

## Task2

### 1. Task Purpose

In this task, we want to know whether the parent's environment variables are inherited by the child process or not.

### 2. Progress

<task2.c>

```
child.txt x parent.txt x task2.c x
#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);
    }
}
```

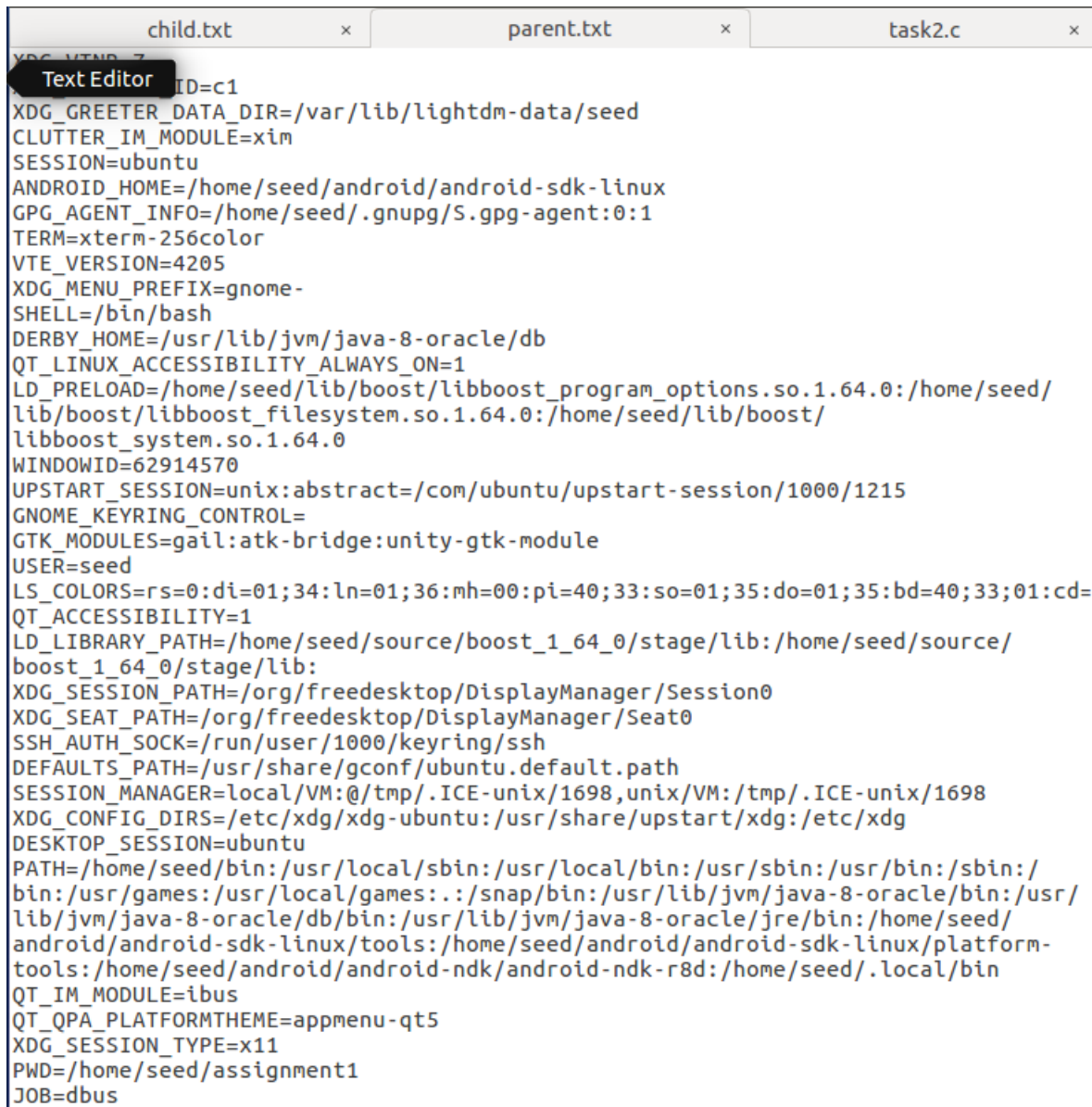
```
[03/20/24]seed@VM:~/assignment1$ vi task2.c
[03/20/24]seed@VM:~/assignment1$ gcc -o task2 task2.c
[03/20/24]seed@VM:~/assignment1$ touch child.txt
[03/20/24]seed@VM:~/assignment1$ ls -l
total 12
-rw-rw-r-- 1 seed seed    0 Mar 20 11:57 child.txt
-rwxrwxr-x 1 seed seed 7496 Mar 20 11:56 task2
-rw-rw-r-- 1 seed seed  448 Mar 20 11:56 task2.c
[03/20/24]seed@VM:~/assignment1$ task2 > child.txt
[03/20/24]seed@VM:~/assignment1$ touch parent.txt
[03/20/24]seed@VM:~/assignment1$ vi task2.c
[03/20/24]seed@VM:~/assignment1$ gcc -o task2 task2.c
[03/20/24]seed@VM:~/assignment1$ task2 > parent.txt
[03/20/24]seed@VM:~/assignment1$ diff child.txt parent.
txt
[03/20/24]seed@VM:~/assignment1$
```

### 3. Result

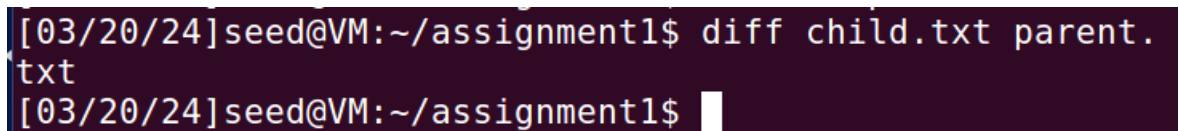
<child.txt>

```
child.txt      x      parent.txt      x      task2.c      x
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0
WINDOWID=62914570
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1215
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=
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
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1698,unix/VM:/tmp/.ICE-unix/1698
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
DESKTOP_SESSION=ubuntu
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
PWD=/home/seed/assignment1
JOB=dbus
```

<parent.txt>



```
child.txt x parent.txt x task2.c x
Text Editor
ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0
WINDOWID=62914570
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1215
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=
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
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1698,unix/VM:/tmp/.ICE-unix/1698
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-ndk/android-ndk-r8d:/home/seed/.local/bin
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
PWD=/home/seed/assignment1
JOB=dbus
```



```
[03/20/24]seed@VM:~/assignment1$ diff child.txt parent.txt
[03/20/24]seed@VM:~/assignment1$
```

#### 4. Consideration

When `fork()` is called, the child process creates an exact copy of the parent process's environment variables. Therefore, if neither the parent nor the child process modifies the environment variables thereafter, the files outputting the environment variables of both processes should be identical. That's why, when comparing `parent.txt` and `child.txt` using the `diff` command, there were no differences found.

## Task3

### 1. Task Purpose

We want to know whether environment variables are inherited by the new program when the function `execve()` makes a system call to load and execute a new command.

### 2. Progress

<task3.c>



```
task3.c (~/assignment1) - gedit
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

extern char **environ;

int main()
{
    char *argv[2];

    argv[0] = "/usr/bin/env";
    argv[1] = NULL;

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

    return 0;
}
```

#### Step1

```
[03/20/24]seed@VM:~/assignment1$ gcc -o task3 task3.c
[03/20/24]seed@VM:~/assignment1$ ./task3
[03/20/24]seed@VM:~/assignment1$ task3
[03/20/24]seed@VM:~/assignment1$ task3 > task3_result.txt
```

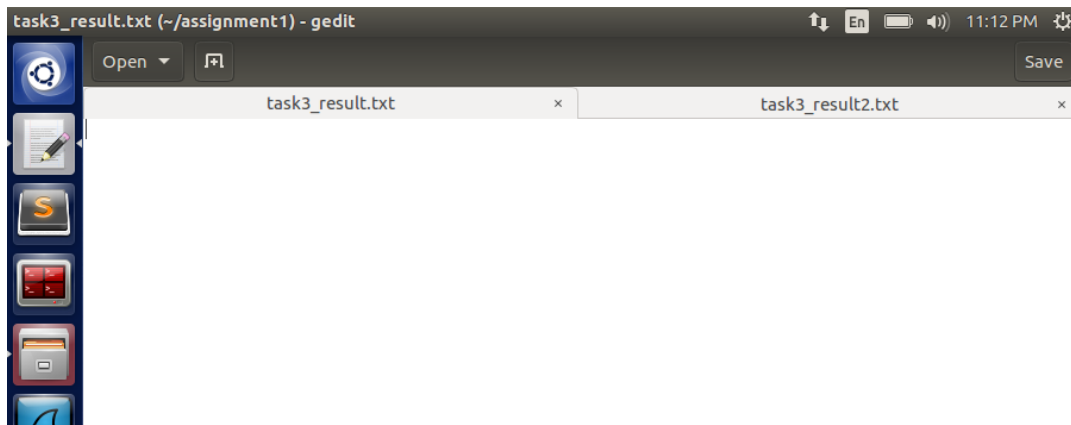
#### Step2

```
[03/20/24]seed@VM:~/assignment1$ gcc -o task3 task3.c
[03/20/24]seed@VM:~/assignment1$ ./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
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT LINUX ACCESSIBILITY ALWAYS ON=1
```

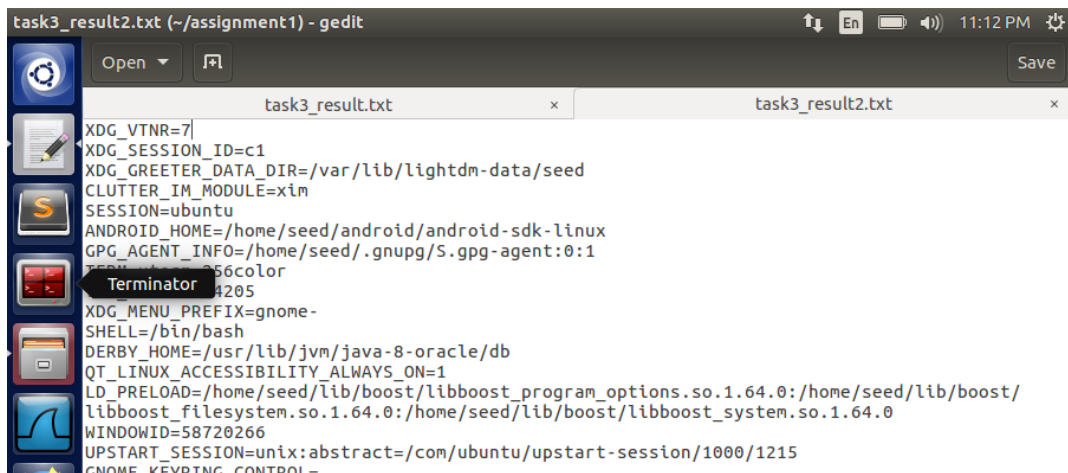
```
[03/20/24]seed@VM:~/assignment1$ task3 > task3_result2.txt
```

### 3. Result

#### Step 1:



#### Step 2:



### 4. Consideration

When creating a process using the `execve()` system call, the memory space of the current process is overwritten by the new program. (During the call to the `execve` function, the array of environment variables provided as the third argument determines the environment variables for the new program.) In Step 1, this array was set to `NULL`, resulting in no environment variables being passed, and thus nothing was outputted. Conversely, in Step 2, the environment variables array from the calling process was passed, leading to all variables in this array being inherited by the new program and outputted by `/usr/bin/env`. This demonstrates that when calling `execve`, environment variables must be explicitly passed to the new process, and this establishes the environment variables that will be used by the new program.

## Task5

### 1. Task Purpose

We want to figure out whether environment variables are inherited by the Set-UID program's process from the user's process.

### 2. Progress

<task5.c>



```
task5.c (~/.assignment1) - gedit
Open Save
before_export.txt x after_export.txt x task5.c x
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

void main()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}
```

```
[03/21/24]seed@VM:~/assignment1$ vi task5.v
[03/21/24]seed@VM:~/assignment1$ vi task5.c
[03/21/24]seed@VM:~/assignment1$ gcc -o task5 task5.c
[03/21/24]seed@VM:~/assignment1$ sudo chown root task5
[03/21/24]seed@VM:~/assignment1$ sudo chmod 4755 task5
[03/21/24]seed@VM:~/assignment1$ whoami
seed
[03/21/24]seed@VM:~/assignment1$ task5 > before_export.txt
```

```
[]seed@VM:~/assignment1$ export PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
[03/21/24]seed@VM:~/assignment1$ export LD_LIBRARY_PATH=/this/is/modified/libraries
[03/21/24]seed@VM:~/assignment1$ export MY_NAME=seoyeong
[03/21/24]seed@VM:~/assignment1$ task5 > after_export.txt
```

### 3. Result



```

[03/21/24]seed@VM:~/assignment1$ diff before_export.txt after_export.txt
13a14
> 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
20a22
> LD_LIBRARY_PATH=/this/is/modified/libraries
28c30
< 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
---
> PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
35a38
> MY_NAME=seoyeong

```

#### 4. Consideration

The environment variables I set goes into a set-UID child process. Since a Set-UID program with root privileges has the authority to make extensive changes to the system, manipulating the environment variables passed through such a program can compromise the fundamental security of the system. For example, an attacker could manipulate the LD\_LIBRARY\_PATH environment variable to inject a malicious shared library, thereby causing a process with root privileges to execute malicious code. Alternatively, by changing the PATH environment variable, it's possible to manipulate a process with root privileges to search for executable files in a location specified by the user, thus injecting and executing malicious code into the system.

#### Task7

##### 1. Task Purpose

We will verify how the inheritance of a process's environment variables changes depending on what the RUID and EUID are.

##### 2. Progress & 3. Result

##### (1) Regular User Running a Regular Program:

```

Terminator 1 - seoyeong
[03/21/24]seed@VM:~/assignment1$ vi mylib.c
[03/22/24]seed@VM:~/assignment1$ gcc -fPIC -g -c mylib.c
[03/22/24]seed@VM:~/assignment1$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[03/22/24]seed@VM:~/assignment1$ export LD_PRELOAD=./libmylib.so.1.0.1
[03/22/24]seed@VM:~/assignment1$ vi myprog.c
[03/22/24]seed@VM:~/assignment1$ gcc myprog.c -o myprog

```

```
[03/22/24]seed@VM:~/assignment1$ vi myprog.c
[03/22/24]seed@VM:~/assignment1$ gcc myprog.c -o myprog
[03/22/24]seed@VM:~/assignment1$ who am i
[03/22/24]seed@VM:~/assignment1$ whoami
seed
[03/22/24]seed@VM:~/assignment1$ ./myprog
I am not sleeping!
```

(2) Regular User Running a Set-UID Program Owned by Root:

```
[03/22/24]seed@VM:~/assignment1$ gcc myprog.c -o myprog
[03/22/24]seed@VM:~/assignment1$ sudo chown root myprog
[03/22/24]seed@VM:~/assignment1$ sudo chmod 4755 myprog
[03/22/24]seed@VM:~/assignment1$ whoami
seed
[03/22/24]seed@VM:~/assignment1$ ./myprog
[03/22/24]seed@VM:~/assignment1$
```

(3) Root User Running a Set-UID Program Owned by Root:

```
root@VM:~/assignment1# unset LD_PRELOAD
root@VM:~/assignment1# gcc -fPIC -g -c mylib.c
root@VM:~/assignment1# gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
root@VM:~/assignment1# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:~/assignment1# vi myprog.c
root@VM:~/assignment1# ls -al
total 28
drwxr-xr-x 2 root root 4096 Mar 22 06:25 .
drwx----- 7 root root 4096 Mar 22 06:25 ..
-rwxr-xr-x 1 root root 7920 Mar 22 06:24 libmylib.so.1.0.1
-rw-r--r-- 1 root root 155 Mar 22 06:21 mylib.c
-rw-r--r-- 1 root root 2580 Mar 22 06:23 mylib.o
-rw-r--r-- 1 root root 73 Mar 22 06:25 myprog.c
root@VM:~/assignment1# gcc myprog.c -o myprog
root@VM:~/assignment1# sudo chown root myprog
root@VM:~/assignment1# sudo chmod 4755 myprog
root@VM:~/assignment1# ./myprog
I am not sleeping!
```

(4) Regular User Running a Set-UID Program Owned by a Different Regular User:

```
user1@VM:~/assignment1$ vi mylib.c
user1@VM:~/assignment1$ gcc -fPIC -g -c mylib.c
user1@VM:~/assignment1$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
user1@VM:~/assignment1$ export LD_PRELOAD=./libmylib.so.1.0.1
user1@VM:~/assignment1$ vi myprog.c
user1@VM:~/assignment1$ gcc myprog.c -o myprog
```



```
user1@VM:~/assignment1$ sudo chown seed myprog
user1@VM:~/assignment1$ sudo chmod 4755 myprog
user1@VM:~/assignment1$ ./myprog
user1@VM:~/assignment1$
```

#### 4. Consideration

##### (1) Regular User Running a Regular Program:

The LD\_PRELOAD is respected. Our custom library is preloaded, and the custom sleep function is called. This is the expected behavior since there are no elevated privileges involved.

##### (2) Regular User Running a Set-UID Program Owned by Root:

The custom sleep function does not execute, indicating that LD\_PRELOAD is ignored. This is a security feature of most Unix-like systems; they ignore the LD\_PRELOAD environment variable for Set-UID programs when run by non-owner users to prevent privilege escalation. ( If a process's real ID and effective ID are different, the LD\_PRELOAD environment variables are ignored. )

##### (3) Root User Running a Set-UID Program Owned by Root:

The custom sleep function executes, which means LD\_PRELOAD is not ignored when the root user runs the program. The root user inherently has all privileges, so the system does not need to protect against privilege escalation. (Because the process's real ID and effective ID are the same, the LD\_PRELOAD environment variables are not ignored and are executed.)

##### (4) Regular User Running a Set-UID Program Owned by a Different Regular User:

Set-UID programs ignore LD\_PRELOAD regardless of ownership when executed by non-root users. Since the process's real ID and effective ID are different, LD\_PRELOAD is ignored and the sleep() function is executed.