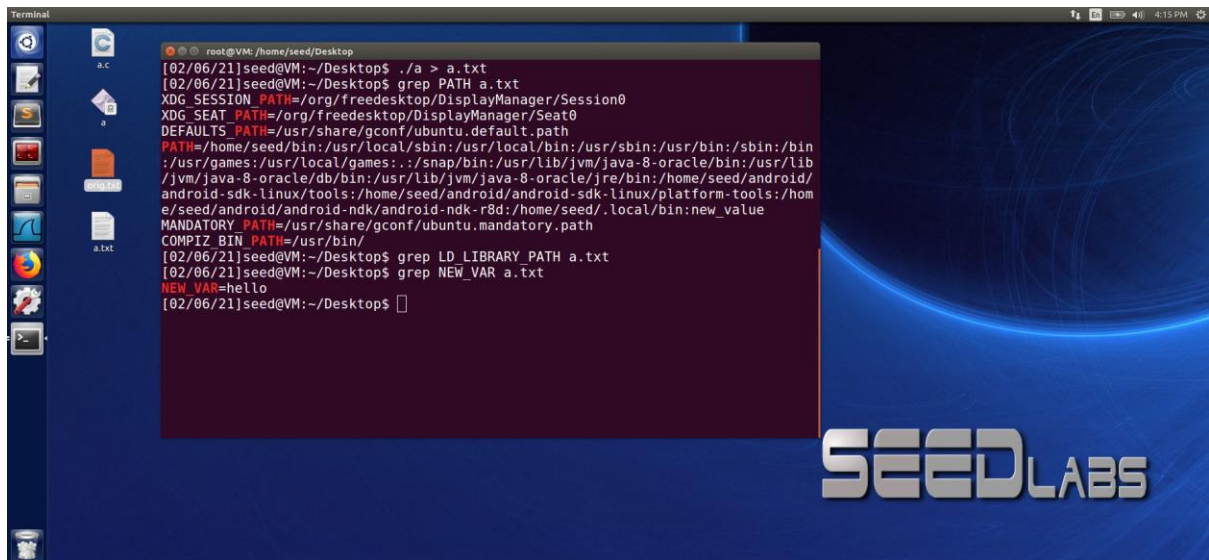
```
Terminal
[02/06/21]seed@VM:~/Desktop$ gcc -o a a.c
[02/06/21]seed@VM:~/Desktop$ sudo chown root a
[02/06/21]seed@VM:~/Desktop$ sudo chmod 4755 a
[02/06/21]seed@VM:~/Desktop$ ls -la a
-rwsr-xr-x 1 root seed 7392 Feb  6 15:07 a
[02/06/21]seed@VM:~/Desktop$
```

SEEDLABS

Step 3:

Checked the variable PATH, LD\_LIBRARY\_PATH and set the NEW\_VAR variable using export.  
(new\_value is appended to all the variables.)





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

Writing the program:

```
Terminal
[02/06/21]seed@VM:~/Desktop$ cat a.c
int main()
{
    system("ls");
    return 0;
}
[02/06/21]seed@VM:~/Desktop$
```

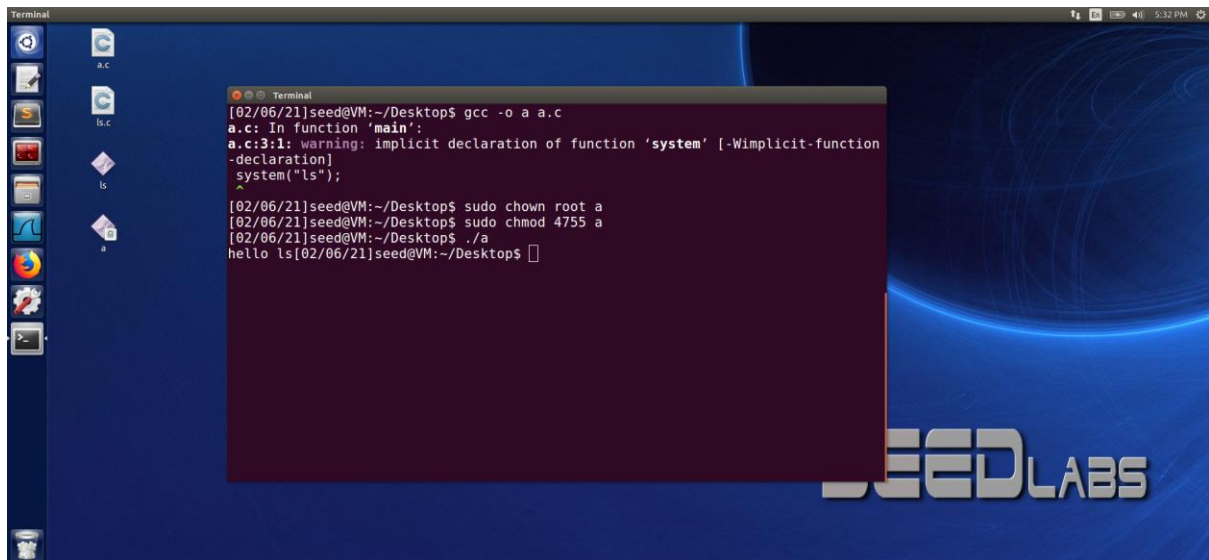
My custom program:

```
Terminal
[02/06/21]seed@VM:~/Desktop$ cat ls.c
#include <stdio.h>

void main()
{
    printf("hello ls");
}
[02/06/21]seed@VM:~/Desktop$
```

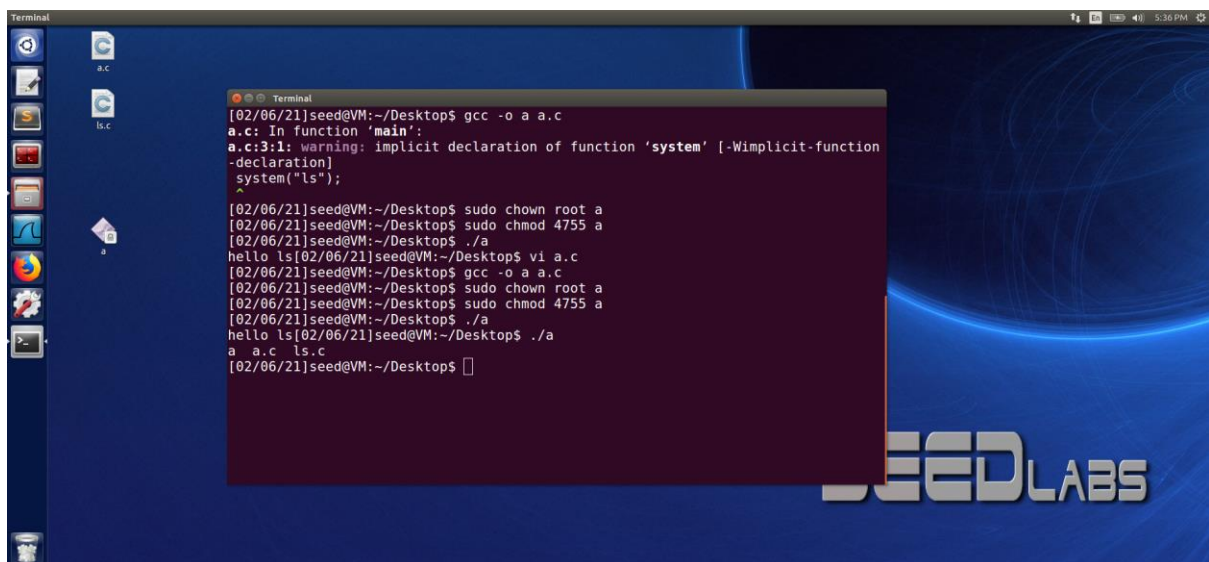
Pre-pending the path of Desktop so that when the program searches for 'ls' implementation while linking it reaches to my custom implementation. On compiling and executing we can see that it prints "hello ls".

```
export PATH=~/.Desktop:$PATH
```



```
Terminal
[02/06/21]seed@VM:~/Desktop$ gcc -o a a.c
a.c: In function 'main':
a.c:3:1: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
system("ls");
^
[02/06/21]seed@VM:~/Desktop$ sudo chown root a
[02/06/21]seed@VM:~/Desktop$ sudo chmod 4755 a
[02/06/21]seed@VM:~/Desktop$ ./a
hello ls[02/06/21]seed@VM:~/Desktop$
```

If I move my "ls" implementation to some place else it runs the /bin/ls implementation



```
Terminal
[02/06/21]seed@VM:~/Desktop$ gcc -o a a.c
a.c: In function 'main':
a.c:3:1: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
system("ls");
^
[02/06/21]seed@VM:~/Desktop$ sudo chown root a
[02/06/21]seed@VM:~/Desktop$ sudo chmod 4755 a
[02/06/21]seed@VM:~/Desktop$ ./a
hello ls[02/06/21]seed@VM:~/Desktop$ vi a.c
[02/06/21]seed@VM:~/Desktop$ gcc -o a a.c
[02/06/21]seed@VM:~/Desktop$ sudo chown root a
[02/06/21]seed@VM:~/Desktop$ sudo chmod 4755 a
[02/06/21]seed@VM:~/Desktop$ ./a
hello ls[02/06/21]seed@VM:~/Desktop$ ./a
a a.c ls.c
[02/06/21]seed@VM:~/Desktop$
```

On adding the following line to my implementation of ls to see the EUID and real UID:



```
#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

void main()
{
    int real = getuid();
    int euid = geteuid();

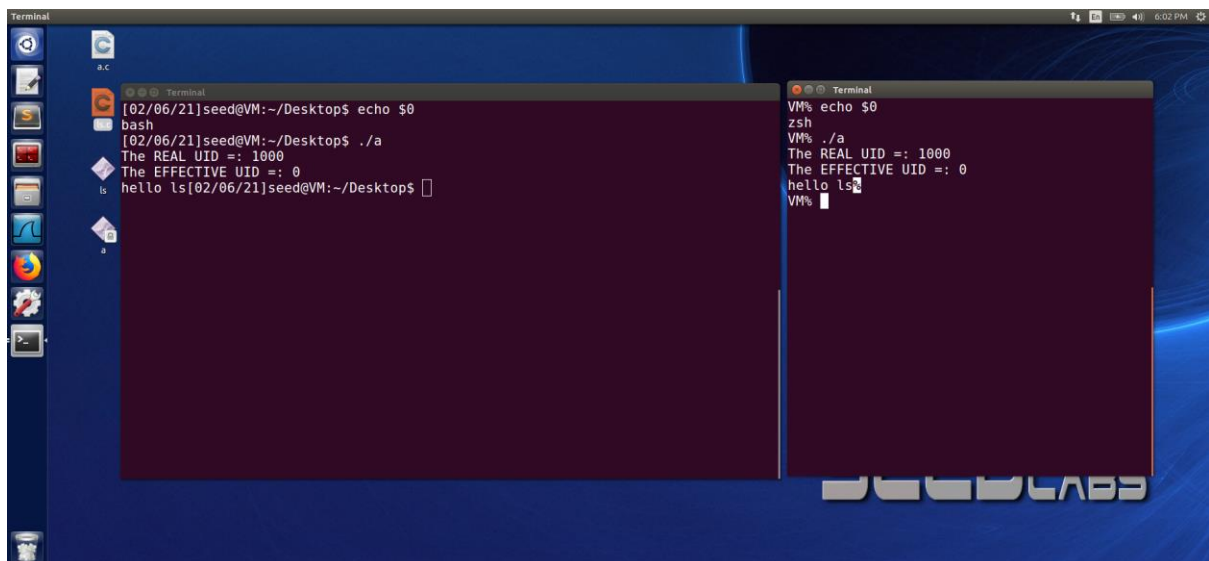
    printf("The REAL UID =: %d\n", real);

    printf("The EFFECTIVE UID =: %d\n",
    euid);

    printf("hello ls");

}
```

The output is same for both the shells (zsh and bash):

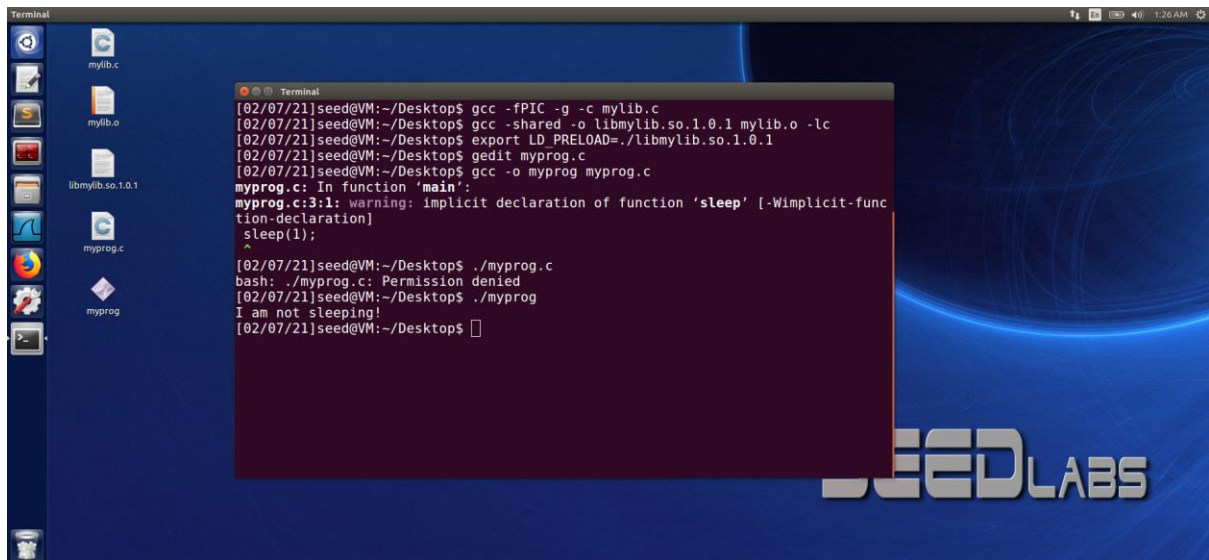


```
Terminal
[02/06/21]seed@VM:~/Desktop$ echo $0
bash
[02/06/21]seed@VM:~/Desktop$ ./a
The REAL UID =: 1000
The EFFECTIVE UID =: 0
hello ls[02/06/21]seed@VM:~/Desktop$

Terminal
VM% echo $0
zsh
VM% ./a
The REAL UID =: 1000
The EFFECTIVE UID =: 0
hello ls
VM%
```

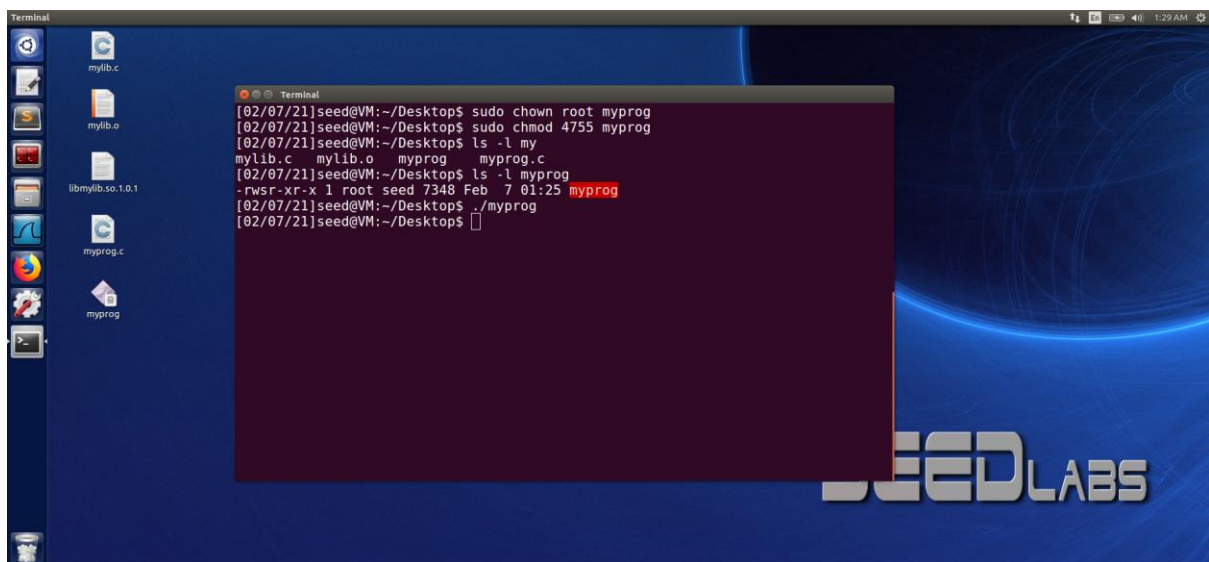
## 2.7 Task 7: The LD PRELOAD Environment Variable and Set-UID Programs:

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



```
Terminal
[02/07/21]seed@VM:~/Desktop$ gcc -fPIC -g -c mylib.c
[02/07/21]seed@VM:~/Desktop$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[02/07/21]seed@VM:~/Desktop$ export LD_PRELOAD=./libmylib.so.1.0.1
[02/07/21]seed@VM:~/Desktop$ gedit myprog.c
[02/07/21]seed@VM:~/Desktop$ gcc -o myprog myprog.c
myprog.c: In function 'main':
myprog.c:3:1: warning: implicit declaration of function 'sleep' [-Wimplicit-func
tion-declaration]
sleep(1);
^
[02/07/21]seed@VM:~/Desktop$ ./myprog.c
bash: ./myprog.c: Permission denied
[02/07/21]seed@VM:~/Desktop$ ./myprog
I am not sleeping!
[02/07/21]seed@VM:~/Desktop$
```

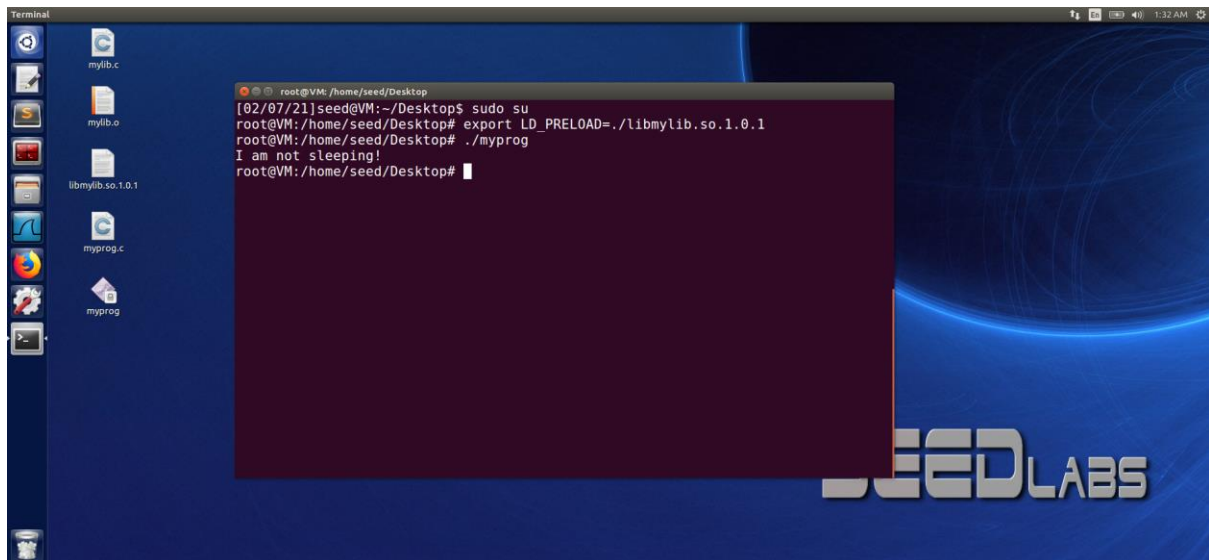
2. Make myprog a Set-UID root program, and run it as a normal user.



```
Terminal
[02/07/21]seed@VM:~/Desktop$ sudo chown root myprog
[02/07/21]seed@VM:~/Desktop$ sudo chmod 4755 myprog
[02/07/21]seed@VM:~/Desktop$ ls -l my
mylib.c  mylib.o  myprog  myprog.c
[02/07/21]seed@VM:~/Desktop$ ls -l myprog
-rwsr-xr-x 1 root seed 7348 Feb  7 01:25 myprog
[02/07/21]seed@VM:~/Desktop$ ./myprog
[02/07/21]seed@VM:~/Desktop$
```

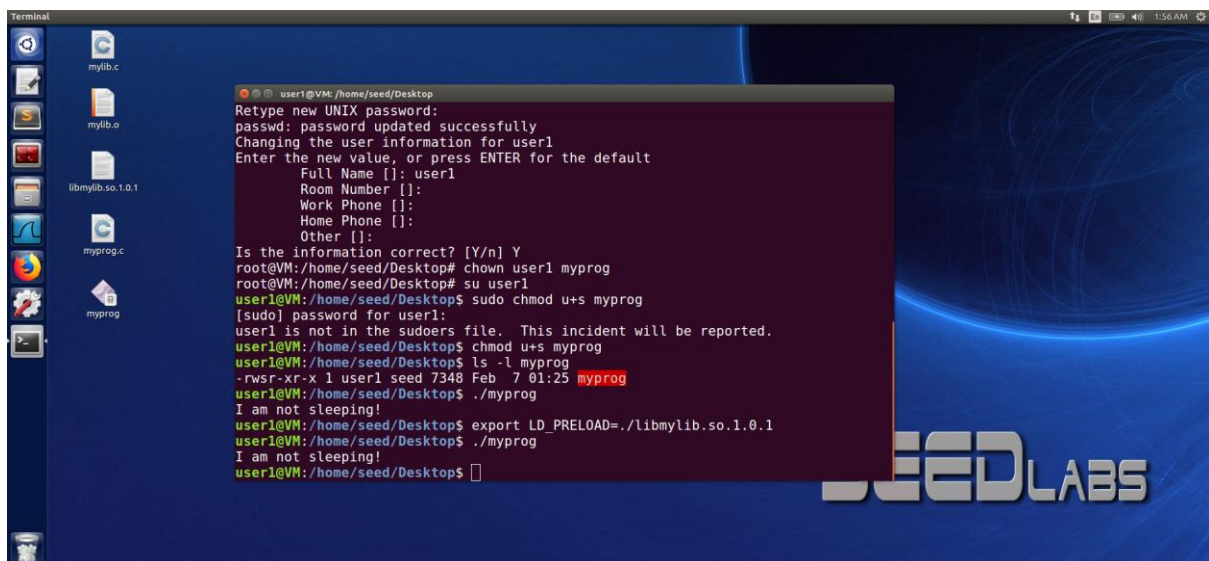
Observation: It paused for a second, i.e. it actually went to sleep for a second.

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



Observation: it ran the custom sleep function implemented by the malicious library.

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.

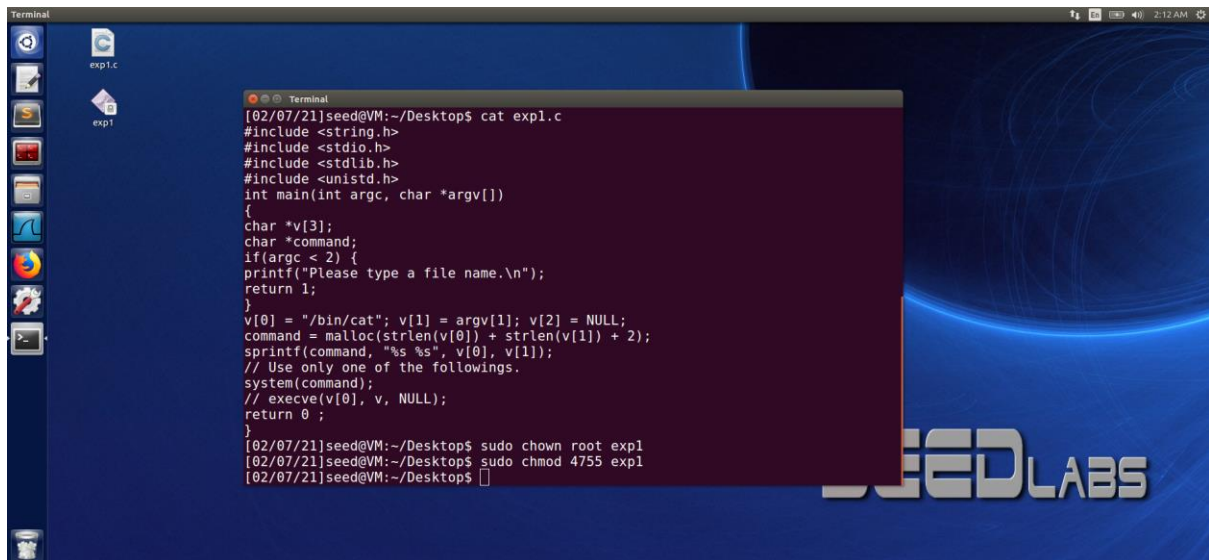


Observation: We can see that the malicious library created by us gets executed here.

## 2.8 (20 Points Total) Task 8: Invoking External Programs Using system() versus execve()

Step 1:

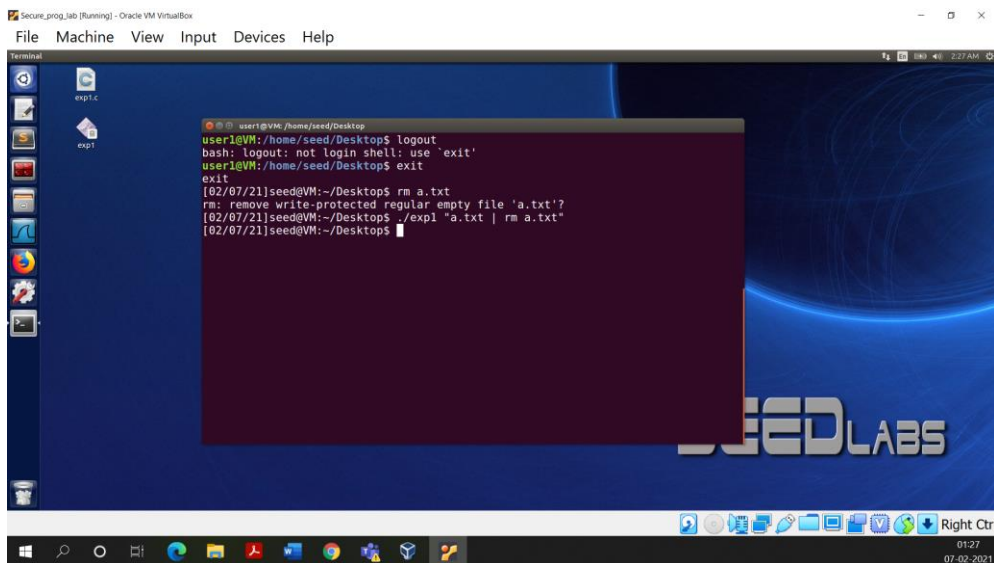
Executing the given code:



```
[02/07/21]seed@VM:~/Desktop$ cat expl.c
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(int argc, char *argv[])
{
    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;
}
[02/07/21]seed@VM:~/Desktop$ sudo chown root expl
[02/07/21]seed@VM:~/Desktop$ sudo chmod 4755 expl
[02/07/21]seed@VM:~/Desktop$
```

### Observation:

I created a new file a.txt and changed its owner to root. Without running the program provided I was not allowed to remove the file. When the script to read the file is implemented using system, I was able to use the rm command to remove the file.

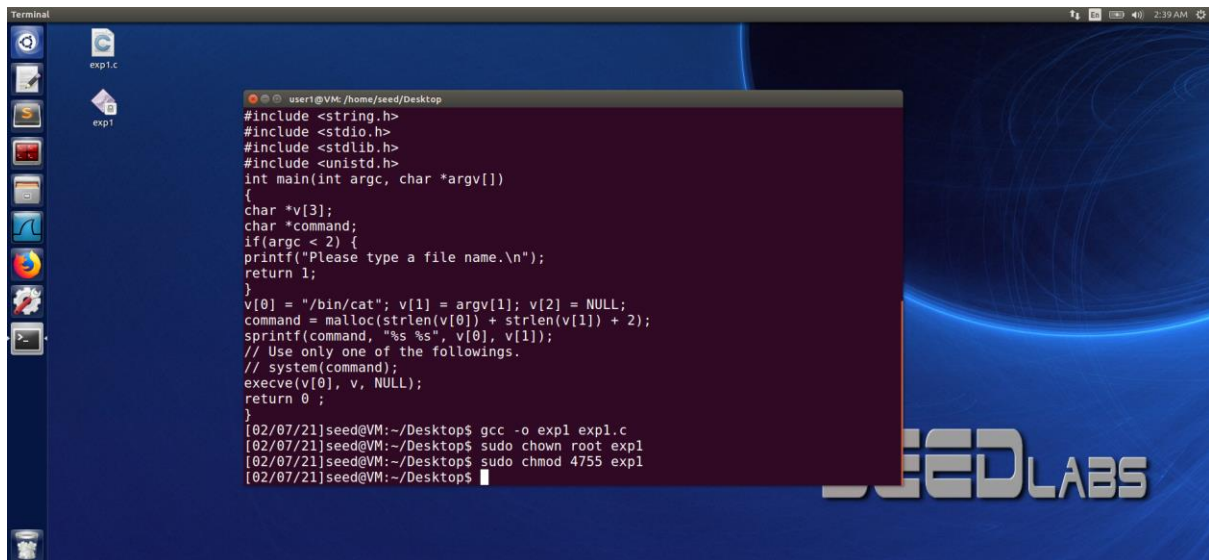


```
user1@VM:/home/seed/Desktop$ logout
bash: logout: not login shell: use 'exit'
user1@VM:/home/seed/Desktop$ exit
exit
[02/07/21]seed@VM:~/Desktop$ rm a.txt
rm: remove write-protected regular empty file 'a.txt'?
[02/07/21]seed@VM:~/Desktop$ ./expl "a.txt" | rm a.txt
[02/07/21]seed@VM:~/Desktop$
```

### Step 2:

Compiling the file again after changes, along with set-UID to root.



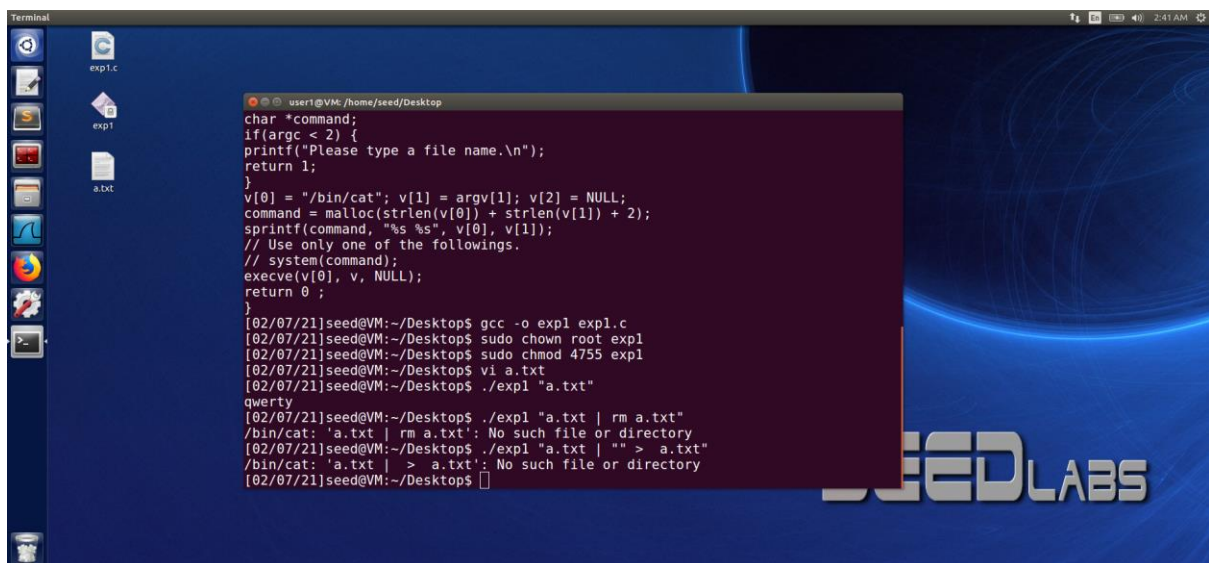


```
Terminal
expl.c
expl

user1@VM: /home/seed/Desktop
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
int main(int argc, char *argv[])
{
    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;
}
[02/07/21]seed@VM:~/Desktop$ gcc -o expl expl.c
[02/07/21]seed@VM:~/Desktop$ sudo chown root expl
[02/07/21]seed@VM:~/Desktop$ sudo chmod 4755 expl
[02/07/21]seed@VM:~/Desktop$
```

### Observation:

On executing the file reading script using the “execve” I was unable to remove the file nor perform any changes to it.

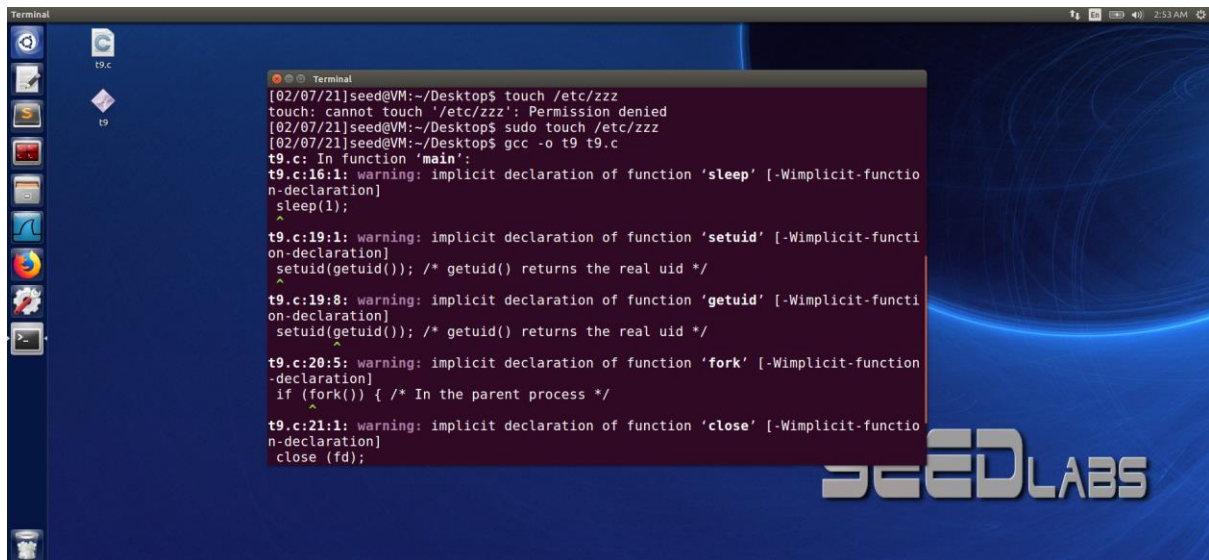


```
Terminal
expl.c
expl
a.txt

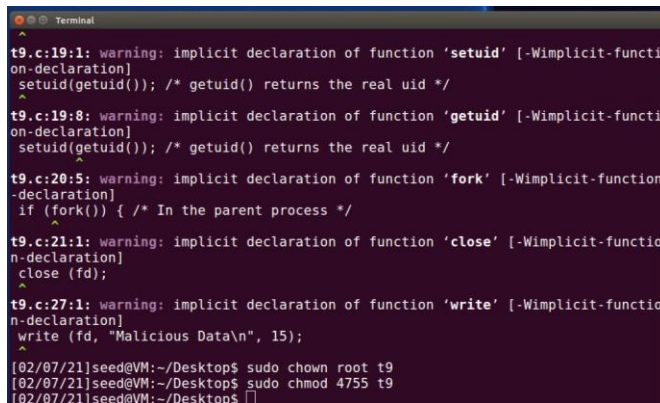
user1@VM: /home/seed/Desktop
char *command;
if(argc < 2) {
    printf("Please type a file name.\n");
    return 1;
}
v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;
command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
sprintf(command, "%s %s", v[0], v[1]);
// Use only one of the followings.
// system(command);
execve(v[0], v, NULL);
return 0;
}
[02/07/21]seed@VM:~/Desktop$ gcc -o expl expl.c
[02/07/21]seed@VM:~/Desktop$ sudo chown root expl
[02/07/21]seed@VM:~/Desktop$ sudo chmod 4755 expl
[02/07/21]seed@VM:~/Desktop$ vi a.txt
[02/07/21]seed@VM:~/Desktop$ ./expl "a.txt"
qwerty
[02/07/21]seed@VM:~/Desktop$ ./expl "a.txt | rm a.txt"
/bin/cat: 'a.txt | rm a.txt': No such file or directory
[02/07/21]seed@VM:~/Desktop$ ./expl "a.txt | "" > a.txt"
/bin/cat: 'a.txt | > a.txt': No such file or directory
[02/07/21]seed@VM:~/Desktop$
```

### 2.9 Task 9: Capability Leaking:

Creating the etc/zzz file and compiling the script.



```
[02/07/21]seed@VM:~/Desktop$ touch /etc/zzz
touch: cannot touch '/etc/zzz': Permission denied
[02/07/21]seed@VM:~/Desktop$ sudo touch /etc/zzz
[02/07/21]seed@VM:~/Desktop$ gcc -o t9 t9.c
t9.c: In function 'main':
t9.c:16:1: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
sleep(1);
^
t9.c:19:1: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
setuid(getuid()); /* getuid() returns the real uid */
^
t9.c:19:8: warning: implicit declaration of function 'getuid' [-Wimplicit-function-declaration]
setuid(getuid()); /* getuid() returns the real uid */
^
t9.c:20:5: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
if (fork()) { /* In the parent process */
^
t9.c:21:1: warning: implicit declaration of function 'close' [-Wimplicit-function-declaration]
close (fd);
^
```



```
t9.c:19:1: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
setuid(getuid()); /* getuid() returns the real uid */
^
t9.c:19:8: warning: implicit declaration of function 'getuid' [-Wimplicit-function-declaration]
setuid(getuid()); /* getuid() returns the real uid */
^
t9.c:20:5: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
if (fork()) { /* In the parent process */
^
t9.c:21:1: warning: implicit declaration of function 'close' [-Wimplicit-function-declaration]
close (fd);
^
t9.c:27:1: warning: implicit declaration of function 'write' [-Wimplicit-function-declaration]
write (fd, "Malicious Data\n", 15);
^
[02/07/21]seed@VM:~/Desktop$ sudo chown root t9
[02/07/21]seed@VM:~/Desktop$ sudo chmod 4755 t9
[02/07/21]seed@VM:~/Desktop$
```

## Observation:

From the below screenshot it is clear that the above script was able to modify the file “zzz”. Since the script is running with root privileges, the below highlighted line returns the root UID:

`setuid(getuid());`

Due to which the programs privileges are not reduced and post the process is forked the malicious code part executes with root privilege.

