

Information Security: Environment Variable and Set-UID Program

Name : Chandan N Bhat

PES1201701593

Section H

Task 1: Manipulating environment variables

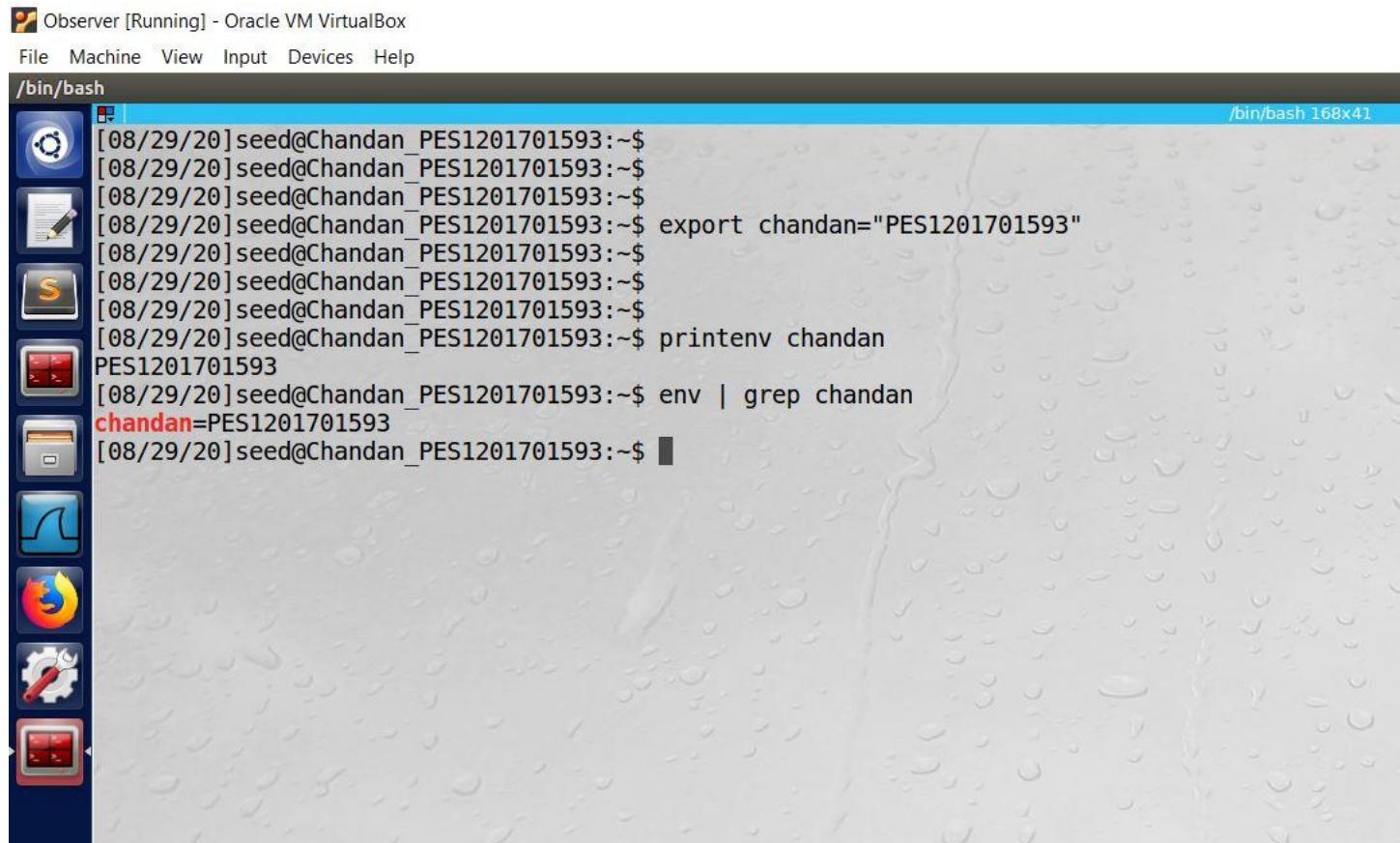
Here we use few Linux commands to set and unset environment variables. We use the default shell, Bash. The “printenv” or “env” command can be used to print out the environment variables

```
[08/29/20]seed@Chandan_PES1201701593:~$ printenv
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=1
TERMINATOR_UUID=urn:uuid:617a902f-c73e-4a0f-83ad-a388f22ffab7
IBUS_DISABLE_SNOOPER=1
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
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=25165828
UPSTART SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1168
GNOME_KEYRING_CONTROL=
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:*.arc=01;31:*.arj=01;31:*.tarz=01;31:*.lha=01;31:*.lza=01;31:*.lz=01;31:*.lzma=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.tar=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.xz=01;31:*.bz=01;31:*.bz2=01;31:*.tbz=01;31:*.tbz2=01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.tar=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:*.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:*.ogg=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:*.fl1=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xct=01;35:*.xwd=01;35:*.yuv=01;35:*.gm=01;35:*.emf=01;35:*.obj=01;35:*.ogv=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.mid=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:*
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_IM_MODULE=ibus
```

Including the key i.e., “pwd” for instance, will print the particular environment variable.

```
[08/29/20]seed@Chandan_PES1201701593:~$ [08/29/20]seed@Chandan_PES1201701593:~$ printenv PWD
/home/seed
[08/29/20]seed@Chandan_PES1201701593:~$ env | grep PWD
PWD=/home/seed
[08/29/20]seed@Chandan_PES1201701593:~$
```

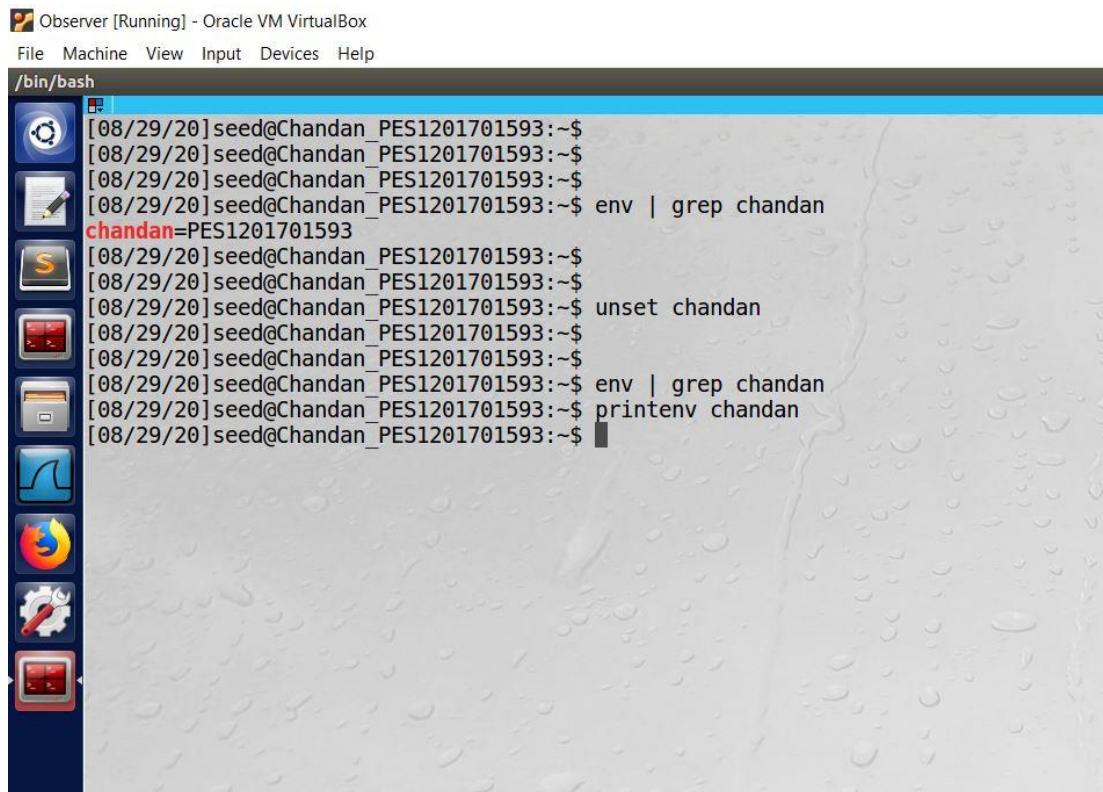
Next using the “export” command we set an environment variable as shown in the below screenshot, where we set an environment variable “chandan=PES1201701593”.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "/bin/bash" and the status bar indicates it's running at 168x41 resolution. The terminal content shows the following session:

```
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$ export chandan="PES1201701593"  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$ printenv chandan  
PES1201701593  
[08/29/20]seed@Chandan_PES1201701593:~$ env | grep chandan  
chandan=PES1201701593  
[08/29/20]seed@Chandan_PES1201701593:~$
```

We can unset an existing environment variable using the unset command as shown below



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is "/bin/bash" and the status bar indicates it's running at 168x41 resolution. The terminal content shows the following session:

```
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$ env | grep chandan  
chandan=PES1201701593  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$ unset chandan  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$  
[08/29/20]seed@Chandan_PES1201701593:~$ env | grep chandan  
[08/29/20]seed@Chandan_PES1201701593:~$ printenv chandan  
[08/29/20]seed@Chandan_PES1201701593:~$
```

We see that once we unset the environment variable, and try to print it, nothing is displayed in the output as we have unset the environment variable.

Thus, we successfully set and unset environment variables using Bash's internal commands "export" and "unset". We also print the environment variables using "printenv" command.

Task 2: Inheriting environment variables from parents

Now we will understand how environment variables are inherited by child processes from the parent processes. The fork() system call duplicates the calling process which is called the child process. During this process the child process inherits various attributes from its parent such as file descriptors. We will check if environment variables are inherited by the child process in this task.

1. We first run a simple C program that prints the environment variables of the process.



```
/*penv program */
#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 */
            printenv();
            exit(0);
        default: /* parent process */
            //printenv();
            exit(0);
    }
}
```

2. We first print all the environment variables of the child process created by fork() in a file ,say child.txt.

Observer [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

/bin/bash

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$ gcc penv.c

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$./a.out > child.txt

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$ ls -l child.txt

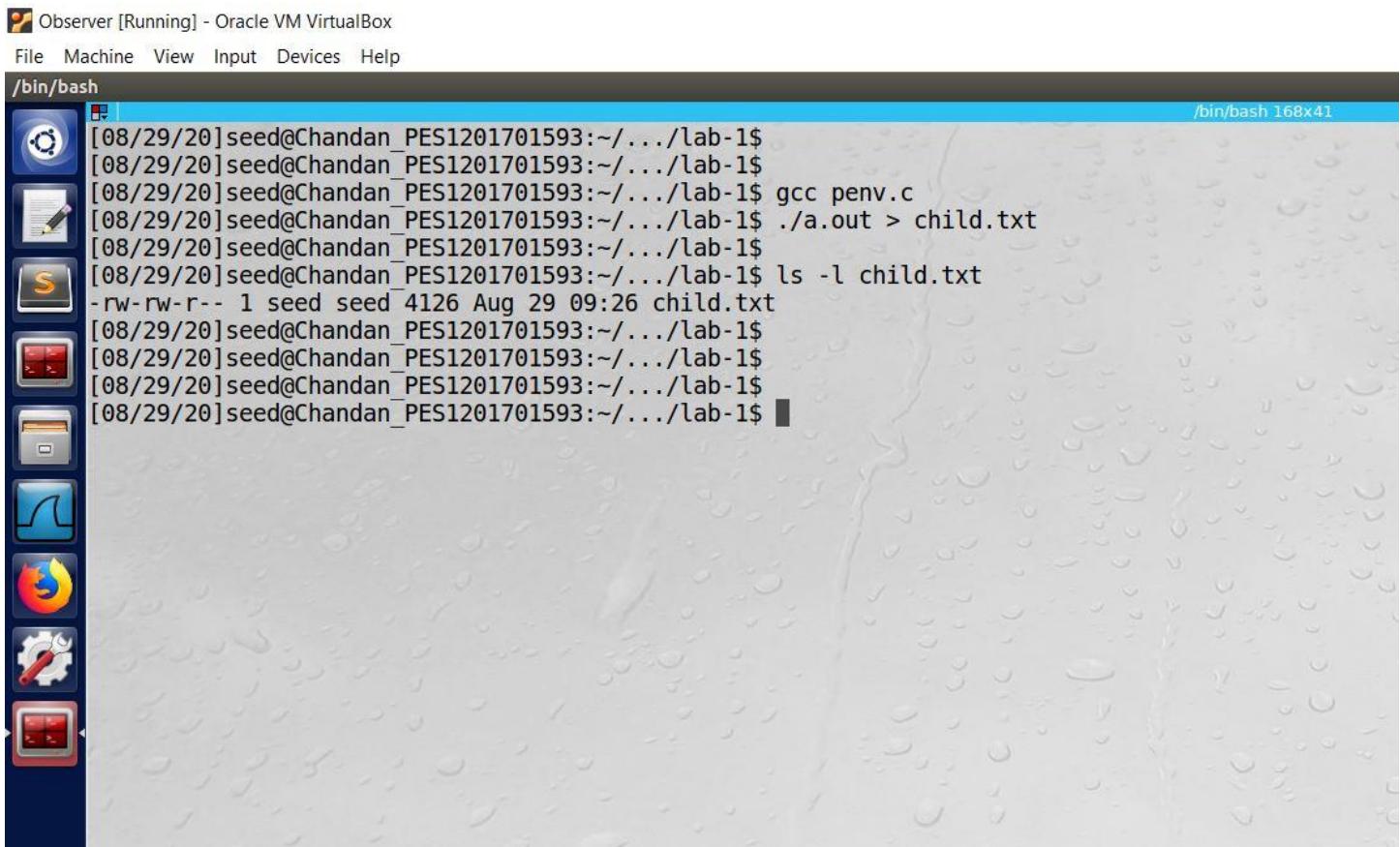
-rw-rw-r-- 1 seed seed 4126 Aug 29 09:26 child.txt

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$

[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1\$



3. Then we print all the environment variables of the parent process in another file, say parent.txt. We make a minor change in the same code, as shown below.

Observer [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

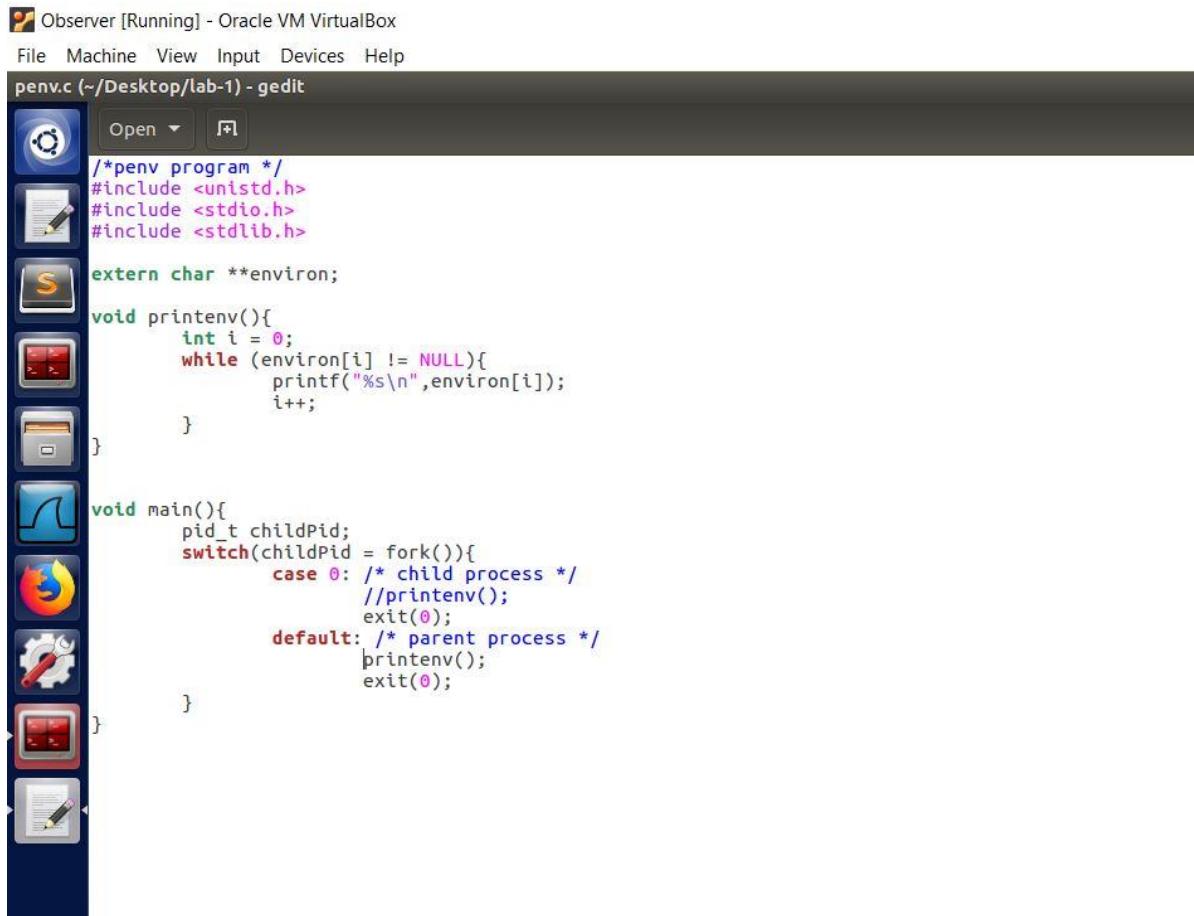
penv.c (~/Desktop/lab-1) - gedit

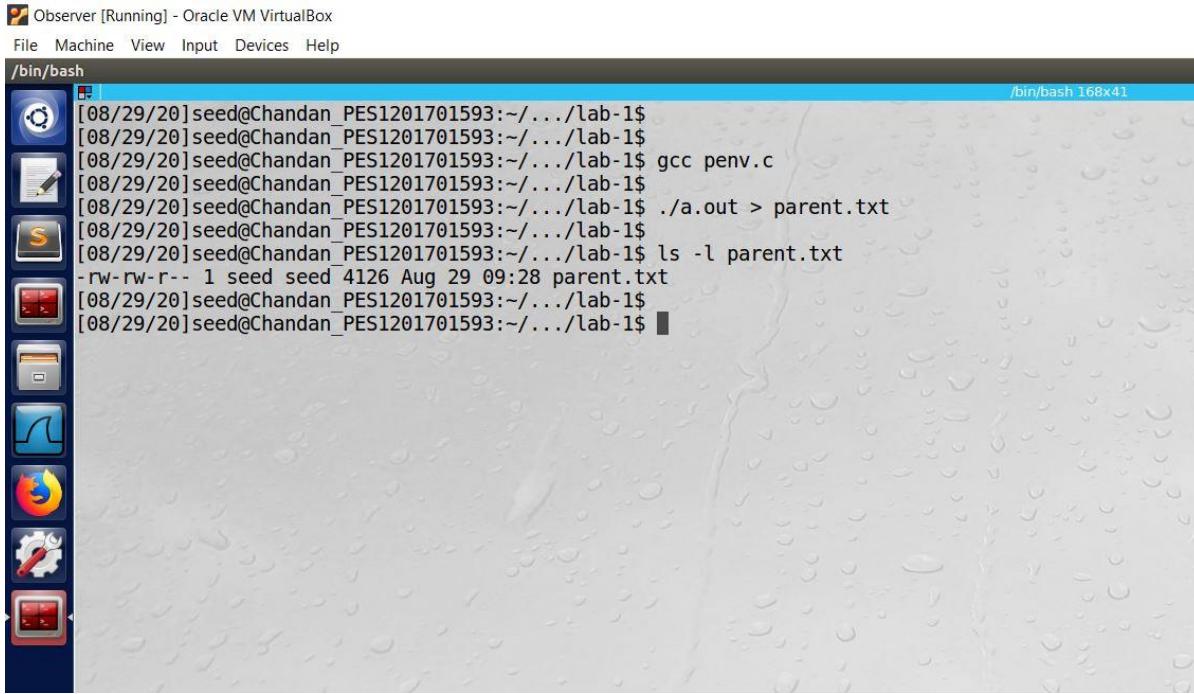
/*penv program */
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

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 */
 //printenv();
 exit(0);
 default: /* parent process */
 printenv();
 exit(0);
 }
}

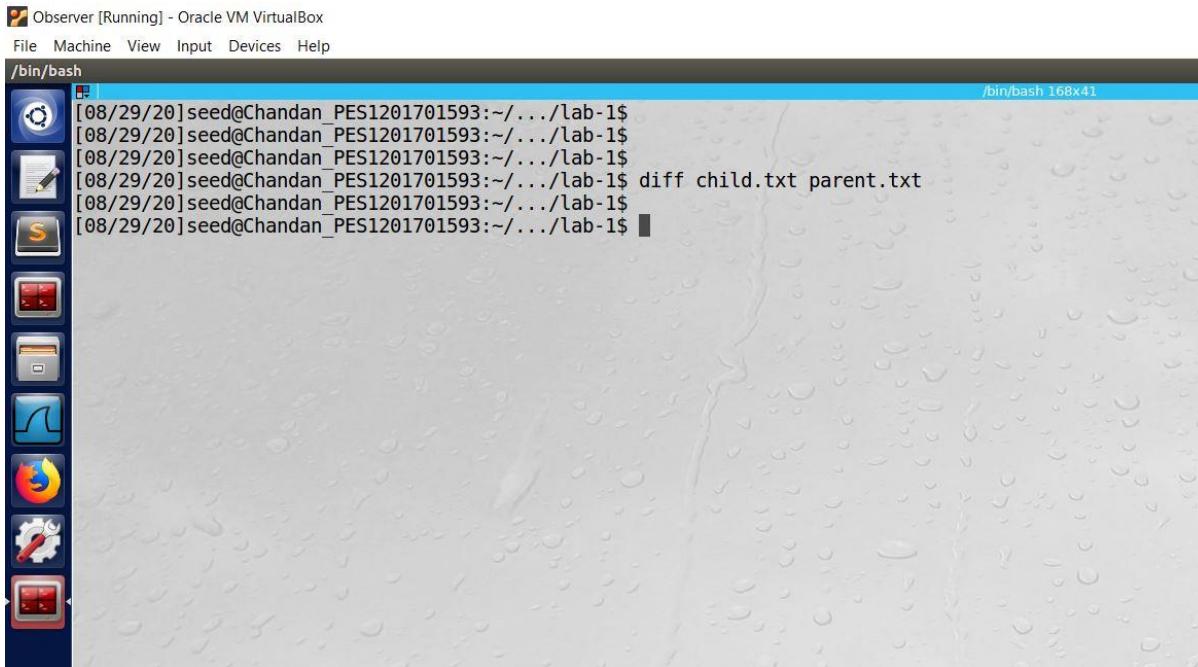




The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is '/bin/bash' and the size is '168x41'. The terminal content shows the following command-line session:

```
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$ gcc penv.c  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$ ./a.out > parent.txt  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$ ls -l parent.txt  
-rw-r--r-- 1 seed seed 4126 Aug 29 09:28 parent.txt  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$
```

4. Then using the “diff” command we compare the differences between the two files.



The screenshot shows a Linux desktop environment with a terminal window open. The terminal window title is '/bin/bash' and the size is '168x41'. The terminal content shows the following command-line session:

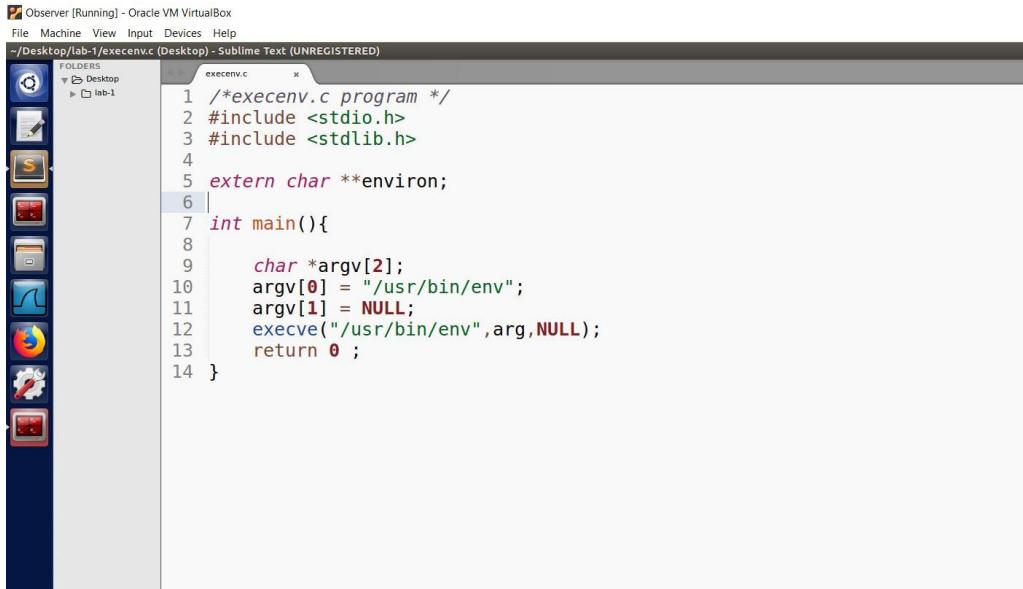
```
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$ diff child.txt parent.txt  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$  
[08/29/20]seed@Chandan_PES1201701593:~/.../lab-1$
```

Thus we observe that the diff command output is empty indicating that there is no difference between the files child.txt and parent.txt which contain the environment variables of the child and parent process. Thus, we can conclude that the child process inherits the environment variables from its parent process. But additionally, a parent process may use CreateProcess() to create a child process and pass a new set of environment variables to it.

Task 3: Environment variables and execve()

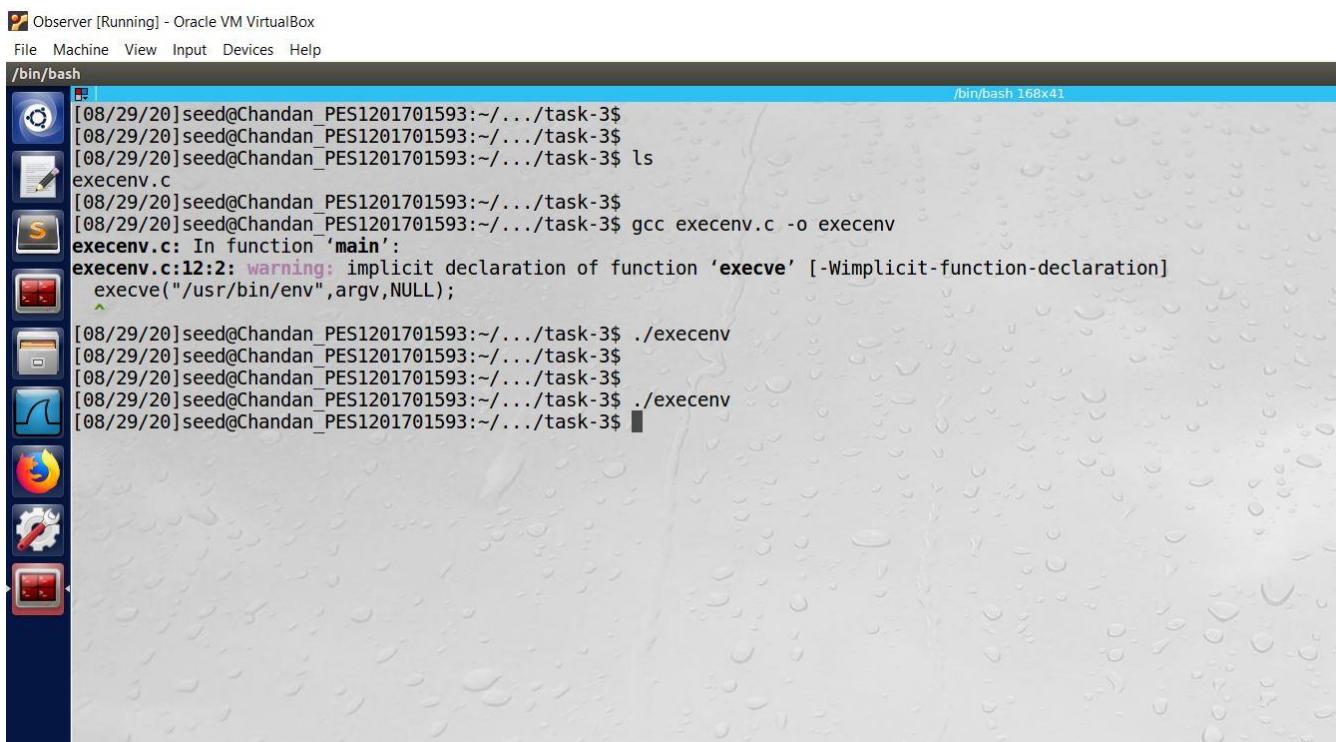
Now we will understand how environment variables are affected when a new program is executed via `execve()`. The function `execve()` calls a system call to load a new command and execute it. No new process is created but the calling process's attributes are overwritten by that of the program overloaded. We will observe if the environment variables are inherited by the new program.

1. We will run a C program which executes the program '`/usr/bin/env`' using `execve()`.



The screenshot shows a Sublime Text window titled 'execenv.c' with the following code:

```
1 /*execenv.c program */
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 extern char **environ;
6
7 int main(){
8
9     char *argv[2];
10    argv[0] = "/usr/bin/env";
11    argv[1] = NULL;
12    execve("/usr/bin/env", argv, NULL);
13    return 0 ;
14 }
```



The screenshot shows a terminal window titled '/bin/bash' with the following session log:

```
[08/29/20]seed@Chandan_PES1201701593:~/....task-3$ [08/29/20]seed@Chandan_PES1201701593:~/....task-3$ [08/29/20]seed@Chandan_PES1201701593:~/....task-3$ ls execenv.c [08/29/20]seed@Chandan_PES1201701593:~/....task-3$ [08/29/20]seed@Chandan_PES1201701593:~/....task-3$ gcc execenv.c -o execenv execenv.c: In function 'main': execenv.c:12:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration] execve("/usr/bin/env", argv, NULL); [08/29/20]seed@Chandan_PES1201701593:~/....task-3$ ./execenv [08/29/20]seed@Chandan_PES1201701593:~/....task-3$ [08/29/20]seed@Chandan_PES1201701593:~/....task-3$ [08/29/20]seed@Chandan_PES1201701593:~/....task-3$ ./execenv [08/29/20]seed@Chandan_PES1201701593:~/....task-3$
```

We will observe that there is no output printed in the standard output, thus we can infer that no environment variables are inherited by the new program when executed via `execve()` sys call.

Next, we tweak the program a bit and observe the output again.

2. Next we change the invocation of execve() to – execve("/usr/bin/env", argv , environ).



File Machine View Input Devices Help

-/Desktop/lab-1/task-3/execenv.c (Desktop) - Sublime Text (UNREGISTERED)

FOLDERS

- Desktop
- lab-1
 - task-2
 - task-3
 - execenv
 - execenv.c
 - execenv2.c

execenv2.c x execenv.c x

```
1 /*execenv.c program */
2 #include <stdio.h>
3 #include <stdlib.h>
4
5 extern char **environ;
6
7 int main(){
8
9     char *argv[2];
10    argv[0] = "/usr/bin/env";
11    argv[1] = NULL;
12    execve("/usr/bin/env", argv, environ);
13    return 0;
14 }
15
```

Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help

/bin/bash

[08/29/20]seed@Chandan_PES1201701593:~/.../task-3\$
[08/29/20]seed@Chandan_PES1201701593:~/.../task-3\$
[08/29/20]seed@Chandan_PES1201701593:~/.../task-3\$ gcc execenv.c -o execenv
execenv.c: In function 'main':
execenv.c:12:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
execve("/usr/bin/env",argv,environ);

[08/29/20]seed@Chandan_PES1201701593:~/.../task-3\$./execenv
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-bitseed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
TERMINATOR_UUID=urn:uuid:617a902f-c73e-4a0f-83ad-a388f22ffab7
IBUS_DISABLE_SNOOPER=1
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
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=25165828
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1168
GNOME_KEYRING_CONTROL=
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;*:arc=01;31;*:arj=01;31;*:tarz=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;*:tar=01;31;*:tar=01;31;*:gz=01;31;*:lz=01;31;*:lrz=01;31;*:lzma=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;*:ppm=01;35;*:tga=01;35;*:xbm=01;35;*:xpm=01;35;*:tif=01;35;*:tiff=01;35;*:png=01;35;*:svg=01;35;*:svg=01;35;*:svgz=01;35;*:mng=01;35;*:pxc=01;35;*:mov=01;35;*:mpg=01;35;*:mpeg=01;35;*:m2v=01;35;*:mkv=01;35;*:webm=01;35;*:ogg=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;*:cm=01;35;*:emf=01;35;*:ogv=01;35;*:ogx=01;35;*:aac=00;36;*:au=00;36;*:flac=00;36;*:m4a=00;36;*:mid=00;36;*:mka=00;36;*:mp3=00;36;*:mpc=00;36;*:ogg=00;36;*:oga=00;36;*:opus=00;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

The above screenshot shows that on running the program with the above-mentioned changes, it prints a list of environment variables. Thus, we can observe that environment variables are inherited by the new program from the calling process when the `execve()` system call is invoked with the 3rd argument as “environ” instead of “NULL”.

Reading from the documentation of `execve()` it takes the new environment of the process as an argument. Thus passing “NULL” as the argument creates the new process with no environment variables while passing “`environ`” as the argument creates a new process with user environment.

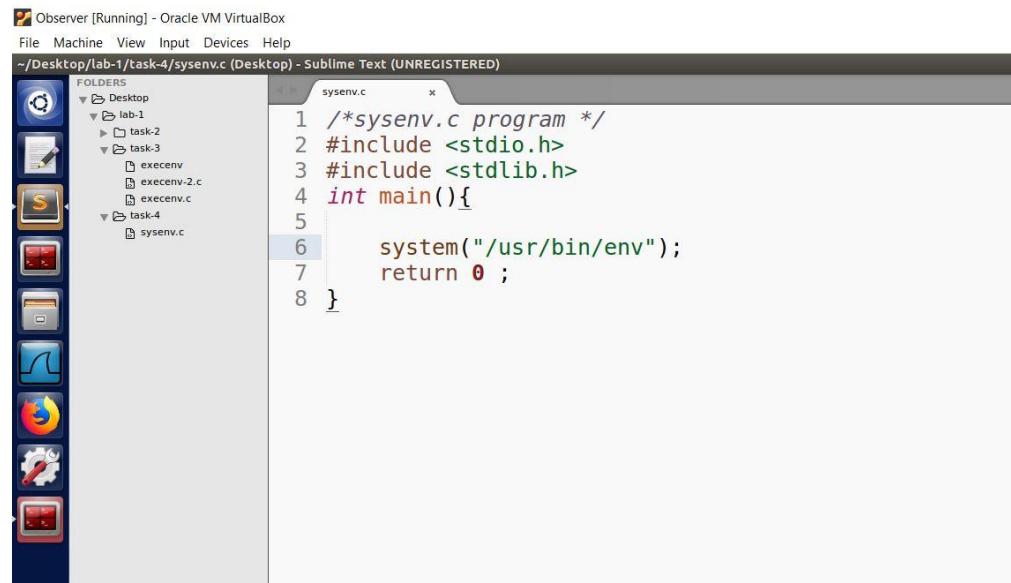
Task 4: Environment variables and system

Now we will understand how environment variables are affected when a new program is executed via `system()`. This is similar to `execve()` except the fact that `system()` actually executes "`/bin/sh -c command`" and asks the shell to execute the command.

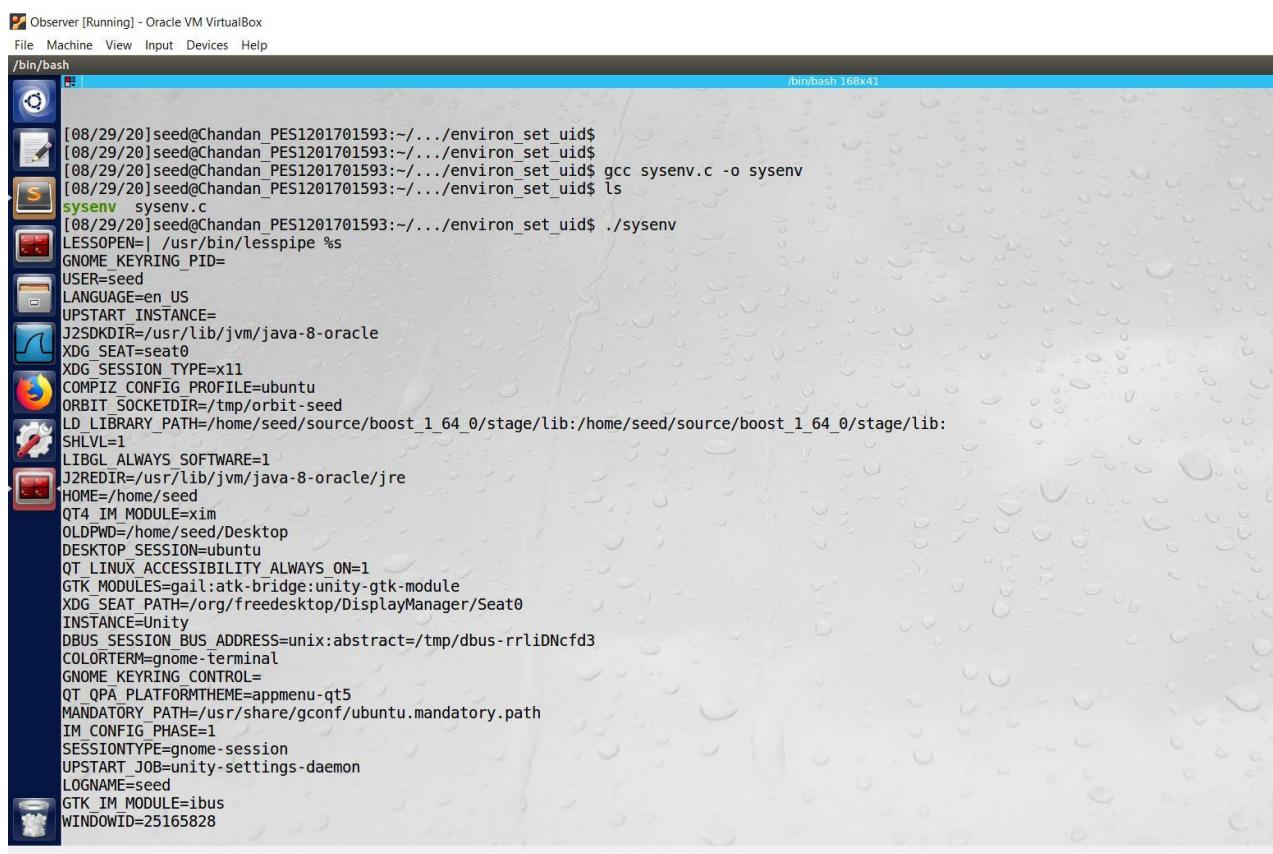
`system() = execel() + execve()`

Therefore using `system()`, the environment variables of the calling process is passed to the new program `/bin/sh`

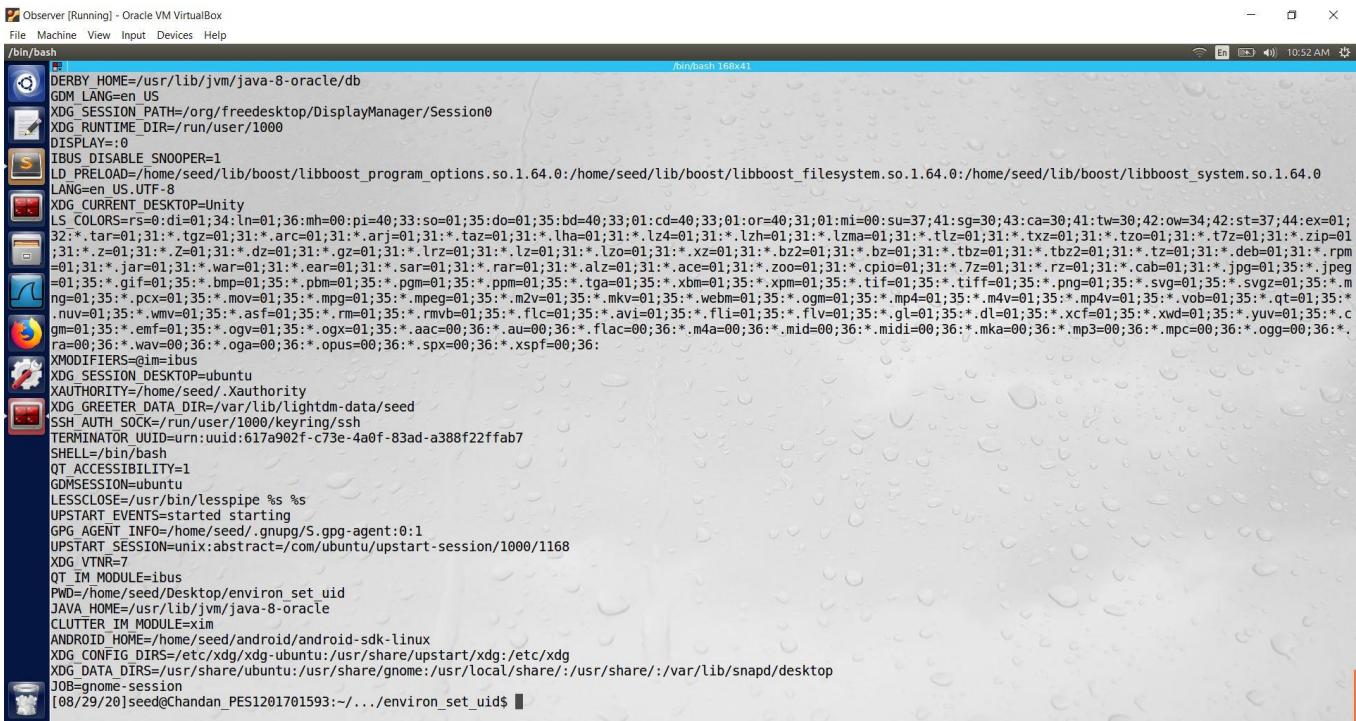
1. We first run a C program that uses the `system()` call that runs "`/usr/bin/env`".



```
Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
~/Desktop/lab-1/task-4/sysenv.c (Desktop) - Sublime Text (UNREGISTERED)
FOLDERS
  Desktop
    lab-1
      task-2
      task-3
        execenv
          execenv-2.c
          execenv.c
      task-4
        sysenv.c
  sysenv.c
1 /*sysenv.c program */
2 #include <stdio.h>
3 #include <stdlib.h>
4 int main(){
5
6     system("/usr/bin/env");
7     return 0 ;
8 }
```



```
[08/29/20]seed@Chandan_PES1201701593:~/.environs_set_uid$ [08/29/20]seed@Chandan_PES1201701593:~/.environs_set_uid$ [08/29/20]seed@Chandan_PES1201701593:~/.environs_set_uid$ gcc sysenv.c -o sysenv [08/29/20]seed@Chandan_PES1201701593:~/.environs_set_uid$ ls sysenv [08/29/20]seed@Chandan_PES1201701593:~/.environs_set_uid$ ./sysenv LESSOPEN=-| /usr/bin/lesspipe % GNOME_KEYRING_PID= USER=seed LANGUAGE=en_US UPSTART_INSTANCE= J2SDKDIR=/usr/lib/jvm/java-8-oracle XDG_SEAT=seat0 XDG_SESSION_TYPE=x11 COMPIZ_CONFIG_PROFILE=ubuntu ORBIT_SOCKETDIR=/tmp/orbit-seat LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib: SHLVL=1 LIBGL_ALWAYS_SOFTWARE=1 J2REDIR=/usr/lib/jvm/java-8-oracle/jre HOME=/home/seed QT4_IM_MODULE=xim OLDPWD=/home/seed/Desktop DESKTOP_SESSION=ubuntu QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1 GTK_MODULES=gail:atk-bridge:unity-gtk-module XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0 INSTANCE=Unity DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-rrliDNCfd3 COLORTERM=gnome-terminal GNOME_KEYRING_CONTROL= QT_QPA_PLATFORMTHEME=appmenu-qt5 MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path IM_CONFIG_PHASE=1 SESSIONTYPE=gnome-session UPSTART_JOB=unity-settings-daemon LOGNAME=seed GTK_IM_MODULE=ibus WINDOWID=25165828
```



```

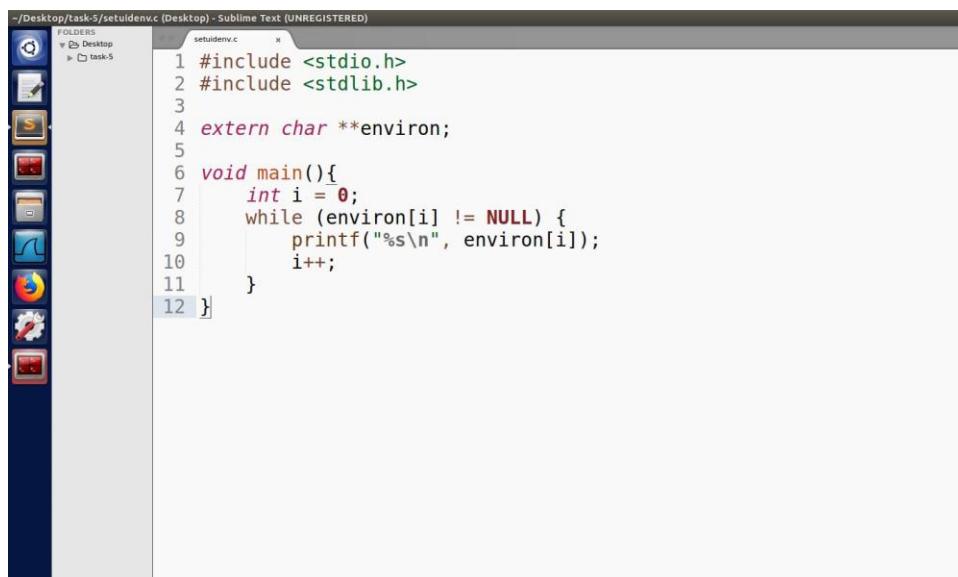
Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
GDM_LANG=en_US
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_RUNTIME_DIR=/run/user/1000
DISPLAY=:0
IBUS_DISABLE_SNOOPER=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0
LANG=en_US.UTF-8
XDG_CURRENT_DESKTOP=Unity
LS_COLORS=rs=0;di=0;34;ln=0;36;mh=0;pi=40;33;so=0;35;do=0;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;*.arc=01;31;*.arj=01;31;*.tar.zst=01;31;*.lzh=01;31;*.lz4=01;31;*.lzma=01;31;*.tlz=01;31;*.txz=01;31;*.tzo=01;31;*.t7z=01;31;*.zip=01;31;*.tar.zst=01;31;*.xz=01;31;*.xz=01;31;*.bz2=01;31;*.bz=01;31;*.tbz2=01;31;*.tbz=01;31;*.t2z=01;31;*.deb=01;31;*.rpm=01;31;*.jar=01;31;*.war=01;31;*.ear=01;31;*.sar=01;31;*.rar=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;*.ppm=01;35;*.tga=01;35;*.xbm=01;35;*.xpm=01;35;*.tif=01;35;*.tiff=01;35;*.png=01;35;*.svg=01;35;*.svga=01;35;*.mng=01;35;*.pcx=01;35;*.mov=01;35;*.mpeg=01;35;*.m2v=01;35;*.mkv=01;35;*.webm=01;35;*.ogg=01;35;*.mp4=01;35;*.m4v=01;35;*.m4a=00;36;*.mid=00;36;*.midi=00;36;*.mp3=00;36;*.ogg=00;36;*.ra=00;36;*.wav=00;36;*.oga=00;36;*.opus=00;36;*.spx=00;36;*.xspf=00;36;
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
TERMINATOR_UUID=urn:uuid:617a902f-c73e-4a0f-83ad-a388f22ffab7
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
UPSTART_EVENTS=started starting
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract:/com/ubuntu/upstart-session/1000/1168
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed/Desktop/.environ_set_uid
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
JOB=0/home-session
[08/29/20]seed@Chandan_PES1201701593:~/.environ_set_uid$ 

```

Task 5: Environment variables and Set UID programs

Here we will explore Set-UID programs which are security mechanism in linux. A Set-UID program assumes root privileges when run, the owner of the program is root. This escalation of the privileges makes it risky, and its behaviour can be affected by environment variables. We will observe if environment variables are inherited by Set-UID program's processes from user's process.

1. We write a program that prints all the environment variables of the current process



```

~/Desktop/task-5/setuidenv.c (Desktop) - Sublime Text (UNREGISTERED)
FOLDERS
  v Ds Desktop
  > task-5
  setuidenv.c
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 extern char **environ;
5
6 void main(){
7     int i = 0;
8     while (environ[i] != NULL) {
9         printf("%s\n", environ[i]);
10        i++;
11    }
12 }

```

2. We compile the above program, change its ownership to root and make it a Set-UID program

A screenshot of a Linux desktop environment within a VirtualBox window. The desktop has a light blue background with a water droplet pattern. A terminal window titled '/bin/bash' is open at the bottom left, showing a shell session:

```
[08/29/28]seed@Chandan_PES1201701593:~/.../task-5$  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ ls  
setuidenv.c  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ gcc setuidenv.c -o setuidenv  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ sudo chmod 4755 setuidenv  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ ./setuidenv  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ ls  
rwsr-xr-x 1 root seed 7400 Sep 1 09:59 setuidenv  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$
```

A screenshot of a Linux desktop environment within a VirtualBox window. The desktop has a light blue background with a water droplet pattern. A terminal window titled '/bin/bash' is open at the bottom left, showing a shell session:

```
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ gcc setuidenv.c -o setuidenv  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ sudo chmod 5744 setuidenv  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ ./setuidenv  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$ ls  
rwsr-r-T 1 seed seed 7400 Sep 1 10:07 setuidenv  
[09/01/28]seed@Chandan_PES1201701593:~/.../task-5$
```

3. In the bash shell, from a non-root account we set the below environment variables using export command and direct its environment variables to env_result file. These environment variables are set in the user's shell process.

```

d-ndk/android-ndk-r8d:/home/seed/.local/bin
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ /home/seed/source/boost_1_64_0/stage/lib;/home/seed/source/boost_1_64_0/stage/lib;
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ export LD_LIBRARY_PATH=/home/seed/:$LD_LIBRARY_PATH
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ /home/seed/source/boost_1_64_0/stage/lib;/home/seed/source/boost_1_64_0/stage/lib;
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ printenv LD_LIBRARY_PATH
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ /home/seed/source/boost_1_64_0/stage/lib;/home/seed/source/boost_1_64_0/stage/lib;
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ printenv chandan
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ export chandan='PES1201701593'
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ printenv chandan
PES1201701593
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ env > env_result
[09/01/20]seed@Chandan_PES1201701593:~/Desktop$ 

```

4. We run the program **setuidenv**, which is a Set-UID program and direct its output(env vars of current process) to another file **setuid_result** file. The shell forks a child process, which further runs the program.

```

chan@Chandan_PES1201701593: ~
/bin/bash
[09/01/20]seed@Chandan_PES1201701593:~/.../task-5$
[09/01/20]seed@Chandan_PES1201701593:~/.../task-5$
[09/01/20]seed@Chandan_PES1201701593:~/.../task-5$
[09/01/20]seed@Chandan_PES1201701593:~/.../task-5$ ./setuidenv > setuid_result
[09/01/20]seed@Chandan_PES1201701593:~/.../task-5$ ls
setuidenv setuidenv.c setuid_result
[09/01/20]seed@Chandan_PES1201701593:~/.../task-5$ 

```

5. Using the **diff** command we can compare both the **env_result** and **setuid_result** to check whether all the environment variables set in parent(shell process) are inherited by the Set-UID child process. We can conclude that the SET-UID program's child process may not inherit all the environment variables of the parent process, **LD_LIBRARY_PATH** being one of them. We have seen why this happens in task 7 of this lab.

```
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$  
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$  
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$  
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$ env > env_result  
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$ ls  
env_result setuidenv setuidenv.c setuid result  
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$  
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$  
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$ diff setuid_result env_result  
5c5  
< TERMINATOR_UUID=urn:uuid:4c339312-4655-43fc-be3a-82a190660ca6  
-->  
> TERMINATOR_UUID=urn:uuid:befef0b1-a442-4dbd-ba24-494aba0ac421  
15c15,16  
< WINDOWID=75497476  
-->  
> WINDOWID=69206020  
> OLDPWD=/home/seed/Desktop  
22c23  
< LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:  
-->  
> LD_LIBRARY_PATH=/home/seed/:/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:  
31a33  
> chandan=PES1201701593  
71,72c73  
<_=./setuidenv  
< OLDPWD=/home/seed/Desktop/lab-1  
-->  
>_= /usr/bin/env  
[09/01/20]seed@Chandan_PES1201701593:~/.task-5$
```

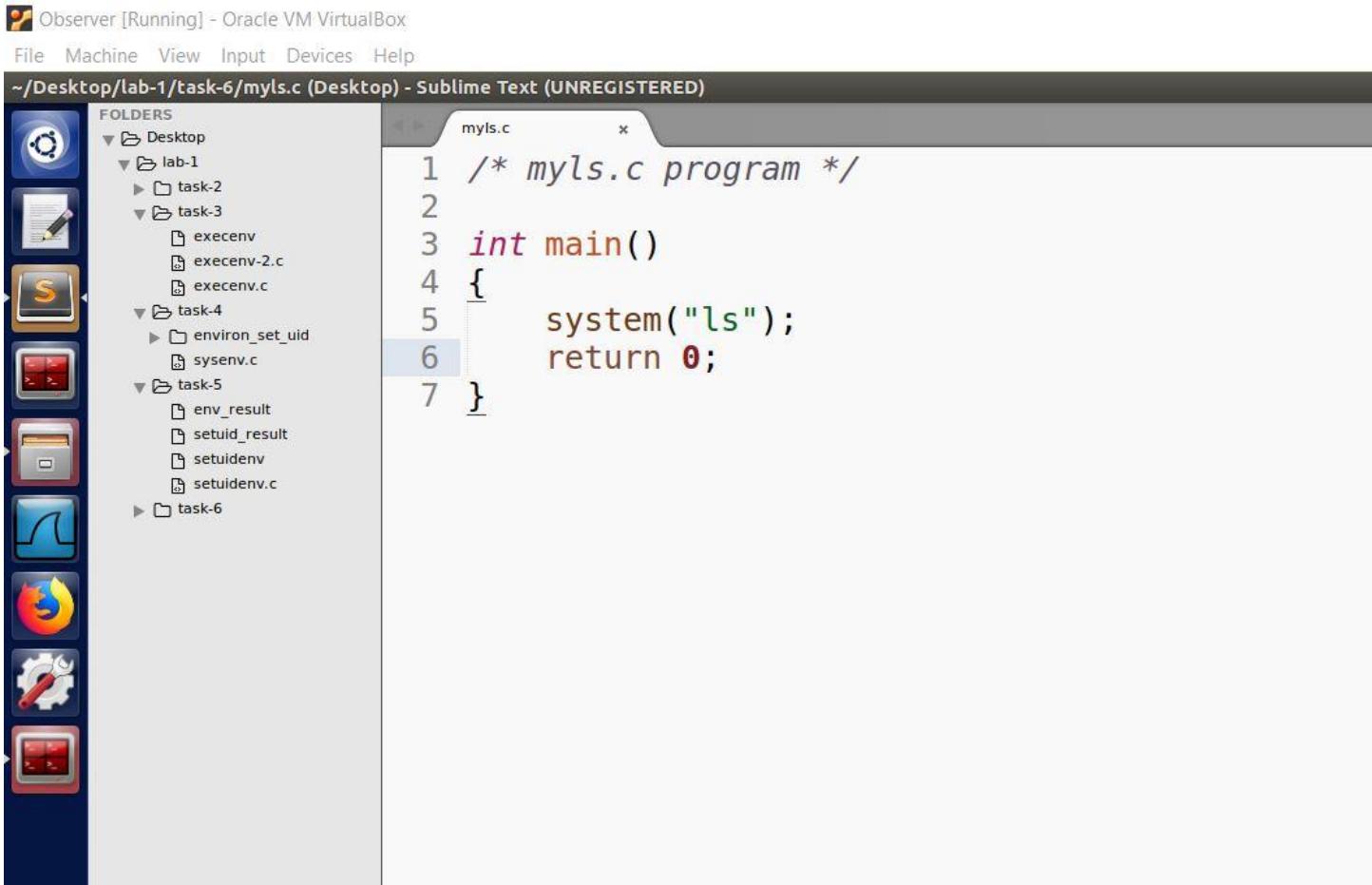
Task 6: The PATH Environment variable and Set-UID Programs

Calling `system()` within a Set-UID program is quite dangerous because of the shell program that it invokes. The actual behaviour of the shell can be affected by environment variables such as `PATH`. By changing these variables, malicious users can control the behaviour of the Set-UID program.

The `PATH` environment variable in bash can be changed using the `export` variable.

“ `export PATH=/home/seed:$PATH` ”

The below Set-UID program is supposed to execute the `/bin/ls` command, however, the programmer specifies just the relative path for “`ls`”, than its absolute path.



Observer [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

~/Desktop/lab-1/task-6/myls.c (Desktop) - Sublime Text (UNREGISTERED)

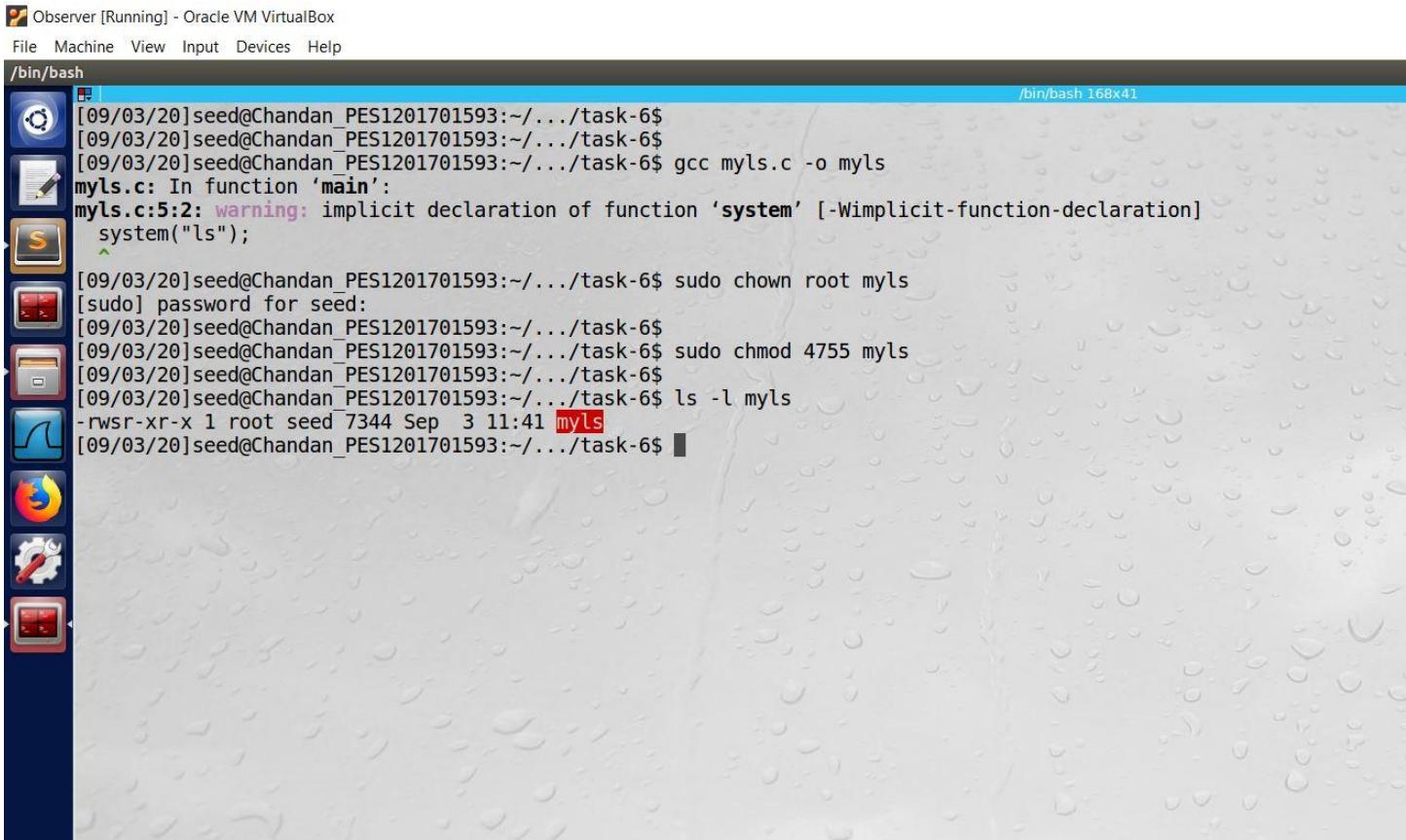
FOLDERS

- Desktop
 - lab-1
 - task-2
 - task-3
 - execenv
 - execenv-2.c
 - execenv.c
 - task-4
 - environ_set_uid
 - sysenv.c
 - task-5
 - env_result
 - setuid_result
 - setuidenv
 - setuidenv.c
 - task-6

myls.c

```
1 /* myls.c program */
2
3 int main()
4 {
5     system("ls");
6     return 0;
7 }
```

We compile the above program, change its ownership to root and make it a Set-UID program.



Observer [Running] - Oracle VM VirtualBox

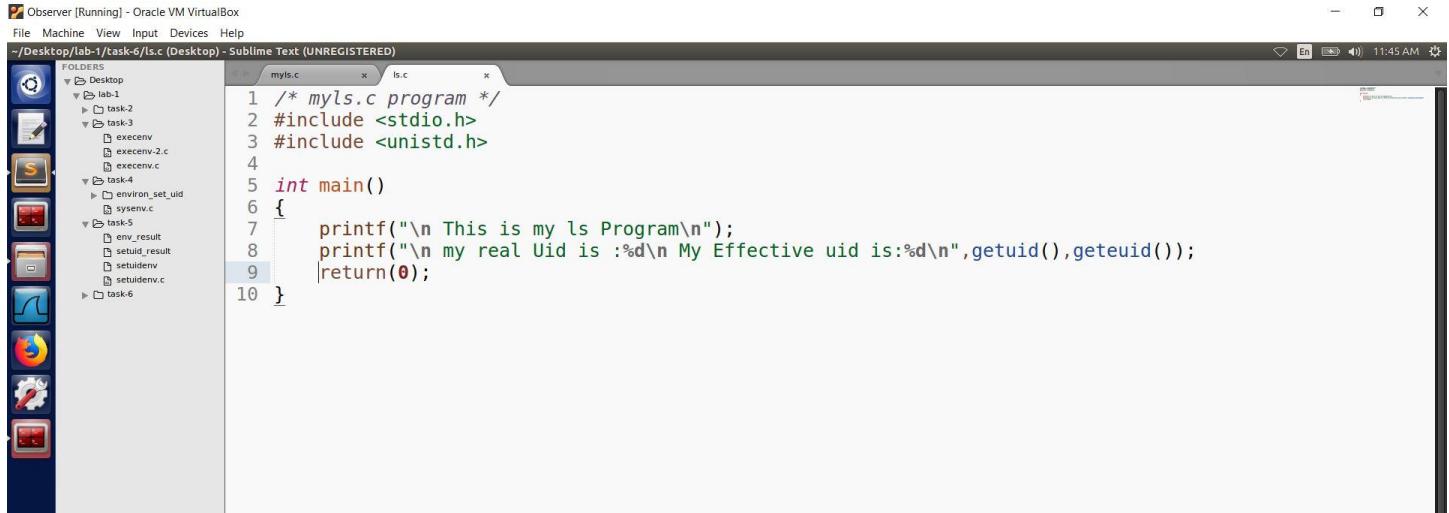
File Machine View Input Devices Help

/bin/bash

```
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ gcc myls.c -o myls
myls.c: In function 'main':
myls.c:5:2: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
    system("ls");
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ sudo chown root myls
[sudo] password for seed:
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ sudo chmod 4755 myls
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ ls -l myls
-rwsr-xr-x 1 root seed 7344 Sep  3 11:41 myls
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$
```

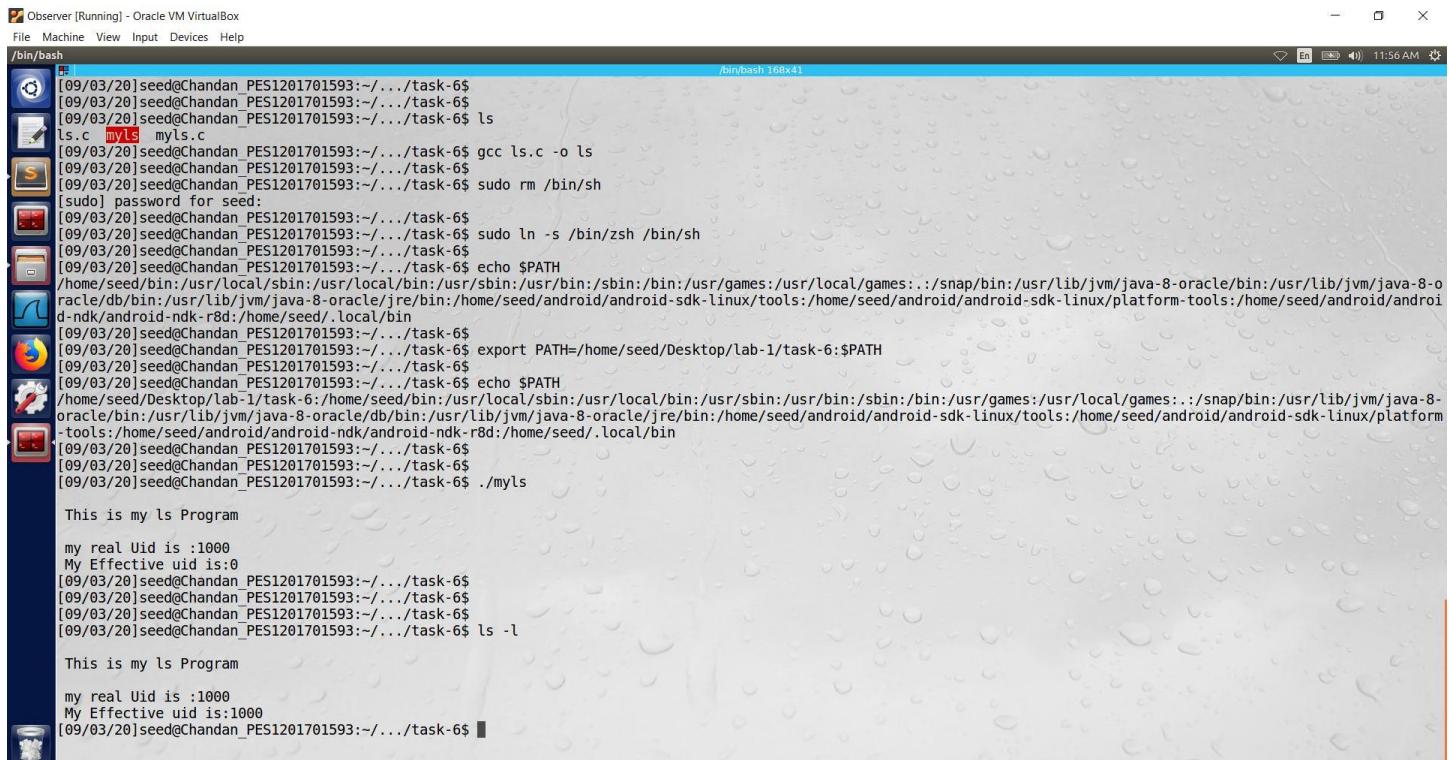
We can confirm that the above program is a Set-UID program by observing “-rwsr-xr-x” and its owner is set to root.

Now we will try to run our code instead of “/bin/ls” using the Set-UID program. In order to achieve this we first compile a program called ls which will be executed instead of /bin/ls. This program that we compile can have any logic that we/malicious user wants to achieve, for now we will just print few things.



```
/* myls.c program */
#include <stdio.h>
#include <unistd.h>
int main()
{
    printf("\n This is my ls Program\n");
    printf("\n my real Uid is :%d\n My Effective uid is:%d\n",getuid(),geteuid());
    return(0);
}
```

The above code is compiled to an executable called “ls”. Next, we change the value of the environment variable PATH, using the export keyword, and provide the path to our file, which could be malicious. This makes the program to search for the file in our directory, and as we have compiled our program as “ls”, the “myls” Set-UID program will in-turn execute our program (our “ls” program) instead of the /bin/ls program.



```
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ gcc ls.c -o ls
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ ls
ls.c  myls  myls.c
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ gcc ls.c -o ls
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ [sudo] password for seed:
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ sudo rm /bin/sh
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ sudo ln -s /bin/zsh /bin/sh
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ echo $PATH
/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-sdk/android-ndk-r8d:/home/seed/.local/bin
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ export PATH=/home/seed/Desktop/lab-1/task-6:$PATH
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ echo $PATH
/home/seed/Desktop/lab-1/task-6:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk-r8d:/home/seed/.local/bin
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ ./myls
This is my ls Program
my real Uid is :1000
My Effective uid is:0
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$ ls -l
This is my ls Program
my real Uid is :1000
My Effective uid is:1000
[09/03/20]seed@Chandan_PES1201701593:~/.../task-6$
```

In order to overcome the dash shell security mechanism of dropping SET-UID program's privileges, we link the “/bin/sh” to another shell as shown in the above screenshot.

Thus, we observe that the environment variables can be altered to control the behaviour of the Set-UID program specially when the shell is invoked. Moreover, we used a relative path instead of absolute path. As a result the system() invoked a shell which looks for a ls program in the “PATH” which was modified. Therefore changing the PATH environment variable to a folder containing malicious file with the same name as the program called by system(), the attacker can easily run the malicious code with root privileges.

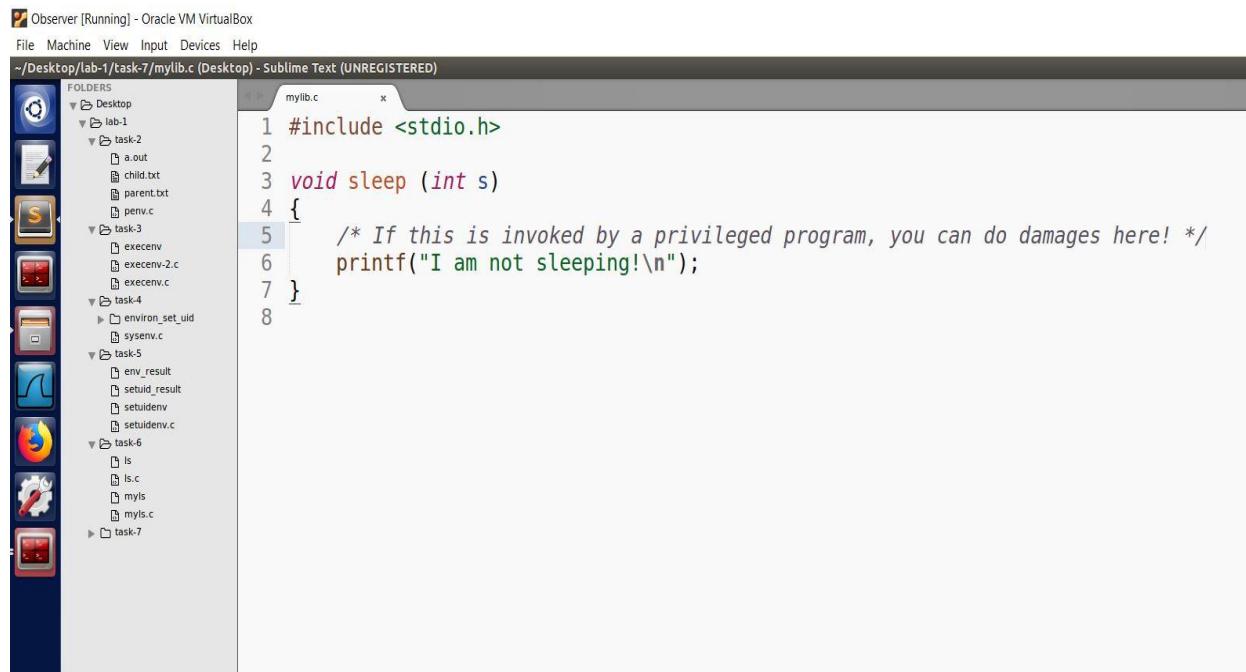
A Set-UID program with system() with relative path of the program can be a major vulnerability.

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

Here we will observe how Set-UID program deals with environment variables such as LD_PRELOAD, LD_LIBRARY_PATH etc. which influence the behaviour of the dynamic linker/loader in linux.

1. First, see how these environment variables influence the behavior of dynamic loader/linker when running a normal program.

We first build a dynamic link library called mylib.c, it overrides the sleep() in libc.



The screenshot shows a Sublime Text window with the file "mylib.c" open. The code contains a function that prints a message if it is invoked by a privileged program:`1 #include <stdio.h>
2
3 void sleep (int s)
4 {
5 /* If this is invoked by a privileged program, you can do damages here! */
6 printf("I am not sleeping!\n");
7 }`

Observer [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

/Desktop/lab-1/task-7/myprog.c (Desktop) - Sublime Text (UNREGISTERED)

```

1 /* myprog.c program */
2
3 int main()
4 {
5     sleep(1);
6     return 0;
7 }

```

FOLDERS

- Desktop
- lab-1
 - task-2
 - a.out
 - child.txt
 - parent.txt
 - penv.c
 - task-3
 - execenv
 - execenv-2.c
 - execenv.c
 - task-4
 - environ_set_uid
 - sysenv.c
 - task-5
 - env_result
 - setuid_result
 - setuidenv
 - setuidenv.c
 - task-6
 - ls
 - ls.c
 - myls
 - myls.c
 - task-7

- We compile the above program. Then set the LD_PRELOAD environment variable. Finally we compile the myprog program in the same directory as that of the dynamic link library as shown below.

Observer [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

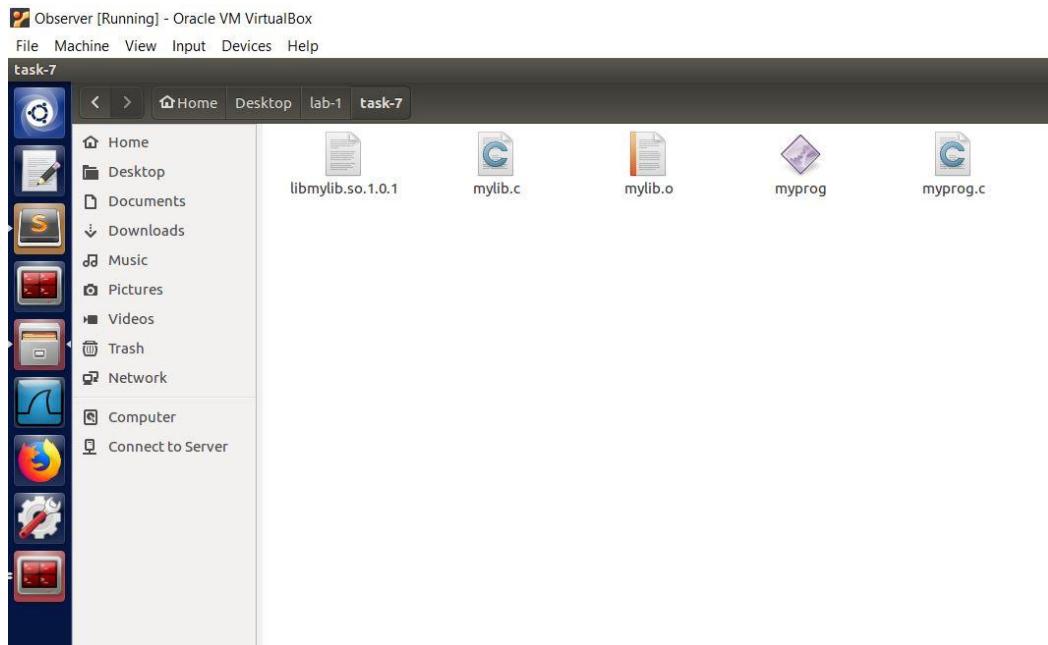
/bin/bash

```

[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ ls
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ gcc -fPIC -g -c mylib.c
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ ./myprog
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ myprog.c: In function 'main':
myprog.c:5:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    sleep(1);
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ 

```

- We can observe that all files are present in the same directory



4. Next we run the myprog program under different conditions and observe the result
 - a. Make myprog a regular program, and run it as a normal user: In this case we observe that the program calls the sleep() defined by us and also prints what we specified in the sleep() function.

```
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ [09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ [09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ la -l myprog -rwxrwxr-x 1 seed seed 7348 Sep 3 14:30 myprog [09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ [09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ ./myprog I am not sleeping! [09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
```

- b. Make myprog a Set-UID root program, and run it as a normal user: In this case we observe that the sleep() function in mylib was not called and the system-defined sleep() was called.

```

Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/bin/bash
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ la -l myprog
-rwxrwxr-x 1 seed seed 7348 Sep 3 14:30 myprog
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ sudo chown root myprog
[sudo] password for seed:
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ sudo chmod 4755 myprog
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ la -l myprog
-rwsr-xr-x 1 root seed 7348 Sep 3 14:30 myprog
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ ./myprog
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ ./myprog
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ 

```

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

In this case we observe that the user defined sleep() function from mylib is called instead of the system defined sleep() is called.

```

Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
root@Chandan_PES1201701593:/home/seed/Desktop/lab-1/task-7
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ la -l myprog
-rwsr-xr-x 1 root seed 7348 Sep 3 14:30 myprog
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ whoami
seed
[09/03/20]seed@Chandan_PES1201701593:~/.../task-7$ sudo -i
[sudo] password for seed:
root@Chandan_PES1201701593:#
root@Chandan_PES1201701593:#
root@Chandan_PES1201701593:~# whoami
root
root@Chandan_PES1201701593:~# cd /home/seed/Desktop/lab-1/task-7/
root@Chandan_PES1201701593:/home/seed/Desktop/lab-1/task-7# export LD_PRELOAD=./libmylib.so.1.0.1
root@Chandan_PES1201701593:/home/seed/Desktop/lab-1/task-7#
root@Chandan_PES1201701593:/home/seed/Desktop/lab-1/task-7# ./myprog
root@Chandan_PES1201701593:/home/seed/Desktop/lab-1/task-7# ./myprog
root@Chandan_PES1201701593:/home/seed/Desktop/lab-1/task-7# ./myprog
I am not sleeping!
root@Chandan_PES1201701593:/home/seed/Desktop/lab-1/task-7# 

```

- d. Make myprog a Set-UID “chandan” program (i.e., the owner is chandan, which is another user account), export the LD_PRELOAD environment variable again in a different user’s account (not-root user) and run it:

We first create a new user account “chandan” as shown below.

```
Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$ whoami
seed
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$ su chandan
Password:
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$ whoami
chandan
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$
```

Next, we make this file's owner as chandan and make it a SET-UID program. After this, we log into the chandan's account and set the LD_PRELOAD variable again. On running the program again, we see that user-defined sleep function is called.

```
Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$ ls
libmylib.so.1.0.1 mylib.c mylib.o myprog myprog.c
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$ sudo chown chandan myprog
[sudo] password for seed:
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$ sudo chmod 4755 myprog
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$ la -l myprog
-rwsr-xr-x 1 chandan seed 7348 Sep 3 14:30 myprog
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$
[09/03/20] seed@Chandan_PES1201701593:~/.task-7$ su chandan
Password:
chandang@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$
chandang@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$ export LD_PRELOAD=./libmylib.so.1.0.1
chandang@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$
chandang@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$ printenv LD_PRELOAD
./libmylib.so.1.0.1
chandang@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$ la -l myprog
-rwsr-xr-x 1 chandan seed 7348 Sep 3 14:30 myprog
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$ ./myprog
I am not sleeping!
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$
chandan@Chandan_PES1201701593: /home/seed/Desktop/lab-1/task-7$
```

Observing the above 4 cases, we can conclude that in the cases where the user-defined sleep() function was called, the LD_PRELOAD environment variable was present/inherited thus the mylib library function was called. In the other case where the system defined sleep() was called the LD_PRELOAD environment variable was not present as it was dropped by the Set-UID child process.

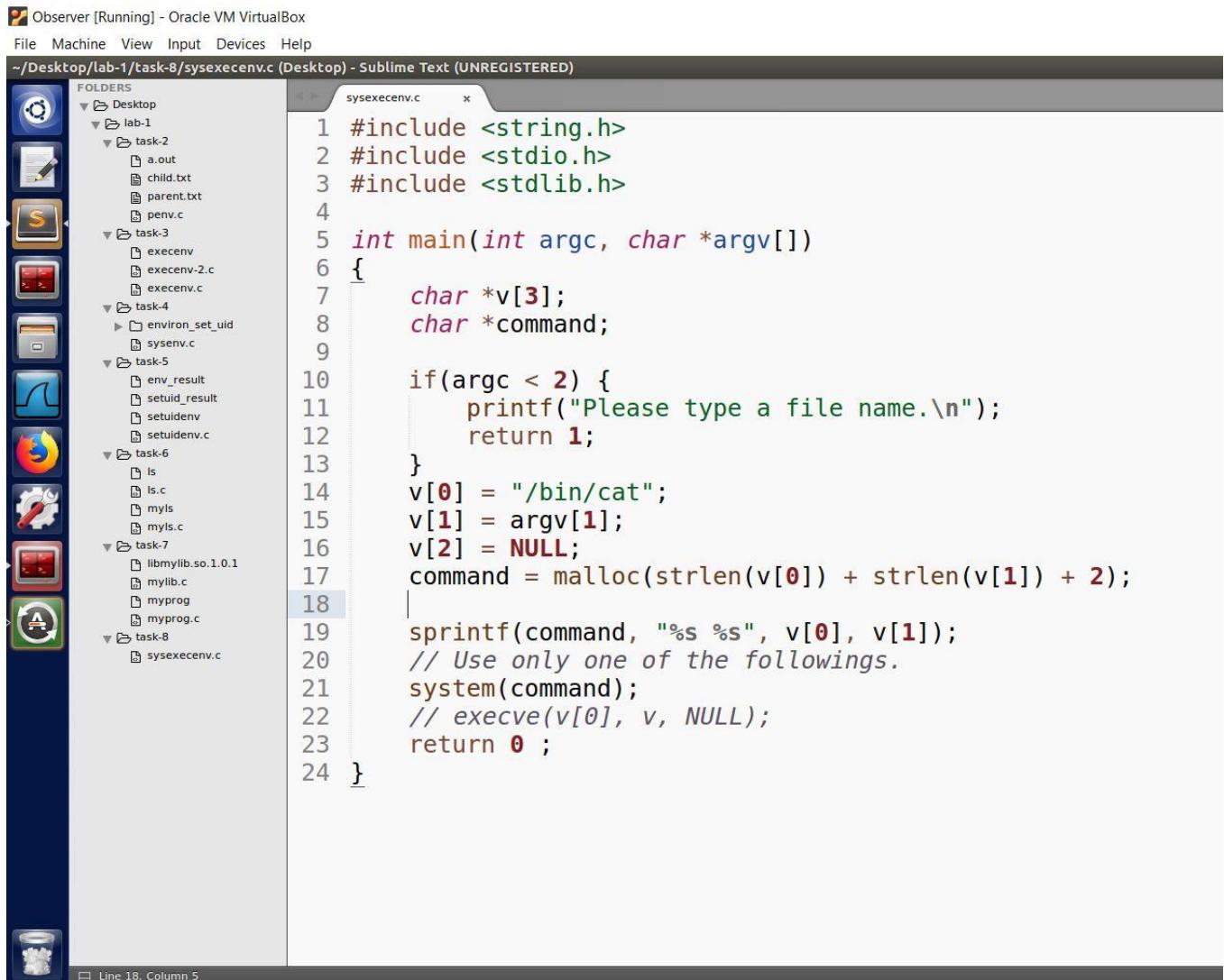
The LD_PRELOAD environment variable is dropped if the real and effective id's are different. This is because of Set-UID security mechanism as mentioned in this article

http://web.archive.org/web/20130711061934/http://www.jayconrod.com/cgi/view_post.py?23 and <https://stackoverflow.com/questions/9232892/ld-preload-with-setuid-binary>.

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

We have observed that system() is quite dangerous when compared execve() when used with Set-UID programs. Apart from affecting environment variables, invoking a shell has various other consequences which we will try to observe in this task.

1. We compile the below program, make root its owner and make it a Set-UID program. The program we run uses system() to invoke the command Bob specifies.



```
#include <string.h>
#include <stdio.h>
#include <stdlib.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 ;
}
```

2. We create two files, “rootfile” whose owner is root and “myfile” whose owner is a non-root account (seed).

The screenshot shows a Linux desktop environment with a window manager. In the top row, there are two Sublime Text windows: one titled "myfile" containing the text "1 This is myfile, OWNER is SEED." and another titled "rootfile" containing the text "1 This is the root file, OWNER is ROOT.". Below these windows is a terminal window titled "/bin/bash" with the following command history:

```

[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ ls -l rootfile myfile
-rw-rw-r-- 1 seed seed 30 Sep 5 12:05 myfile
-rw-rw-r-- 1 seed seed 37 Sep 5 12:05 rootfile
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ sudo chown root rootfile
[sudo] password for seed:
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ ls -l rootfile myfile
-rw-rw-r-- 1 seed seed 30 Sep 5 12:05 myfile
-rw-rw-r-- 1 root seed 37 Sep 5 12:05 rootfile
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 

```

The terminal window also shows a vertical dock on the left side with icons for various applications like a browser, file manager, and system tools.

3. We run the Set-UID program `sysexecenv` which uses `system()` and passing “myfile;rm rootfile” as the argument to the executable as shown below.

After executing the `sysexecenv` program we observe that the “rootfile” is no longer available and is been deleted. If we observe this it is evident that the program reads the entire input i.e. reads the file name “myfile” and later also executes the command “`rm rootfile`”. Since this being a Set-UID program it has the root privileges even though the program is run by a normal user (Bob). So even if Bob had no permission to write, he can easily remove the file with the root privileges. Thus `system()` is quite dangerous in case of Set-UID programs.

```

[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ gcc sysexecenv.c -o sysexecenv
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ sudo chown root sysexecenv
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ sudo chmod 4755 sysexecenv
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ ls -l sysexecenv
-rwsr-xr-x 1 root seed 7552 Sep 5 12:00 sysexecenv
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ ls -l sysexecenv rootfile myfile
-rw-rw-r-- 1 seed seed 30 Sep 5 12:05 myfile
-rw-rw-r-- 1 root seed 37 Sep 5 12:05 rootfile
-rwsr-xr-x 1 root seed 7552 Sep 5 12:00 sysexecenv
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ ./sysexecenv "myfile;rm rootfile"
This is myfile, OWNER is SEED. [09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ ls -l rootfile
ls: cannot access 'rootfile': No such file or directory
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ ls -l
total 16
-rw-rw-r-- 1 seed seed 30 Sep 5 12:05 myfile
-rwsr-xr-x 1 root seed 7552 Sep 5 12:00 sysexecenv
-rw-rw-r-- 1 seed seed 435 Sep 5 11:51 sysexecenv.c
[09/05/20]seed@Chandan_PES1201701593:~/....task-8$ 

```

4. Now we replace the system() call with execve() unlike system() considers the input from user as a string rather than considering it a command. We compile it, make its owner root and make it a Set-UID program.

```

1 #include <string.h>
2 #include <stdio.h>
3 #include <stdlib.h>
4 
5 int main(int argc, char *argv[])
6 {
7     char *v[3];
8     char *command;
9 
10    if(argc < 2) {
11        printf("Please type a file name.\n");
12        return 1;
13    }
14    v[0] = "/bin/cat";
15    v[1] = argv[1];
16    v[2] = NULL;
17    command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
18 
19    sprintf(command, "%s %s", v[0], v[1]);
20    // Use only one of the followings.
21    // system(command);
22    execve(v[0], v, NULL);
23    return 0 ;
24 }

```

5. We run the above compiled program with the same string as argument to the executable. But this time instead of considering it as commands, the entire string is treated as user input string thus printing an error 'myfile;rm rootfile': No such file or directory unlike system(). Thus execve is safer than system() especially with Set-UID programs.

```

[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ gcc sysexecenv.c -o sysexecenv
sysexecenv.c: In function 'main':
sysexecenv.c:22:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
    execve(v[0], v, NULL);
^
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ sudo chown root sysexecenv
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ sudo chmod 4755 sysexecenv
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ ls -l sysexecenv rootfile myfile
-rw-rw-r-- 1 seed seed 31 Sep  5 12:13 myfile
-rw-rw-r-- 1 root seed 37 Sep  5 12:15 rootfile
-rwsr-xr-x 1 root seed 7552 Sep  5 12:16 sysexecenv
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ ./sysexecenv "myfile;rm rootfile"
/bin/cat: 'myfile;rm rootfile': No such file or directory
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ ls -l rootfile
-rw-rw-r-- 1 root seed 37 Sep  5 12:15 rootfile
[09/05/20]seed@Chandan_PES1201701593:~/.../task-8$ 

```

Task 9: Capability Leaking

Set-UID programs permanently relinquish their root privileges if such privileges are not needed. But sometimes the root privileges must be revoked and setuid() system call can be used. setuid() sets the effective user ID of calling process and if the effective UID of caller is root, the real UID and saved set-user-ID are also set.

But the above is associated with capability leaking. In this case although the effective user ID of the process becomes non-privileged, the process is still privileged because it possesses privileged capabilities.

1. We compile the below program, make its owner root and make it a Set-UID program.

```

Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
~/Desktop/lab-1/task-9/capleak.c (Desktop) - Sublime Text (UNREGISTERED)
FOLDERS
  Desktop
    lab-1
      task-2
        a.out
        child.txt
        parent.txt
        penv.c
      task-3
        execenv
        execenv-2.c
        execenv.c
      task-4
        environ_set_uid
      sysenv.c
    task-5
      env_result
      setuid_result
      setuidenv
      setuidenv.c
    task-6
      ls
      ls.c
      myls
      myls.c
    task-7
      libmylib.so.1.0.1
      mylib.c
      myprog
      myprog.c
    task-8
      myfile
      rootfile
      sysexecenv
      sysexecenv.c
  task-9
    capleak.c
Line 25, Column 19

```

```

capleak.c
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4 void main()
5 {
6     int fd;
7     /* Assume that /etc/zzz is an important system file,
8      * and it is owned by root with permission 0644.
9      * Before running this program, you should create
10     * the file /etc/zzz first. */
11
12     fd = open("/etc/zzz", O_RDWR | O_APPEND);
13     if (fd == -1) {
14         printf("Cannot open /etc/zzz\n");
15         exit(0);
16     }
17     /* Simulate the tasks conducted by the program */
18     sleep(1);
19
20     /* After the task, the root privileges are no longer needed,
21      * it's time to relinquish the root privileges permanently. */
22     setuid(getuid()); /* getuid() returns the real uid */
23
24
25     if (fork()) {
26         /* In the parent process */
27         close(fd);
28         exit(0);
29     }
30     else {
31         /* in the child process */
32         /* Now, assume that the child process is compromised, malicious
33         attackers have injected the following statements
34         into this process */
35         write(fd, "Malicious Data\n", 15);
36         close(fd);
37     }
38 }

```

- We create a file named zzz at /etc/ with its owner being root assuming it to be a very important file.

```

Observer [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
/etc/zzz (Desktop) - Sublime Text (UNREGISTERED)
FOLDERS
  Desktop
    lab-1
      task-2
        a.out
        child.txt
        parent.txt
        penv.c
      task-3
        execenv
        execenv-2.c
        execenv.c
      task-4
        environ_set_uid
        sysenv.c
    task-5

```

1 This is a VERY IMPORTANT file!!!

- We compile and run the capleak program as shown below.

Observer [Running] - Oracle VM VirtualBox

File Machine View Input Devices Help

/bin/bash

```
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ ls capleak.c [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ gcc capleak.c -o capleak capleak.c: In function 'main': capleak.c:18:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration] sleep(1); ^ capleak.c:22:2: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration] setuid(getuid()); /* getuid() returns the real uid */ ^ capleak.c:22:9: warning: implicit declaration of function 'getuid' [-Wimplicit-function-declaration] setuid(getuid()); /* getuid() returns the real uid */ ^ capleak.c:25:6: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration] if (fork()) { ^ capleak.c:27:3: warning: implicit declaration of function 'close' [-Wimplicit-function-declaration] close (fd); ^ capleak.c:35:3: warning: implicit declaration of function 'write' [-Wimplicit-function-declaration] write (fd, "Malicious Data\n", 15); ^ [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ sudo chown root capleak [sudo] password for seed: [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ sudo chmod 4755 capleak [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ ls -l capleak -rwsr-xr-x 1 root seed 7640 Sep 5 20:38 capleak [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ cat /etc/zzz This is a VERY IMPORTANT file!!! [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ ls -l /etc/zzz -rw-r--r-- 1 root root 33 Sep 5 20:43 /etc/zzz [09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ ]
```

The screenshot shows a terminal window titled '/bin/bash' with a resolution of 168x41. The terminal content is as follows:

```
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$  
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$  
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$  
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ ./cableak  
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$  
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ cat /etc/z  
zsh/  
          zsh_command not found zzz  
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$ cat /etc/zzz  
This is a VERY IMPORTANT file!!!  
Malicious Data  
[09/05/20]seed@Chandan_PES1201701593:~/.../task-9$
```

After running the capleak program we observe that the content of /etc/zzz file is modified as seen in the above screenshot. This happened because even if the privileges were dropped the file

was still open with all the privileges and after forking the attacker can easily modify the contents of the file. Thus closing the file descriptor was essential to prevent the capability leak.

.....
THANK YOU
.....