

Environment Variable and Set-UID Program Lab

Task 1: Manipulating environment variables

In this task, we study the commands that can be used to set and unset environment variables.

Use `printenv` or `env` command to print out the environment variables. There are some particular environment variables, such as `PWD`, we can use "`printenv PWD`" or "`env | grep PWD`".

Use `export` and `unset` to set or unset environment variables.



```
Terminal
[02/19/24]seed@VM:~/.../lab$ printenv PWD
/home/seed/Documents/lab
[02/19/24]seed@VM:~/.../lab$ env | grep PWD
PWD=/home/seed/Documents/lab
[02/19/24]seed@VM:~/.../lab$ unset PWD
[02/19/24]seed@VM:~/.../lab$ env | grep PWD
[02/19/24]seed@VM:~/.../lab$ export PWD="try"
[02/19/24]seed@VM:try$ printenv PWD
try
[02/19/24]seed@VM:try$ export PWD="/home/seed"
[02/19/24]seed@VM:~$ printenv PWD
/home/seed
[02/19/24]seed@VM:~$
```

Task 2: Inheriting environment variables from parents

In this task, we study how environment variables are inherited by child processes from their parents. In Unix, `fork()` creates a new process by duplicating the calling process. The new process, referred to as the child, is an exact duplicate of the calling process, referred to as the parent; however, several things are not inherited by the child (please see the manual of `fork()` by typing the following command: `man fork`). In this task, we would like to know whether the parent's environment variables are inherited by the child process or not.

Code used for this task:



The image shows a code editor window with a C program and a terminal window showing the execution of the program. The code editor has a sidebar with icons for various applications. The C code defines a function `printenv()` that iterates through the `environ` array and prints each environment variable. The `main()` function forks a child process, which calls `printenv()`, while the parent process also calls `printenv()` before exiting. The terminal window shows the compilation of `myprintenv.c` into `child` and the execution of `./child`, which outputs the current environment variables.

```
#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);
    }
}
```

Terminal output:

```
[02/19/24]seed@VM:~/.../lab$ gcc myprintenv.c -o child
[02/19/24]seed@VM:~/.../lab$ ./child
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_option.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=52429880
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1342
GNOME_KEYRING_CONTROL=
```

```
Terminal
XDG_SEAT=seat0
LANGUAGE=en_US
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=seed
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-nZCBDhD7vt
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
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
./child
[02/19/24]seed@VM:~/.../lab$
```

```
Terminal
[02/19/24]seed@VM:~/.../lab$ gcc myprintenv.c -o parent
[02/19/24]seed@VM:~/.../lab$ ./parent
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=52429880
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1342
GNOME_KEYRING_CONTROL=
```

```
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
./parent
[02/19/24]seed@VM:~/.../lab$
```

```
[02/19/24]seed@VM:~/.../lab$ ./child > count.txt
[02/19/24]seed@VM:~/.../lab$ ./parent > pout.txt
[02/19/24]seed@VM:~/.../lab$
```



```
[02/19/24]seed@VM:~/.../lab$ diff count.txt pout.txt
67c67
< _=./child
---
> _=./parent
[02/19/24]seed@VM:~/.../lab$
```

The program prints all environment variables in the parent or the child process

From the screenshot above, Parent process and child process have the same environment variables since they have the same output.

Task 3: Environment variables and execve()

In this task, we study 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; this function never returns. No new process is created; instead, the calling process's text, data, bss, and stack are overwritten by that of the program loaded. Essentially, `execve()` runs the new program inside the calling process.

Code used for this task:

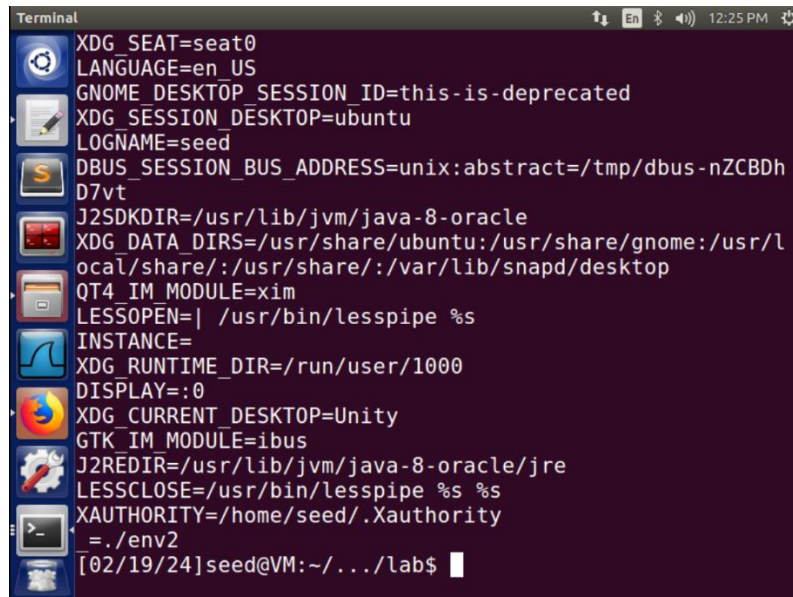
```
myprintenv.c
#include <unistd.h>
extern char **environ;
int main()
{
    char *argv[2];
    argv[0] = "/usr/bin/env";
    argv[1] = NULL;
    execve("/usr/bin/env", argv, NULL);
    return 0 ;
}
```

```
[02/19/24]seed@VM:~/.../lab$ gcc myenv.c -o env1
[02/19/24]seed@VM:~/.../lab$ ./env1
[02/19/24]seed@VM:~/.../lab$
```

```
*myenv.c (~/Downloads/lab) - gedit
myprintenv.c
#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 ;
}
```

```
Terminal
[02/19/24]seed@VM:~/.../lab$ gcc myenv.c -o env2
[02/19/24]seed@VM:~/.../lab$ ./env2
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_option.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=52429880
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1342
GNOME_KEYRING_CONTROL=
```

```
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
./env2
[02/19/24]seed@VM:~/.../lab$
```

A terminal window with a dark purple background and white text. It lists various environment variables such as XDG_SEAT, LANGUAGE, GNOME_DESKTOP_SESSION_ID, XDG_SESSION_DESKTOP, LOGNAME, DBUS_SESSION_BUS_ADDRESS, J2SDKDIR, XDG_DATA_DIRS, ocal/share, QT4_IM_MODULE, LESSOPEN, INSTANCE, XDG_RUNTIME_DIR, DISPLAY, XDG_CURRENT_DESKTOP, GTK_IM_MODULE, J2REDIR, LESSCLOSE, XAUTHORITY, and a command to source a file named env2. The prompt at the bottom is [02/19/24]seed@VM:~/.../lab\$.

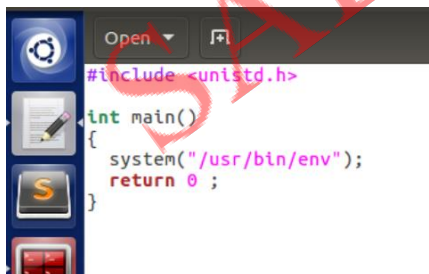
```
Terminal
XDG_SEAT=seat0
LANGUAGE=en_US
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=seed
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-nZCDBhD7vt
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
LESSOPEN=| /usr/bin/lesspipe %s
INSTANCE=
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
= ./env2
[02/19/24]seed@VM:~/.../lab$
```

The program compiled in step 1 (env1) prints nothing, whereas the program in step 2 (env2) prints all environment variables. Therefore we know the environment variables are not inherited in the program. Instead, the program get them by external pointer `environ`.

Task 4: Environment variables and system()

In this task, we study how environment variables are affected when a new program is executed via the system() function.

Code used for this task:

A code editor window with a dark background and light-colored text. It shows a C program that includes <unistd.h>, defines a main function, and calls system("/usr/bin/env") before returning 0.

```
Open [ ]
#include <unistd.h>

int main()
{
    system("/usr/bin/env");
    return 0 ;
}
```

```
Terminal
[02/19/24]seed@VM:~/.../lab$ gcc env.c -o env
[02/19/24]seed@VM:~/.../lab$ ./env
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
SHLVL=1
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
HOME=/home/seed
QT4_IM_MODULE=xim
DESKTOP_SESSION=ubuntu
GTK_LINUX_ACCESSIBILITY_ALWAYS_ON=1
GTK_MODULES=gail:atk-bridge:unity-gtk-module
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
INSTANCE=
```

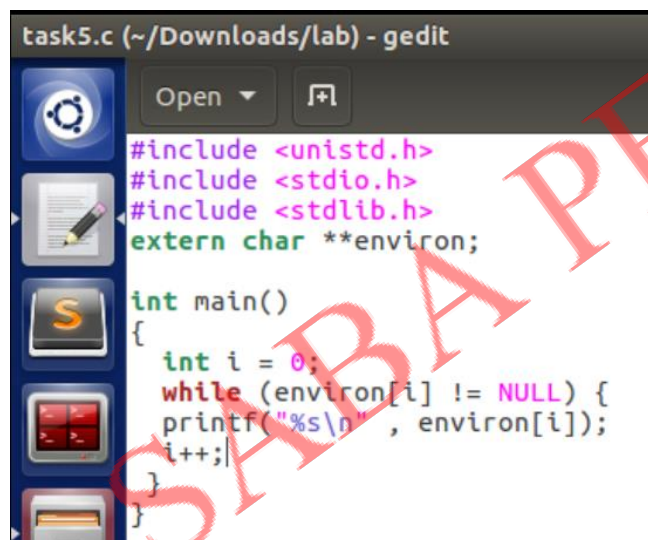
```
Terminal
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
Search your computer: /run/user/1000/keyring/ssh
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1342
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed/Downloads/lab
JAVA_HOME=/usr/lib/jvm/java-8-oracle
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share:/var/lib/snapd/desktop
VTE_VERSION=4205
JOB=dbus
[02/19/24]seed@VM:~/.../lab$
```

From the screenshot, the program prints all environment variables under `/usr/bin/env`. Therefore the `system()` function has passed environment variables to `/bin/sh`.

Task 5: Environment variable and Set-UID Programs

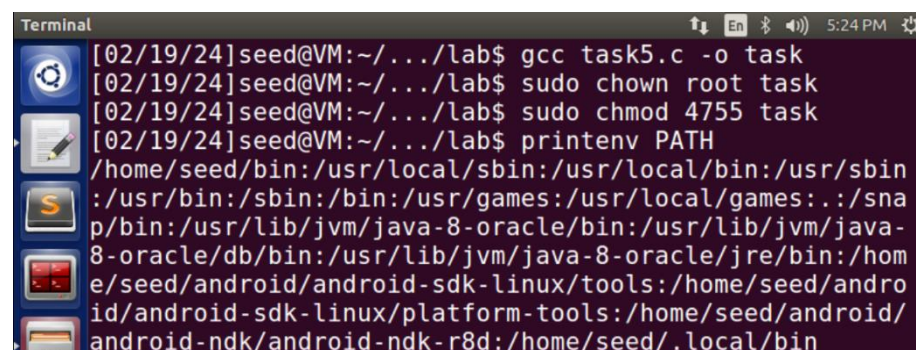
Set-UID is an important security mechanism in Unix operating systems. When a Set-UID program runs, it assumes the owner's privileges. For example, if the program's owner is root, then when anyone runs this program, the program gains the root's privileges during its execution. Set-UID allows us to do many interesting things, but it escalates the user's privilege when executed, making it quite risky. Although the behaviors of Set-UID programs are decided by their program logic, not by users, users can indeed affect the behaviors via environment variables. To understand how Set-UID programs are affected, let us first figure out whether environment variables are inherited by the Set-UID program's process from the user's process.

Code used for this task:



```
task5.c (~/Downloads/lab) - gedit
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
extern char **environ;

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



```
Terminal
[02/19/24]seed@VM:~/.../lab$ gcc task5.c -o task
[02/19/24]seed@VM:~/.../lab$ sudo chown root task
[02/19/24]seed@VM:~/.../lab$ sudo chmod 4755 task
[02/19/24]seed@VM:~/.../lab$ 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/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
```



```
[02/19/24]seed@VM:~/.../lab$ printenv LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/sou
rce/boost_1_64_0/stage/lib:
[02/19/24]seed@VM:~/.../lab$ export COLOR=red
[02/19/24]seed@VM:~/.../lab$ printenv COLOR
red
[02/19/24]seed@VM:~/.../lab$
[02/19/24]seed@VM:~/.../lab$ ./task | grep -i PATH
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Sessio
n0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
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/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
[02/19/24]seed@VM:~/.../lab$
```

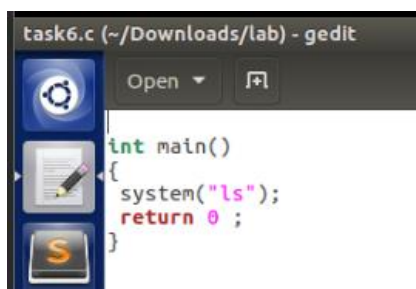
```
[02/19/24]seed@VM:~/.../lab$ ./task | grep -i LD_LIBRAR
Y_PATH
[02/19/24]seed@VM:~/.../lab$ ./task | grep -i COLOR
TERM=xterm-256color
COLOR=red
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;3
5:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su
=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=
01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*
.taz=01;31:*.lha=01;31:*.lz4=01;31:*.lzh=01;31:*.lzma=0
1;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01;31:*
.zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*
.lrz=01;31:*.lz=01;31:*.lzo=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:*.s
ar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;3
1:*.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=0
1;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*
```

As shown in the screenshot, the `Path` is set to `/bin:/home/bin:/usr/bin`, but `LD_LIBRARY_PATH` is not printed, which means it didn't enter the child process.

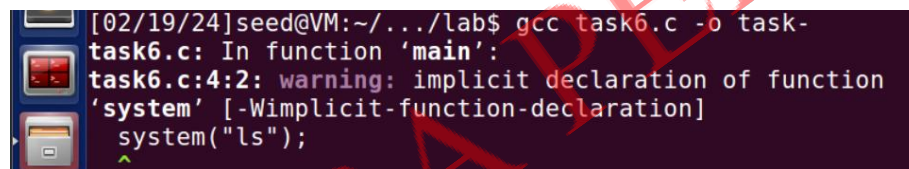
Task 6: The PATH Environment variable and Set-UID Programs

Because of the shell program invoked, calling `system()` within a Set-UID program is quite dangerous. This is because the actual behavior of the shell program can be affected by environment variables, such as `PATH`; these environment variables are provided by the user, who may be malicious. By changing these variables, malicious users can control the behavior of the Set-UID program.

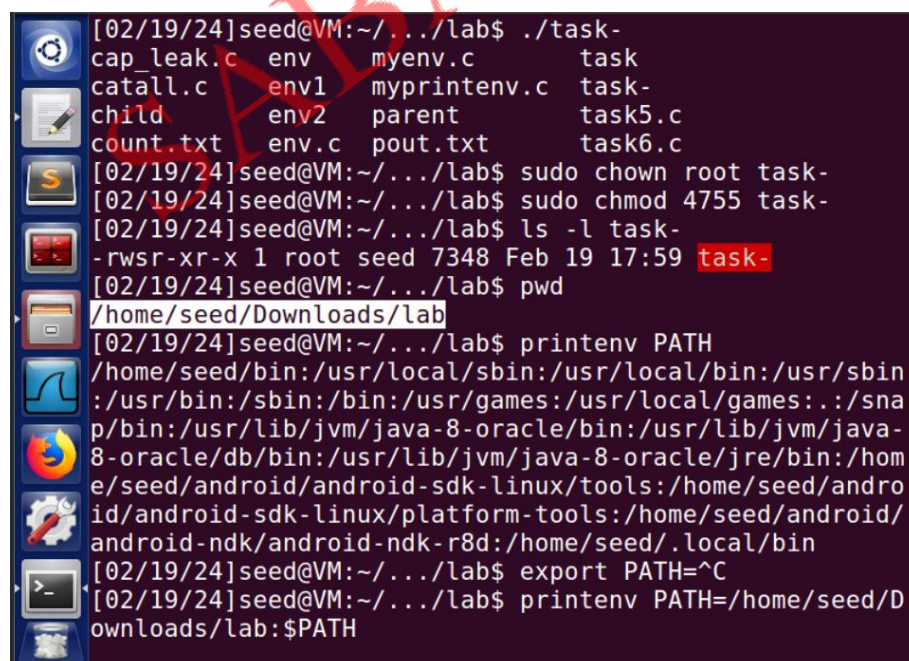
Code used for this task:



```
task6.c (~/Downloads/lab) - gedit
int main()
{
    system("ls");
    return 0;
}
```



```
[02/19/24]seed@VM:~/.../lab$ gcc task6.c -o task-
task6.c: In function 'main':
task6.c:4:2: warning: implicit declaration of function
'system' [-Wimplicit-function-declaration]
    system("ls");
    ^
```



```
[02/19/24]seed@VM:~/.../lab$ ./task-
cap leak.c env myenv.c task
catall.c env1 myprintenv.c task-
child env2 parent task5.c
count.txt env.c pout.txt task6.c
[02/19/24]seed@VM:~/.../lab$ sudo chown root task-
[02/19/24]seed@VM:~/.../lab$ sudo chmod 4755 task-
[02/19/24]seed@VM:~/.../lab$ ls -l task-
-rwsr-xr-x 1 root seed 7348 Feb 19 17:59 task-
[02/19/24]seed@VM:~/.../lab$ pwd
/home/seed/Downloads/lab
[02/19/24]seed@VM:~/.../lab$ 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/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[02/19/24]seed@VM:~/.../lab$ export PATH=^C
[02/19/24]seed@VM:~/.../lab$ printenv PATH=/home/seed/Downloads/lab:$PATH
```



```

[02/19/24]seed@VM:~/.../lab$ task-
cap_leak.c  env    myenv.c    task-
catall.c    env1   myprintenv.c task-
child       env2   parent    task5.c
count.txt   env.c  pout.txt  task6.c
[02/19/24]seed@VM:~/.../lab$ ls
cap_leak.c  env    myenv.c    task-
catall.c    env1   myprintenv.c task-
child       env2   parent    task5.c
count.txt   env.c  pout.txt  task6.c
[02/19/24]seed@VM:~/.../lab$

```

```

Terminal
[02/19/24]seed@VM:~/.../lab$ task-
cap_leak.c  count.txt  env2    myprintenv.c  task-
catall.c    env        env.c   parent        task5.c
child       env1       myenv.c pout.txt      task6.c
[02/19/24]seed@VM:~/.../lab$ ls
cap_leak.c  count.txt  env2    myprintenv.c  task-
catall.c    env        env.c   parent        task5.c
child       env1       myenv.c pout.txt      task6.c
[02/19/24]seed@VM:~/.../lab$

```

The program can gain root privilege if we copy `\bin\sh` to current directory and add current directory to `PATH`.

Task 7: The LD_PRELOAD environment variable and Set-UID Programs

In this task, we study how Set-UID programs deal with some of the environment variables. Several environment variables, including `LD_PRELOAD`, `LD_LIBRARY_PATH`, and other `LD_*` influence the behavior of dynamic loader/linker. A dynamic loader/linker is the part of an operating system (OS) that loads (from persistent storage to RAM) and links the shared libraries needed by an executable at run time.

In Linux, `ld.so` or `ld-linux.so`, are the dynamic loader/linker. Among the environment variables that affect their behaviors, `LD_LIBRARY_PATH` and `LD_PRELOAD` are the two that we are concerned in this lab. In

Linux, `LD_LIBRARY_PATH` is a colon-separated set of directories where libraries should be searched for first, before the standard set of directories. `LD_PRELOAD` specifies a list of additional, user-specified, shared libraries to be loaded before all others. In this task, we will only study `LD_PRELOAD`.

Code used for this task:

```
mylib.c (~/.Downloads/lab) - gedit
#include <stdio.h>
void sleep()
{
    printf("i am not sleeping!\n ");
}
```

```
myprog.c (~/.Downloads/lab) - gedit
#include <unistd.h>
int main()
{
    sleep(1);
    return 0 ;
}
```

```
Terminal
[02/19/24]seed@VM:~/.../lab$ gcc -fPIC -g -c mylib.c
[02/19/24]seed@VM:~/.../lab$ gcc -shared -o libmylib.so
1.0.1 mylib.o -lc
[02/19/24]seed@VM:~/.../lab$ export LD_PRELOAD=./libmyl
ib.so.1.0.1
[02/19/24]seed@VM:~/.../lab$ gcc myprog.c -o myprog
[02/19/24]seed@VM:~/.../lab$ ./myprog
i am not sleeping!
```

```
root@VM: /home/seed/Downloads/lab
[02/19/24]seed@VM:~/.../lab$ gcc -fPIC -g -c mylib.c
[02/19/24]seed@VM:~/.../lab$ gcc -shared -o libmylib.so
1.0.1 mylib.o -lc
[02/19/24]seed@VM:~/.../lab$ export LD_PRELOAD=./libmyl
ib.so.1.0.1
[02/19/24]seed@VM:~/.../lab$ gcc myprog.c -o myprog
[02/19/24]seed@VM:~/.../lab$ ./myprog
i am not sleeping!
[02/19/24]seed@VM:~/.../lab$ sudo chown root myprog
[02/19/24]seed@VM:~/.../lab$ sudo chmod 4755 myprog
[02/19/24]seed@VM:~/.../lab$ ./myprog
[02/19/24]seed@VM:~/.../lab$ sudo su
root@VM:/home/seed/Downloads/lab# export LD_PRELOAD=./l
ibmylib.so.1.0.1
root@VM:/home/seed/Downloads/lab# ./myprog
i am not sleeping!
root@VM:/home/seed/Downloads/lab# exit
exit
[02/19/24]seed@VM:~/.../lab$ sudo adduser user1
```

```
[02/19/24]seed@VM:~/.../lab$ sudo adduser user1
Adding user `user1' ...
Adding new group `user1' (1001) ...
Adding new user `user1' (1001) with group `user1' ...
Creating home directory `/home/user1' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for user1
Enter the new value, or press ENTER for the default
    Full Name []: user1
    Room Number []:
    Work Phone []:
    Home Phone []:
    Other []:
Is the information correct? [Y/n] y
[02/19/24]seed@VM:~/.../lab$
```

```
root@VM: /home/seed/Downloads/lab
[02/19/24]seed@VM:~/.../lab$ sudo chown user1 myprog
[02/19/24]seed@VM:~/.../lab$ sudo chmod 4755 myprog
[02/19/24]seed@VM:~/.../lab$ export LD_PRELOAD=./libmylib.so.1.0.1
[02/19/24]seed@VM:~/.../lab$ ./myprog
[02/19/24]seed@VM:~/.../lab$
```

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

The program will use the environment variable set by user and call the `sleep()` in `libmylib.so.1.0.1`.

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

In this case, the program will ignore `LD_PRELOAD` set by user and use the default `sleep()` function.

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

In this case, exported `LD_PRELOAD` dominates. The program will use the `sleep()` in `libmylib.so.1.0.1`.

Make `myprog` a Set-UID user1 program (i.e., the owner is user1, which is another user account), export the

`LD_PRELOAD` environment variable again in a different user's account (not-root user) and run it.

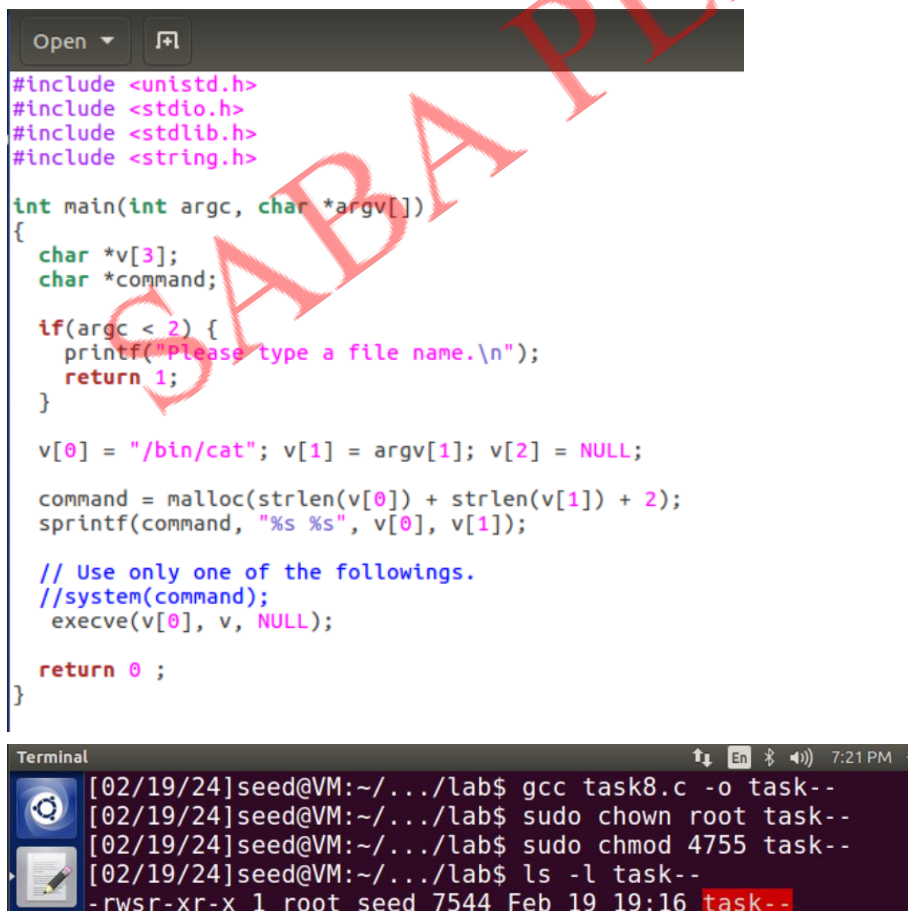
In this case, `LD_PRELOAD` is not overwritten.

In conclusion, only the program owner can run the program with overwritten environment variables.

Task 8: Invoking external program using `system()` versus `execve()`

Although `system()` and `execve()` can both be used to run new programs, `system()` is quite dangerous if used in a privileged program, such as Set-UID programs. We have seen how the `PATH` environment variable affect the behavior of `system()`, because the variable affects how the shell works. `execve()` does not have the problem, because it does not invoke shell. Invoking shell has another dangerous consequence, and this time, it has nothing to do with environment variables.

Code used for this task:



```
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.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
[02/19/24]seed@VM:~/.../lab$ gcc task8.c -o task--
[02/19/24]seed@VM:~/.../lab$ sudo chown root task--
[02/19/24]seed@VM:~/.../lab$ sudo chmod 4755 task--
[02/19/24]seed@VM:~/.../lab$ ls -l task--
-rwsr-xr-x 1 root seed 7544 Feb 19 19:16 task--
```



```
task8.c (~/Downloads/lab) - gedit
Open  [icon]

#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main(int argc, char *argv[])
{
    char *v[3];
    char *command;

    if(argc < 2) {
        printf("Please type a file name.\n");
        return 1;
    }

    v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;

    command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
    sprintf(command, "%s %s", v[0], v[1]);

    // Use only one of the followings.
    system(command);
    // execve(v[0], v, NULL);

    return 0 ;
}
```

```
[02/19/24]seed@VM:~/.../lab$ ls
cap_leak.c  env2      mylib.o      task--
child       env.c     myprintenv.c task5.c
count.txt   libmylib.so.1.0.1 myprog.c     task6.c
env         myenv.c   parent       task8.c
env1        mylib.c   pout.txt

[02/19/24]seed@VM:~/.../lab$ gcc task8.c -o task1
[02/19/24]seed@VM:~/.../lab$ sudo chown root task1
[02/19/24]seed@VM:~/.../lab$ sudo chmod 4755 task1
[02/19/24]seed@VM:~/.../lab$ ./task1 task8.txt
task8 executing

[02/19/24]seed@VM:~/.../lab$
```

we can insert a command after `;` to modify protected le.
However, in second scenario, `execve()` sees the argument as a whole name so we cannot make exploit on that.

Task 9: Capability Leaking

To follow the Principle of Least Privilege, Set-UID programs often permanently relinquish their root privileges if such privileges are not needed anymore. Moreover, sometimes, the program needs to hand over its control to the user; in this case, root privileges must be revoked. The `setuid()` system call can be used to revoke the privileges. According to the manual, "`setuid()` sets the effective user ID of the calling process. If the effective UID of the caller is root, the real UID and saved set-user-ID are also set". Therefore, if a Set-UID program with effective UID 0 calls `setuid(n)`, the process will become a normal process, with all its UIDs being set .

When revoking the privilege, one of the common mistakes is capability leaking. The process may have gained some privileged capabilities when it was still privileged; when the privileged is downgraded, if the program does not clean up those capabilities, they may still be accessible by the non-privileged process. In other words, although the effective user ID of the process becomes non-privileged, the process is still privileged because it possesses privileged capabilities.

```
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>

void main()
{
    int fd;
    char *v[2];

    /* Assume that /etc/zxx is an important system file,
     * and it is owned by root with permission 0644.
     * Before running this program, you should create
     * the file /etc/zxx first. */
    fd = open("/etc/zxx", O_RDWR | O_APPEND);
    if (fd == -1) {
        printf("Cannot open /etc/zxx\n");
        exit(0);
    }

    // Print out the file descriptor value
    printf("fd is %d\n", fd);

    // Permanently disable the privilege by making the
    // effective uid the same as the real uid
    setuid(getuid());

    // Execute /bin/sh
    v[0] = "/bin/sh"; v[1] = 0;
    execve(v[0], v, 0);
}
```

```
[02/19/24]seed@VM:~/.../lab$ ls /etc/zzz
ls: cannot access '/etc/zzz': No such file or directory
[02/19/24]seed@VM:~/.../lab$
[02/19/24]seed@VM:~/.../lab$
[02/19/24]seed@VM:~/.../lab$
[02/19/24]seed@VM:~/.../lab$ sudo gedit /etc/zzz
(gedit:6336): Gtk-WARNING **: Calling Inhibit failed: G
DBus.Error:org.freedesktop.DBus.Error.ServiceUnknown: T
he name org.gnome.SessionManager was not provided by any
.service files
** (gedit:6336): WARNING **: Set document metadata fail
ed: Setting attribute metadata::gedit-spell-enabled not
supported
** (gedit:6336): WARNING **: Set document metadata fail
ed: Setting attribute metadata::gedit-encoding not supp
```

```
[02/19/24]seed@VM:~/.../lab$ ls -l /etc/zzz
-rw-r--r-- 1 root root 18 Feb 19 19:45 /etc/zzz
[02/19/24]seed@VM:~/.../lab$ sudo chmod 0644 /etc/zzz
[02/19/24]seed@VM:~/.../lab$ gcc cap_leak.c -o capleak
[02/19/24]seed@VM:~/.../lab$ sudo chown root:root capleak
```

```
[02/19/24]seed@VM:~/.../lab$ sudo chmod +s capleak
[02/19/24]seed@VM:~/.../lab$ ls -l /etc/zzz
-rw-r--r-- 1 root root 18 Feb 19 19:45 /etc/zzz
[02/19/24]seed@VM:~/.../lab$ ./capleak
fd is 3
$ echo "algo" >&3
$ cat /etc/zzz
text of /etc/zzz
algo
$ echo "important data" >&3
$ cat /etc/zzz
text of /etc/zzz
algo
important data
$
```

From the screenshot above, we can see that the `le` has been modified. This is because `zzz` is opened before `setuid()` and has root privilege.