Created by: **Eduardo Argueta Cantizzano**
Vulnerability: **Environment Variables and Set-UID Program**

Environment Variable and Set-UID Program Lab:

Environment Variables

Environment variables are dynamic values that affect the processes or programs running on a computer.

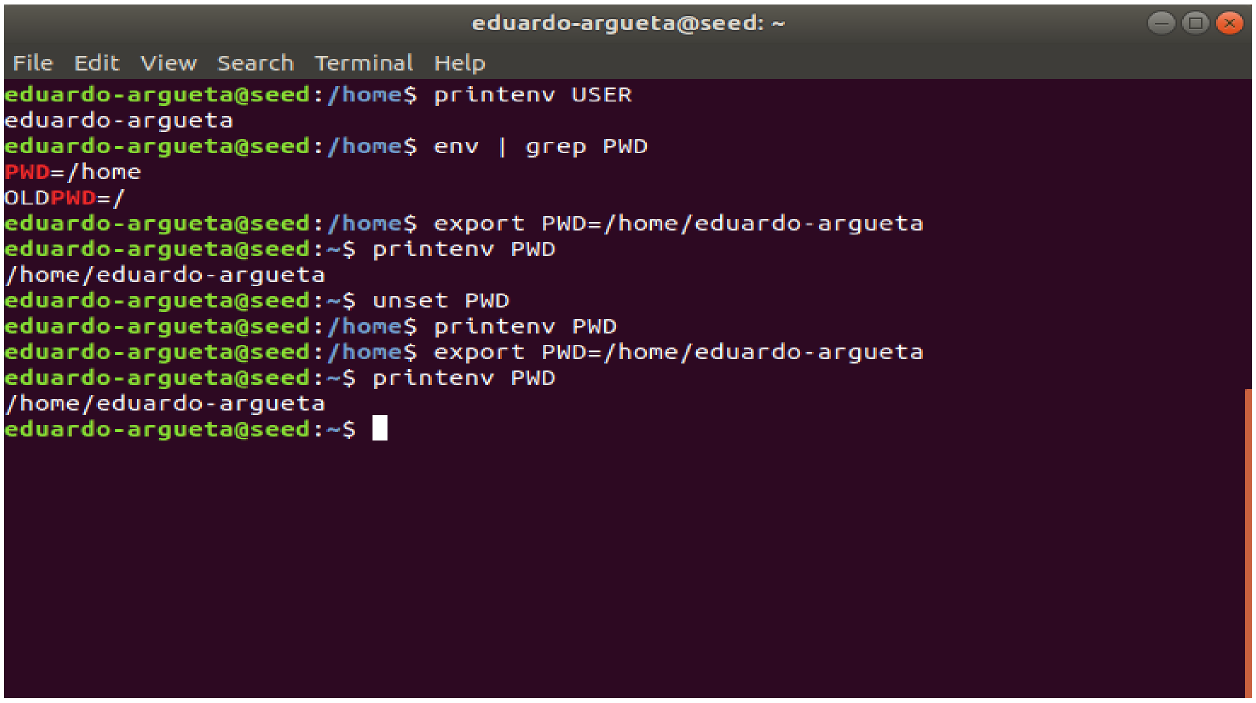
Set-UID Programs

Set-UID (Set User ID upon execution) is a Unix/Linux feature that allows users to run an executable file with the file owner's permissions rather than the permissions of the user running the executable.

The purpose of this lab is to understand how environment variables affect programs and system behaviors. I will be explaining how the environment variables work, how they are propagated from parent process to child, and how they affect system/program behaviors. It also explains how environment variables affect the behavior of *Set-UID* programs.

Task 1: Manipulating Environment Variables

In this task, I am using different commands to see, set and unset environment variables. So for this purpose, we can use ***printenv*** or ***env*** command which would list all the environment variables of our system. To see the value of some particular environment variable, we write that variable's name right after the printenv or env command. We use ***export*** command to set and ***unset*** command to unset any variable's value.

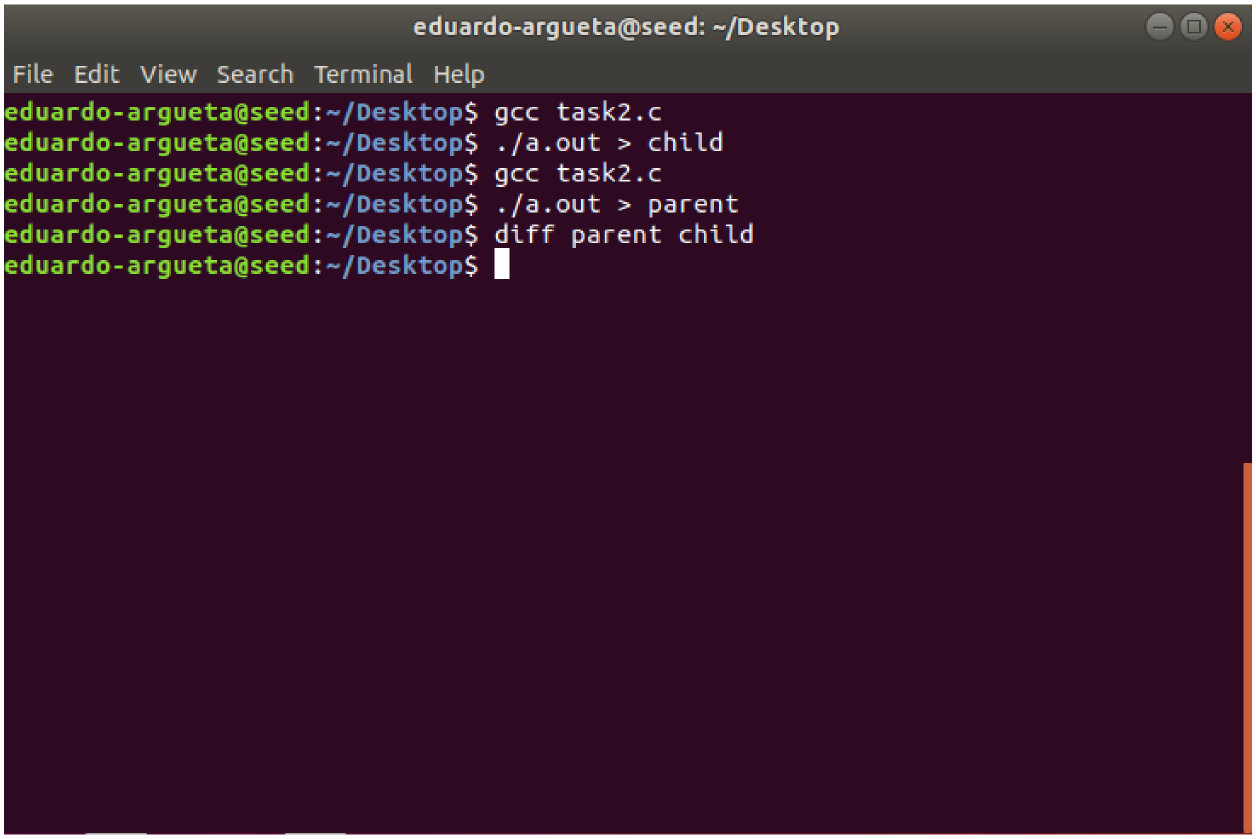


```
eduardo-argueta@seed:/home$ printenv USER
eduardo-argueta
eduardo-argueta@seed:/home$ env | grep PWD
PWD=/home
OLDPWD=/
eduardo-argueta@seed:/home$ export PWD=/home/eduardo-argueta
eduardo-argueta@seed:~$ printenv PWD
/home/eduardo-argueta
eduardo-argueta@seed:~$ unset PWD
eduardo-argueta@seed:/home$ printenv PWD
eduardo-argueta@seed:/home$ export PWD=/home/eduardo-argueta
eduardo-argueta@seed:~$ printenv PWD
/home/eduardo-argueta
eduardo-argueta@seed:~$ 
```

Observation: ***printenv*** command showed all the environment variables. As did the ***env*** command. The ***printenv PWD*** showed the value of a PWD, an environment variable. Then, ***env | grep PWD*** command will search PWD variable from the list of all the environment variables and then prints its value. Using ***export PWD=/home*** command, the value of PWD variable is being changed which can then be seen. It has also been observed that ***unset*** command basically removes that variable but one can again set its value. So ***unset PWD*** unsets the PWD. After unsetting, when its value is printed, it showed nothing exists.

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

In this task, we are going to analyze how a child process gets its environment variables from its parent. So, we would observe whether the parent's environment variables are inherited by the child process or not. This is done by the use of the ***fork()*** which is used to make a new process referred to as child process.


eduardo-argueta@seed: ~/Desktop
File Edit View Search Terminal Help

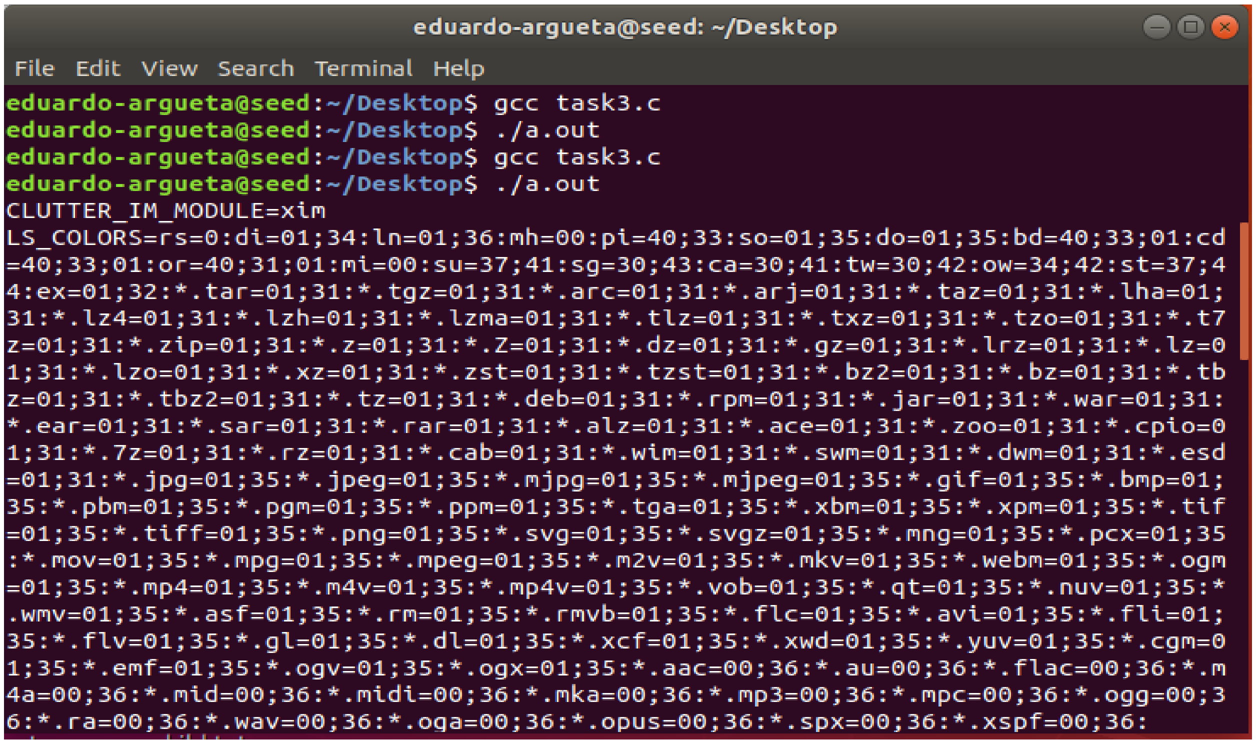
```
eduardo-argueta@seed:~/Desktop$ gcc task2.c
eduardo-argueta@seed:~/Desktop$ ./a.out > child
eduardo-argueta@seed:~/Desktop$ gcc task2.c
eduardo-argueta@seed:~/Desktop$ ./a.out > parent
eduardo-argueta@seed:~/Desktop$ diff parent child
eduardo-argueta@seed:~/Desktop$ 
```

Observations: First of all, all the environment variables were printed in the child process of ***task2.c*** file and stored in child file. Then, all the environment variables were printed in the parent process of ***task2.c*** file and stored in parent file. To observe, ***diff*** command is used to compare the environment variables in both the child and parent process. It was observed that there is no difference in the values of the environment variables of the child and process file. Both of them were exactly same.

Conclusion: It concludes that both child and parent process share the same environment variables. Hence, changing the value of environment variable in one process would also change to the other as well.

Task 3: Environment Variables and *execve()*

In this task, we would see how environment variables are affected when a new program is executed via ***execve()***. We would also observe whether the environment variables will automatically be inherited by the new program or we would have to pass them to the new program.


eduardo-argueta@seed: ~/Desktop
File Edit View Search Terminal Help

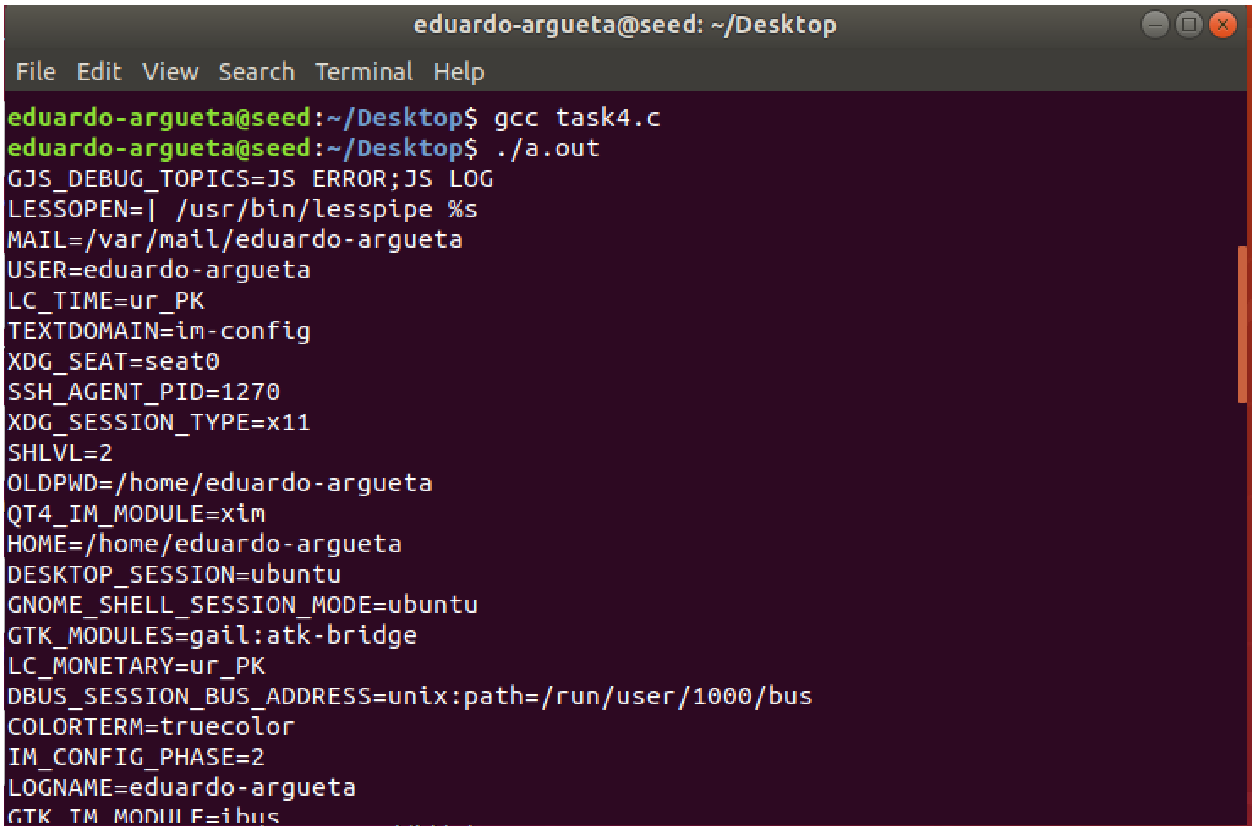
```
eduardo-argueta@seed:~/Desktop$ gcc task3.c
eduardo-argueta@seed:~/Desktop$ ./a.out
eduardo-argueta@seed:~/Desktop$ gcc task3.c
eduardo-argueta@seed:~/Desktop$ ./a.out
CLUTTER_IM_MODULE=xim
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=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:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=0
1;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tb
z=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=0
1;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd
=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=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:*.tiff=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=0
1;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m
4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;3
6:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
```

Observations: At first, the ***execve()*** is used but the environment or environment variables were not passed. It was supposed to print out all the environment variables but it didn't. Then, the ***execve()*** is used, now this time, it was passed with environment variables. When it's executed, it does print out all the environment variables.

Conclusion: It concludes that the new program doesn't automatically inherit the environment variables. We have to pass the environment variables if we are going to execute the new program through ***execve()***.

Task 4: Environment Variables and *system()*

In this task, we would see how environment variables are affected when a new program is executed via the ***system()*** function. This function doesn't directly execute the command, instead, the command is further executed exec* family functions which pass the environment variables when they execute ***execve()***. So, we don't have to pass the environment variables. To verify:

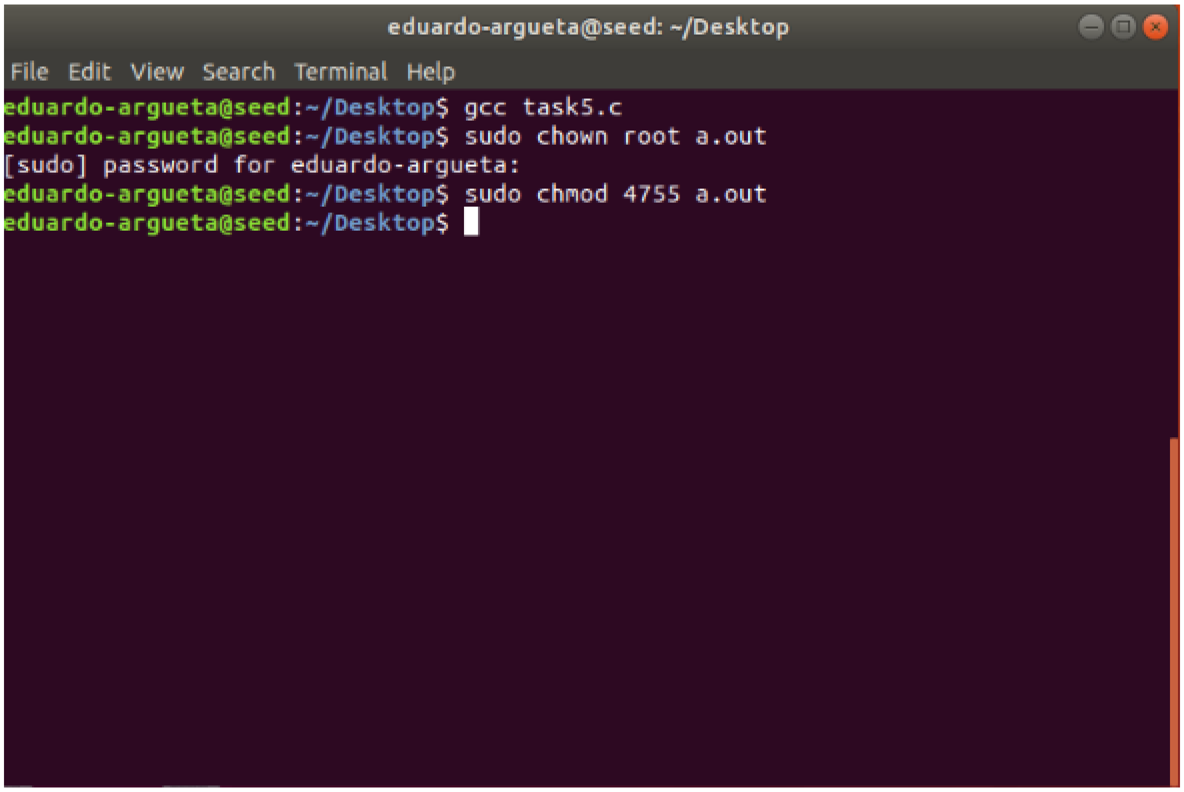

eduardo-argueta@seed: ~/Desktop

```
eduardo-argueta@seed:~/Desktop$ gcc task4.c
eduardo-argueta@seed:~/Desktop$ ./a.out
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
MAIL=/var/mail/eduardo-argueta
USER=eduardo-argueta
LC_TIME=ur_PK
TEXTDOMAIN=im-config
XDG_SEAT=seat0
SSH_AGENT_PID=1270
XDG_SESSION_TYPE=x11
SHLVL=2
OLDPWD=/home/eduardo-argueta
QT4_IM_MODULE=xim
HOME=/home/eduardo-argueta
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
LC_MONETARY=ur_PK
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
IM_CONFIG_PHASE=2
LOGNAME=eduardo-argueta
GTK_IM_MODULE=ibus
```

Verification: It has been verified when we execute the new program using ***system()*** function, we wouldn't have had to pass the environment variables. It still prints out all the environment variables.

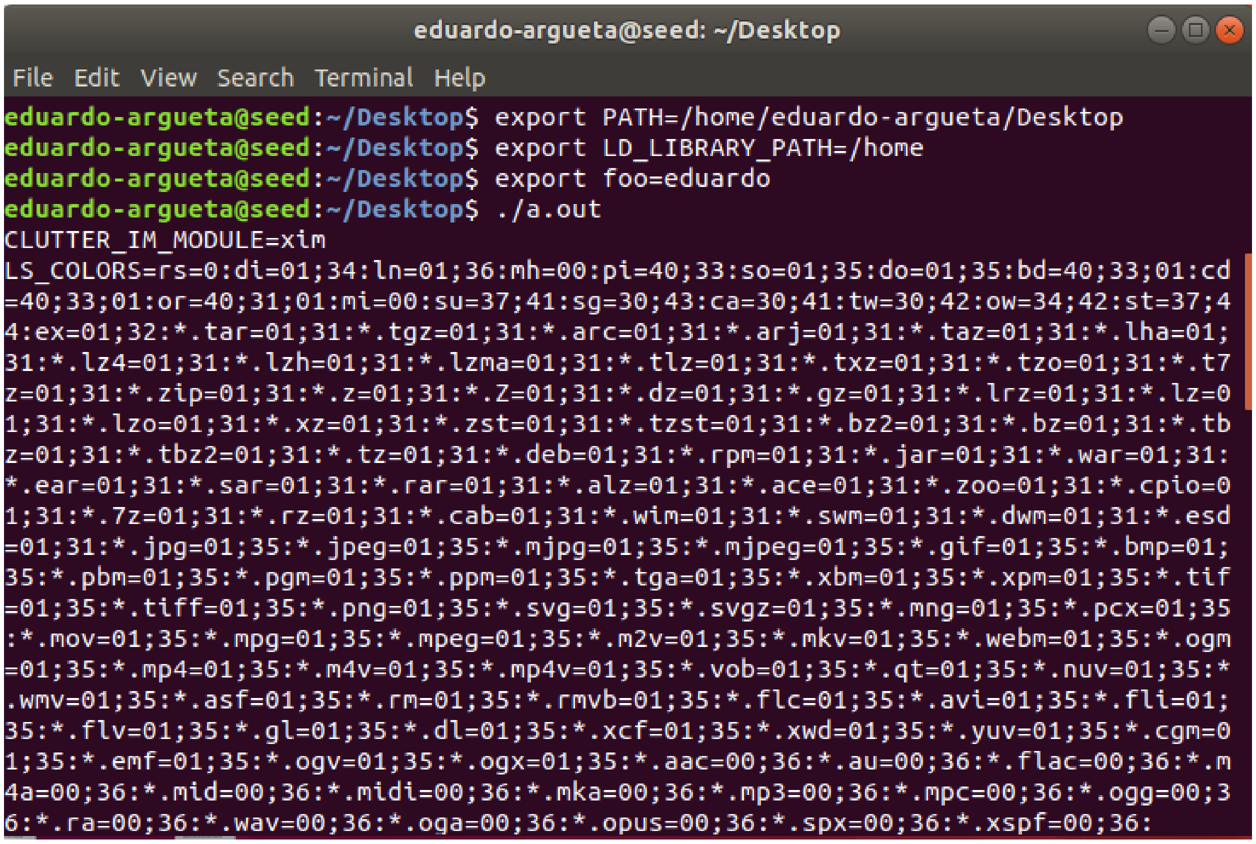
Task 5: Environment Variable and *Set-UID* Programs

In this task, we'll explore ***Set-UID*** programs. ***Set-UID*** is an important security mechanism in Unix operating systems. When these programs run, they assume the owner's privileges. We will see how users affect these programs via environment variables. First of all we would make our program to ***Set-UID*** program.


eduardo-argueta@seed: ~/Desktop
File Edit View Search Terminal Help

```
eduardo-argueta@seed:~/Desktop$ gcc task5.c
eduardo-argueta@seed:~/Desktop$ sudo chown root a.out
[sudo] password for eduardo-argueta:
eduardo-argueta@seed:~/Desktop$ sudo chmod 4755 a.out
eduardo-argueta@seed:~/Desktop$ 
```

Now, we have made our executable file (prints all the environment variables) as ***Set-UID*** program. Then just open as a normal user (not ***root***) and change the environment variables.


eduardo-argueta@seed: ~/Desktop
File Edit View Search Terminal Help

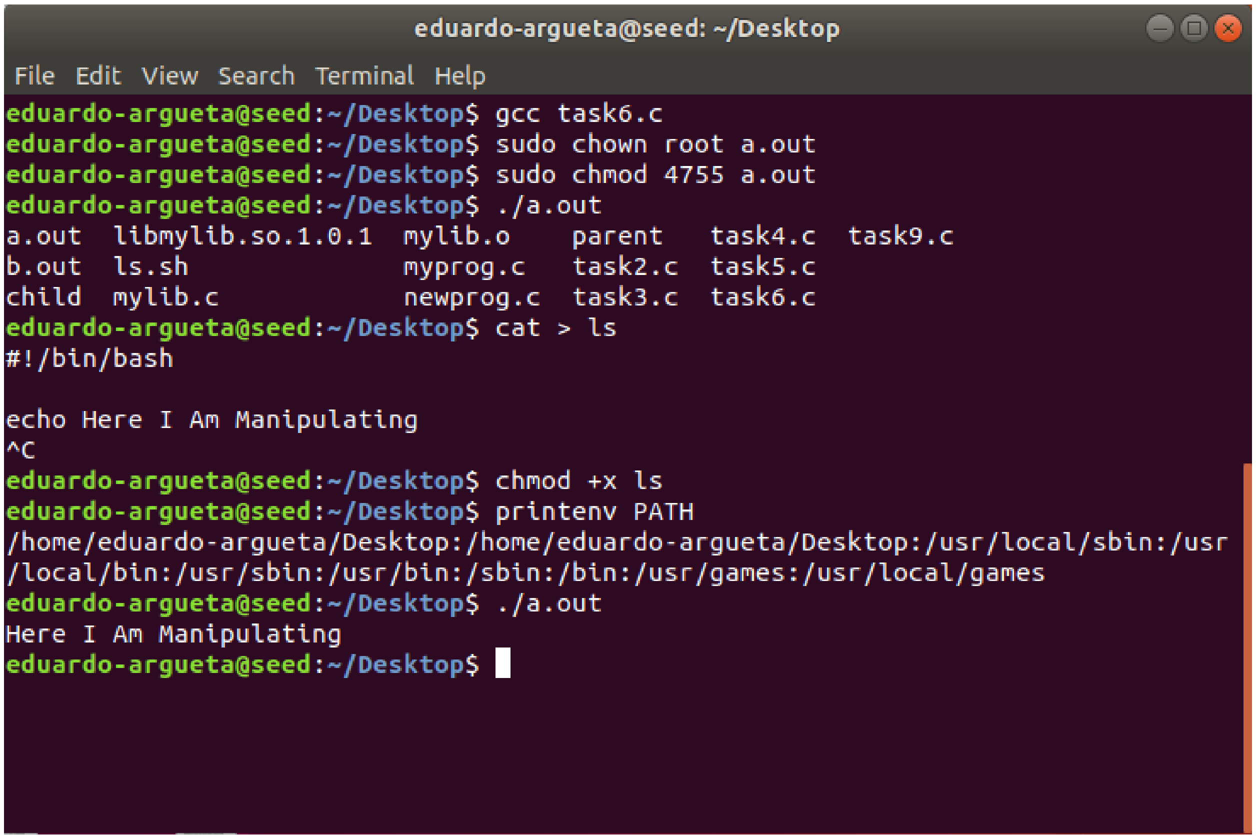
```
eduardo-argueta@seed:~/Desktop$ export PATH=/home/eduardo-argueta/Desktop
eduardo-argueta@seed:~/Desktop$ export LD_LIBRARY_PATH=/home
eduardo-argueta@seed:~/Desktop$ export foo=eduardo
eduardo-argueta@seed:~/Desktop$ ./a.out
CLUTTER_IM_MODULE=xim
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=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;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=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:*.t7
z=01;31:*.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=0
1;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzst=01;31:*.bz2=01;31:*.bz=01;31:*.tb
z=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=0
1;31:*.7z=01;31:*.rz=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd
=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=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:*.tiff=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=0
1;35:*.emf=01;35:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m
4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;3
6:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
```

Here, we have changed the ***PATH***, ***LD_LIBRARY_PATH*** and ***foo*** environment variable. And then executed that ***Set-UID*** program.

Observation: It has been observed that all the environment variables we set in the shell process (parent) does get into the ***Set-UID*** child process. It was surprising that even though we have changed the value of ***PATH*** and ***LD_LIBRARY_PATH***, still the ***Set-UID*** program executes (It must not as we changed the path).

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

In this task, we would see how the use of ***System()*** in a ***Set-UID*** program can be dangerous due to the fact that ***Set-UID*** programs are affected by environment variables and ***System()*** executes a command via shell.


eduardo-argueta@seed: ~/Desktop
File Edit View Search Terminal Help

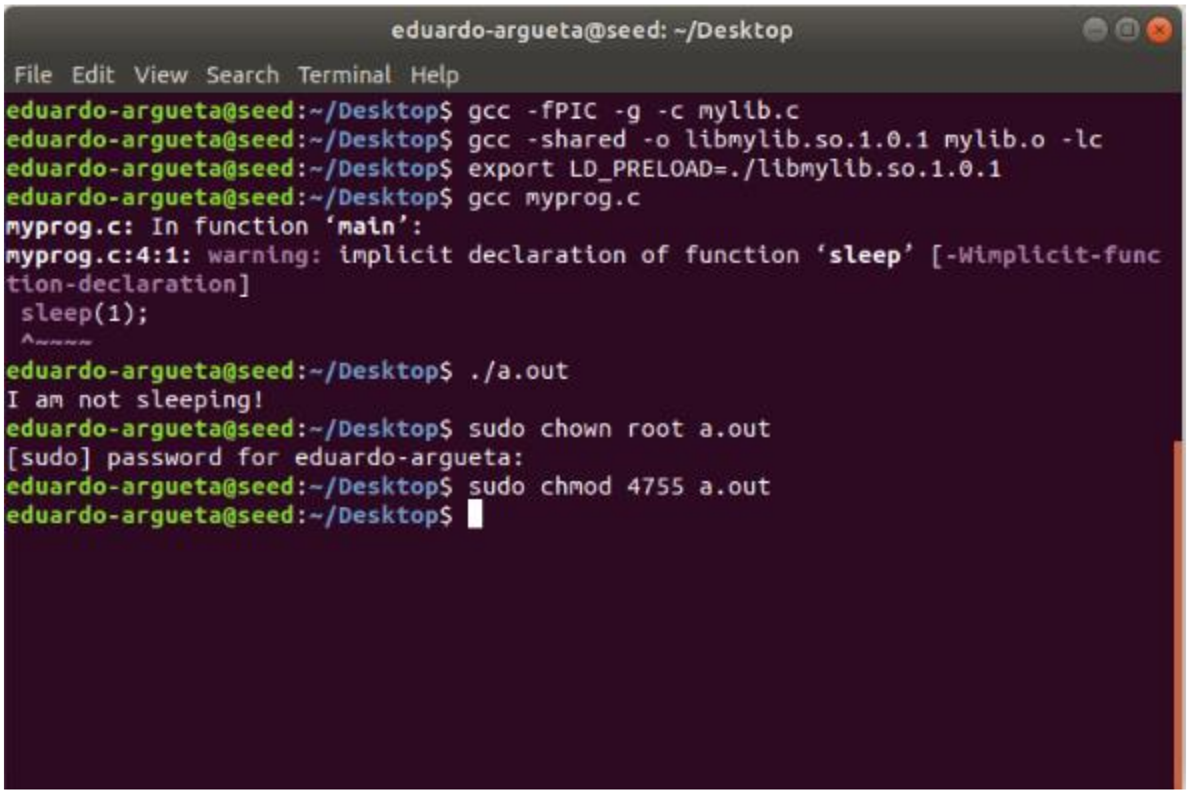
```
eduardo-argueta@seed:~/Desktop$ gcc task6.c
eduardo-argueta@seed:~/Desktop$ sudo chown root a.out
eduardo-argueta@seed:~/Desktop$ sudo chmod 4755 a.out
eduardo-argueta@seed:~/Desktop$ ./a.out
a.out  libmylib.so.1.0.1  mylib.o    parent   task4.c  task9.c
b.out  ls.sh              myprog.c   task2.c  task5.c
child  mylib.c            newprog.c  task3.c  task6.c
eduardo-argueta@seed:~/Desktop$ cat > ls
#!/bin/bash

echo Here I Am Manipulating
^C
eduardo-argueta@seed:~/Desktop$ chmod +x ls
eduardo-argueta@seed:~/Desktop$ printenv PATH
/home/eduardo-argueta/Desktop:/home/eduardo-argueta/Desktop:/usr/local/sbin:/usr
/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games
eduardo-argueta@seed:~/Desktop$ ./a.out
Here I Am Manipulating
eduardo-argueta@seed:~/Desktop$ 
```

Observations: Yes, we can let this ***Set-UID*** program to run our code instead of ***/bin/ls*** (it is supposed to run this). We would simple make a bash file and change the PATH environment variable and we know that it does affect the ***Set-UID*** programs which would let our program to run instead of ***/bin/ls***. We will keep the script (our code that we want to execute instead of ***/bin/ls***) named ***"ls"*** in a current directory and then append at the start our current directory. It means that it would start looking for ***ls*** firstly in our current directory and because it's a script, it would execute it. So, this is how we can manipulate the ***Set-UID*** programs by varying the ***PATH*** environment variable knowing that it would affect our ***Set-UID*** program using our own script.

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

In this task, we would analyze how ***Set-UID*** programs deal with several environment variables like ***LD_PRELOAD***, ***LD_LIBRARY_PATH*** and say ***LD_****. We would also analyze that how they influence the behavior of dynamic loader/linker. We are considering only ***LD_PRELOAD*** for this task.


eduardo-argueta@seed: ~/Desktop
File Edit View Search Terminal Help

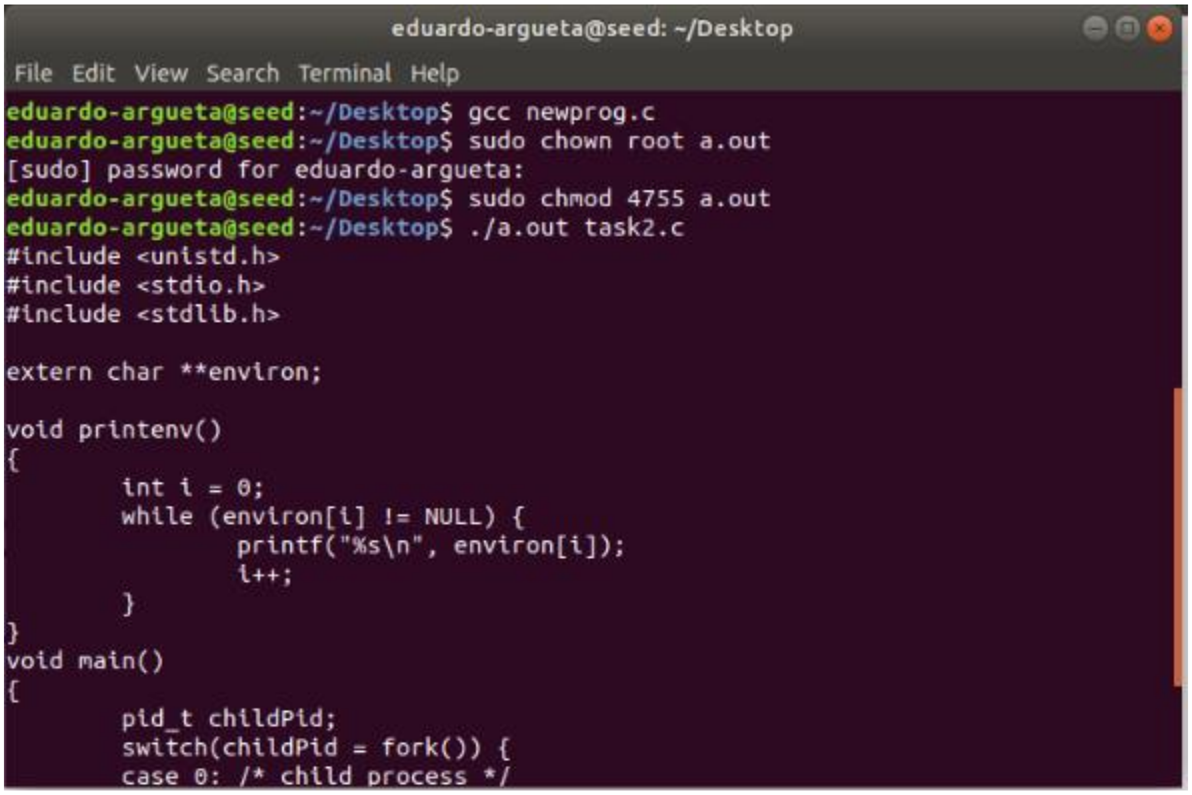
```
eduardo-argueta@seed:~/Desktop$ gcc -fPIC -g -c mylib.c
eduardo-argueta@seed:~/Desktop$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
eduardo-argueta@seed:~/Desktop$ export LD_PRELOAD=./libmylib.so.1.0.1
eduardo-argueta@seed:~/Desktop$ gcc myprog.c
myprog.c: In function ‘main’:
myprog.c:4:1: warning: implicit declaration of function ‘sleep’ [-Wimplicit-func
tion-declaration]
 sleep(1);
 ^~~~~
eduardo-argueta@seed:~/Desktop$ ./a.out
I am not sleeping!
eduardo-argueta@seed:~/Desktop$ sudo chown root a.out
[sudo] password for eduardo-argueta:
eduardo-argueta@seed:~/Desktop$ sudo chmod 4755 a.out
eduardo-argueta@seed:~/Desktop$ 
```

Observations: I observed that ***LD_PRELOAD***, ***LD_LIBRARY_PATH*** or other shared library variables do not apply to ***Set-UID*** programs/executables. When we made a regular ***myprog.c*** program and executes it, it runs fine and it does override the sleep function and print accordingly. But when we made it a ***Set-UID*** program, it never executes accordingly. The reason behind it might be that child process of a ***Set-UID*** program doesn't inherit ***LD_**** environment variables. That's why these environment variables doesn't apply on ***Set-UID*** programs.

Experiment: I did an experiment to see whether ***LD_**** environment variables are being inherited by child process or not. The hypothesis was true that ***LD_**** are not being inherited by the child process of ***Set-UID*** program.

Task 8: Invoking External Programs Using *System()* versus *execve()*

In this task, we will be analyzing the difference between ***System()*** and ***execve()***. We have seen that ***System()*** is quite dangerous as it executes the command through shell whereas on the other hand, in ***execve()***, we must pass the program that must be a binary executable or a script.



```
eduardo-argueta@seed:~/Desktop$ gcc newprog.c
eduardo-argueta@seed:~/Desktop$ sudo chown root a.out
[sudo] password for eduardo-argueta:
eduardo-argueta@seed:~/Desktop$ sudo chmod 4755 a.out
eduardo-argueta@seed:~/Desktop$ ./a.out task2.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void printenv()
{
        int i = 0;
        while (environ[i] != NULL) {
                printf("%s\n", environ[i]);
                i++;
        }
}
void main()
{
        pid_t childPid;
        switch(childPid = fork()) {
        case 0: /* child process */
```

Observations: If I have used ***System()***, then as a Bob, I can remove the file that is not writable to me. I can do manipulation by adding a script in which I can easily remove that file or manipulate it because ***System()*** invokes through shell. Hence, I can attack if ***System()*** is being used. On Contrary, my attacks won't work if ***execve()*** is being used because we must pass a program which is a binary executable or a script. Therefore, ***execve()*** must be used to fully protect the integrity of the system.