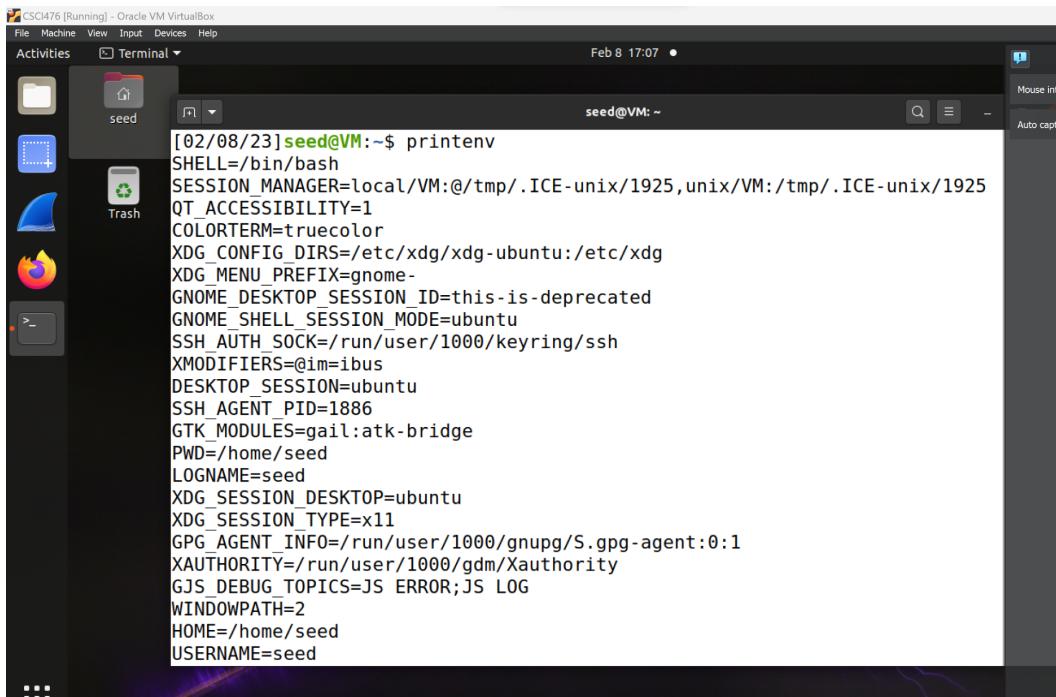
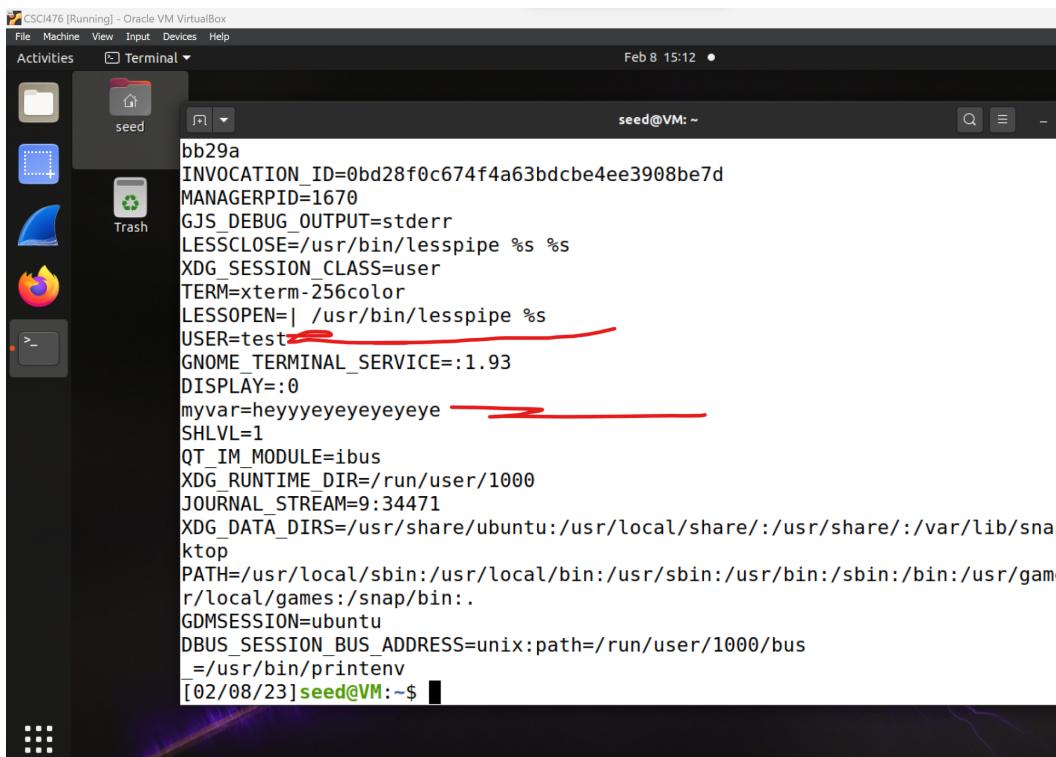


Task1: use printenv



```
[02/08/23]seed@VM:~$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1925,unix/VM:/tmp/.ICE-unix/1925
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1886
GTK_MODULES=gail:atk-bridge
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
WINDOWPATH=2
HOME=/home/seed
USERNAME=seed
```

Task 1.2: environment variables



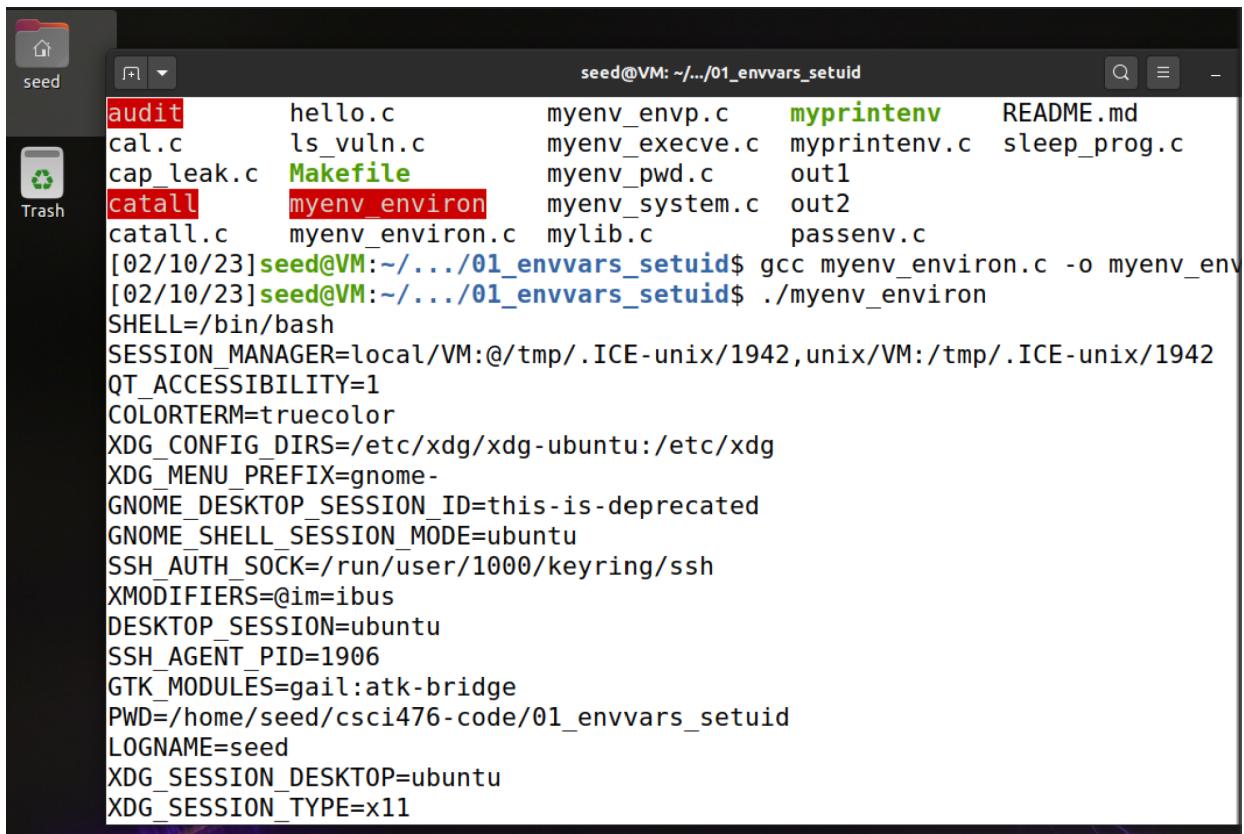
```
bb29a
INVOCATION_ID=0bd28f0c674f4a63bdcbe4ee3908be7d
MANAGERPID=1670
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=test
GNOME_TERMINAL_SERVICE=:1.93
DISPLAY=:0
myvar=heyyyyeyeyeyeye
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:34471
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/db
ktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
/usr/bin/printenv
[02/08/23]seed@VM:~$
```

Tasks 2.1- 2.3:

```
[02/08/23]seed@VM:~/.../01_envvars_setuid$ vim myprintenv.c
[02/08/23]seed@VM:~/.../01_envvars_setuid$ gcc myprintenv.c -o myprintenv
[02/08/23]seed@VM:~/.../01_envvars_setuid$ ./myprintenv > out1
[02/08/23]seed@VM:~/.../01_envvars_setuid$ ls
audit      hello.c          myenv_envp.c    mylib.c      passenv.c
cal.c       ls_vuln.c        myenv_execve.c  myprintenv   README.md
cap_leak.c  Makefile         myenv_pwd.c    myprintenv.c sleep_prog.c
catall.c    myenv_environ.c myenv_system.c out1
[02/08/23]seed@VM:~/.../01_envvars_setuid$ vim out1
[02/08/23]seed@VM:~/.../01_envvars_setuid$ vim myprintenv.c
[02/08/23]seed@VM:~/.../01_envvars_setuid$ ./myprintenv > out2
[02/08/23]seed@VM:~/.../01_envvars_setuid$ diff out1 out2
[02/08/23]seed@VM:~/.../01_envvars_setuid$
```

There is no difference between the environment variables between the parent and the child forks, therefore, the child and the parent are inheriting the same environment variables.

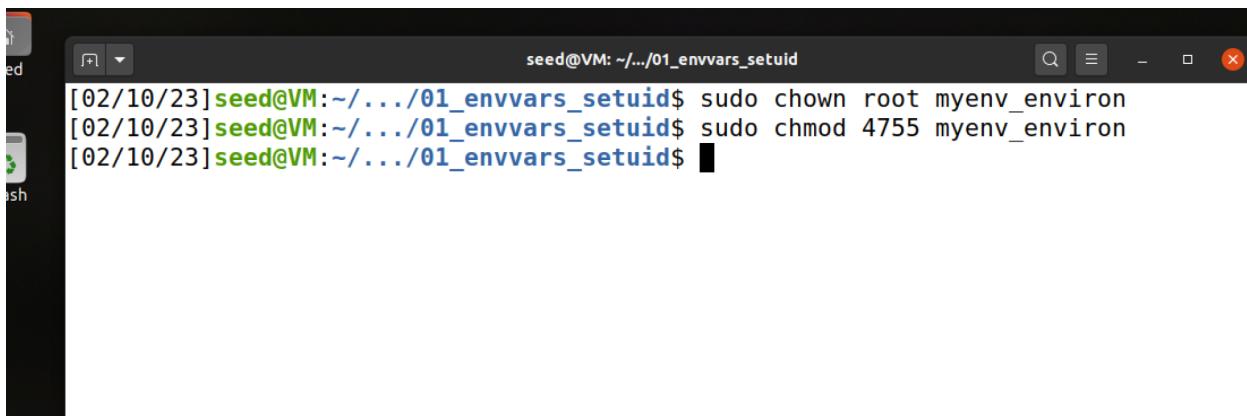
Tasks 3.1: Verify that your implementation correctly prints the environment variables.



The screenshot shows a terminal window titled "seed@VM: ~/.../01_envvars_setuid". The window displays the output of the "myenv_environ" program, which prints all environment variables. The output includes standard variables like PATH, HOME, and USER, along with many system-specific variables such as SESSION_MANAGER, QT_ACCESSIBILITY, COLORTERM, XDG_CONFIG_DIRS, XDG_MENU_PREFIX, GNOME_DESKTOP_SESSION_ID, GNOME_SHELL_SESSION_MODE, SSH_AUTH_SOCK, XMODIFIERS, DESKTOP_SESSION, SSH_AGENT_PID, GTK_MODULES, PWD, LOGNAME, XDG_SESSION_DESKTOP, and XDG_SESSION_TYPE. The terminal interface includes a file browser sidebar on the left and a standard Linux-style title bar at the top.

```
audit      hello.c          myenv_envp.c    mylib.c      passenv.c
cal.c       ls_vuln.c        myenv_execve.c  myprintenv   README.md
cap_leak.c  Makefile         myenv_pwd.c    myprintenv.c sleep_prog.c
catall.c    myenv_environ.c myenv_system.c out1
[02/10/23]seed@VM:~/.../01_envvars_setuid$ gcc myenv_environ.c -o myenv_environ
[02/10/23]seed@VM:~/.../01_envvars_setuid$ ./myenv_environ
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1942,unix/VM:/tmp/.ICE-unix/1942
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1906
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/csci476-code/01_envvars_setuid
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
```

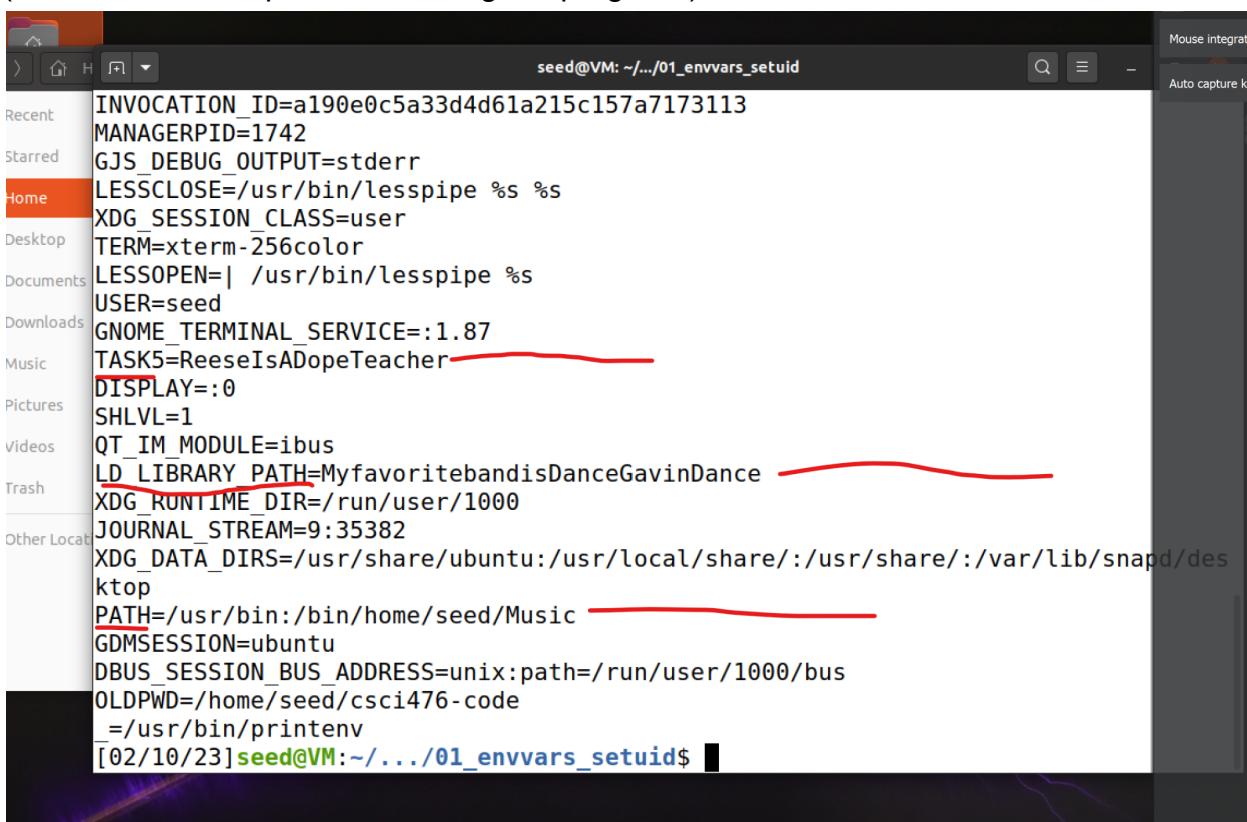
3.2: Compile the above program, change its ownership to root, and make it a Set-UID program.



```
seed@VM:~/.../01_envvars_setuid$ sudo chown root myenv_environ
[02/10/23]seed@VM:~/.../01_envvars_setuid$ sudo chmod 4755 myenv_environ
[02/10/23]seed@VM:~/.../01_envvars_setuid$
```

Task 3.3: In your shell (you need to be in a normal user account, not the root account), use the export command to set the following environment variables

Here is proof that I was able to set the specified environment variables to what I'd like.
(This is not the output from running the program.)



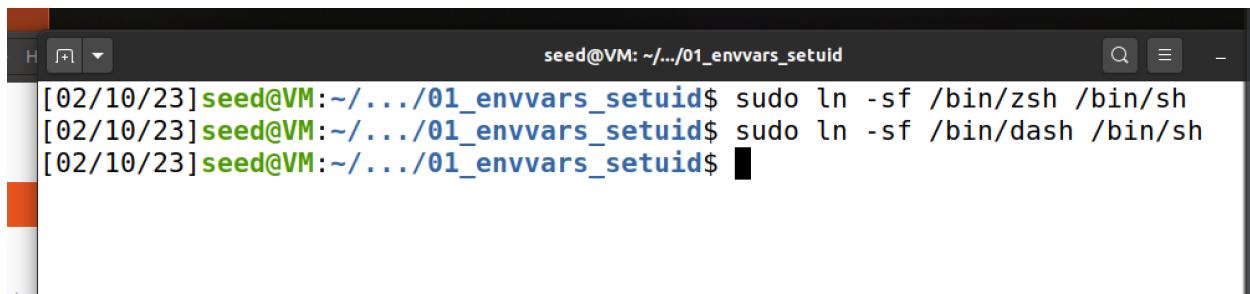
```
seed@VM: ~/.../01_envvars_setuid$ INVOCATION_ID=a190e0c5a33d4d61a215c157a7173113
MANAGERPID=1742
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.87
TASK5=ReeseIsADopeTeacher
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
LD_LIBRARY_PATH=MyfavoritebandisDanceGavinDance
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:35382
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share:/usr/share:/var/lib/snapd/desktop
PATH=/usr/bin:/bin/home/seed/Music
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/seed/csci476-code
/usr/bin/printenv
[02/10/23]seed@VM:~/.../01_envvars_setuid$
```

After running the program:

```
TASK5=ReeseIsADopeTeacher
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
LD_LIBRARY_PATH=MyfavoritebandisDanceGavinDance
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:35382
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/c
ktop
PATH=/usr/bin:/bin/home/seed/Music
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/seed/csci476-code
./myenv_environ
[02/10/23]seed@VM:~/.../01_envvars_setuid$
```

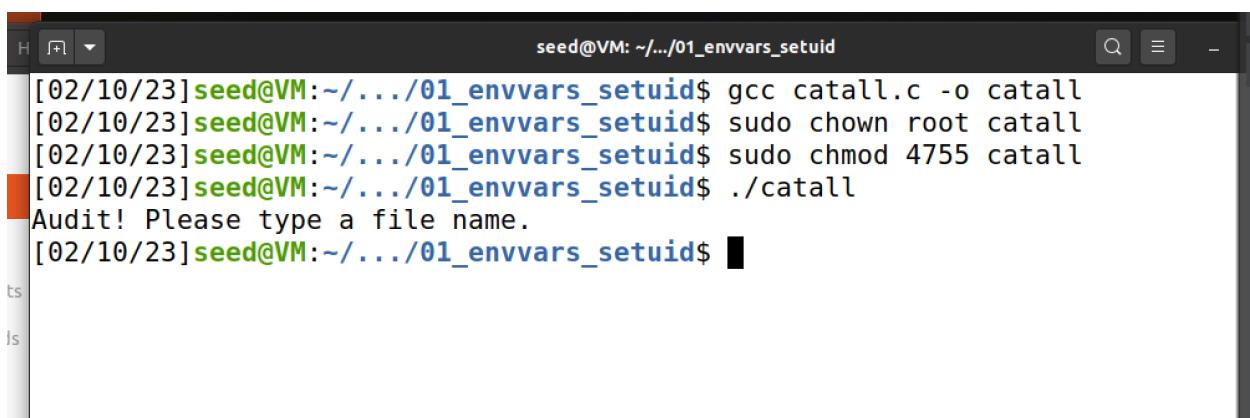
The variables **PATH**, **LD_LIBARARY_PATH**, and **TASK5** are present in the child fork process; I don't think there is anything too surprising about this.

4.0 : disabling counter measures



```
seed@VM: ~/.../01_envvars_setuid
[02/10/23]seed@VM:~/.../01_envvars_setuid$ sudo ln -sf /bin/zsh /bin/sh
[02/10/23]seed@VM:~/.../01_envvars_setuid$ sudo ln -sf /bin/dash /bin/sh
[02/10/23]seed@VM:~/.../01_envvars_setuid$
```

Task 4.1 :



```
seed@VM: ~/.../01_envvars_setuid
[02/10/23]seed@VM:~/.../01_envvars_setuid$ gcc catall.c -o catall
[02/10/23]seed@VM:~/.../01_envvars_setuid$ sudo chown root catall
[02/10/23]seed@VM:~/.../01_envvars_setuid$ sudo chmod 4755 catall
[02/10/23]seed@VM:~/.../01_envvars_setuid$ ./catall
Audit! Please type a file name.
[02/10/23]seed@VM:~/.../01_envvars_setuid$
```

Yes, since the program uses **System(command)**, Bob can abuse this by running malicious lines of code as appended arguments when he tries to read a file. Since the program above has higher privileges as a SET-UID program during execution, and

System() will run with these higher privileges, Bob can trick System() to run these malicious lines of code with its higher privileges.

Task 4.2: Comment out the system(command) statement, and uncomment the execve() statement; the program will use execve() to invoke the command. Compile the program, and make it a root-owned Set-UID.

```
[02/10/23]seed@VM:~/.../01_envvars_setuid$ vim catall.c
[02/10/23]seed@VM:~/.../01_envvars_setuid$ gcc catall.c -o catall
[02/10/23]seed@VM:~/.../01_envvars_setuid$ sudo chown root catall
[02/10/23]seed@VM:~/.../01_envvars_setuid$ sudo chmod 4755 catall
[02/10/23]seed@VM:~/.../01_envvars_setuid$ ./catall
Audit! Please type a file name.
[02/10/23]seed@VM:~/.../01_envvars_setuid$
```

Running the execve() command will not let Bob's attack from task 4.1 pass. This is because execve() replaces the contents of the process and a whole new executable is loaded.

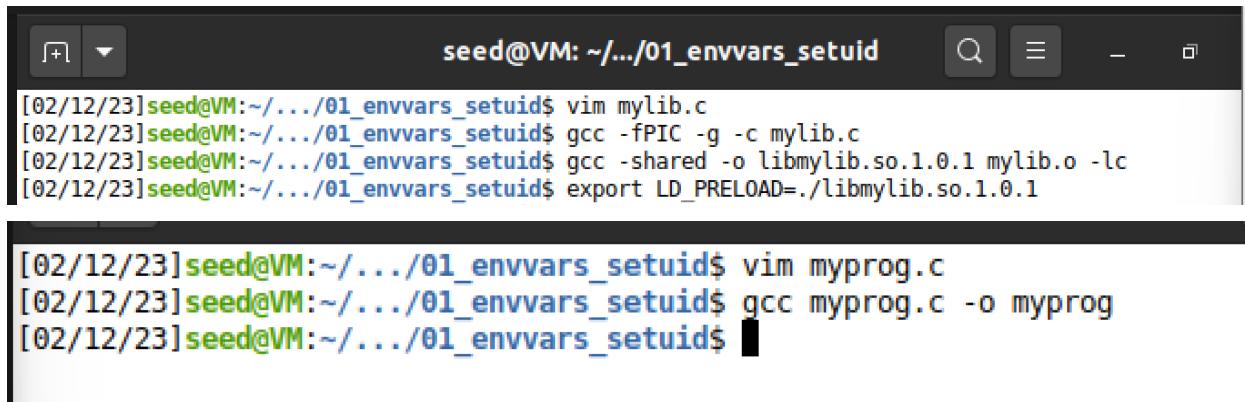
Task 5: PATH and Set-UID Programs

```
[02/12/23]seed@VM:~/.../01_envvars_setuid$ export PATH=/home/seed:$PATH
[02/12/23]seed@VM:~/.../01_envvars_setuid$ echo $SHELL
/bin/bash
[02/12/23]seed@VM:~/.../01_envvars_setuid$ gcc ls_vuln.c -o ls_vuln
[02/12/23]seed@VM:~/.../01_envvars_setuid$ sudo chown root ls_vuln
[02/12/23]seed@VM:~/.../01_envvars_setuid$ sudo chmod 4755 ls_vuln
[02/12/23]seed@VM:~/.../01_envvars_setuid$ ./ls_vuln
audit      hello.c        myenv_environ.c   mylib.c       passenv.c
cal.c       ls_vuln        myenv_envp.c     myprintenv  README.md
cap_leak.c  ls_vuln.c     myenv_execve.c  myprintenv.c sleep_prog.c
catall     Makefile        myenv_pwd.c     out1
catall.c    myenv_environ  myenv_system.c  out2
[02/12/23]seed@VM:~/.../01_envvars_setuid$
```

Yes, we can make this Set-UID program run our code by changing the PATH environment variable to locate a directory we choose, and then naming our own code the same name as ls_vuln. The computer will think our *potentially malicious* code is the safe code it's expecting.

This will also run our code with root privileges as this Set-UID program runs System(), which will give our program temporarily-high privileges.

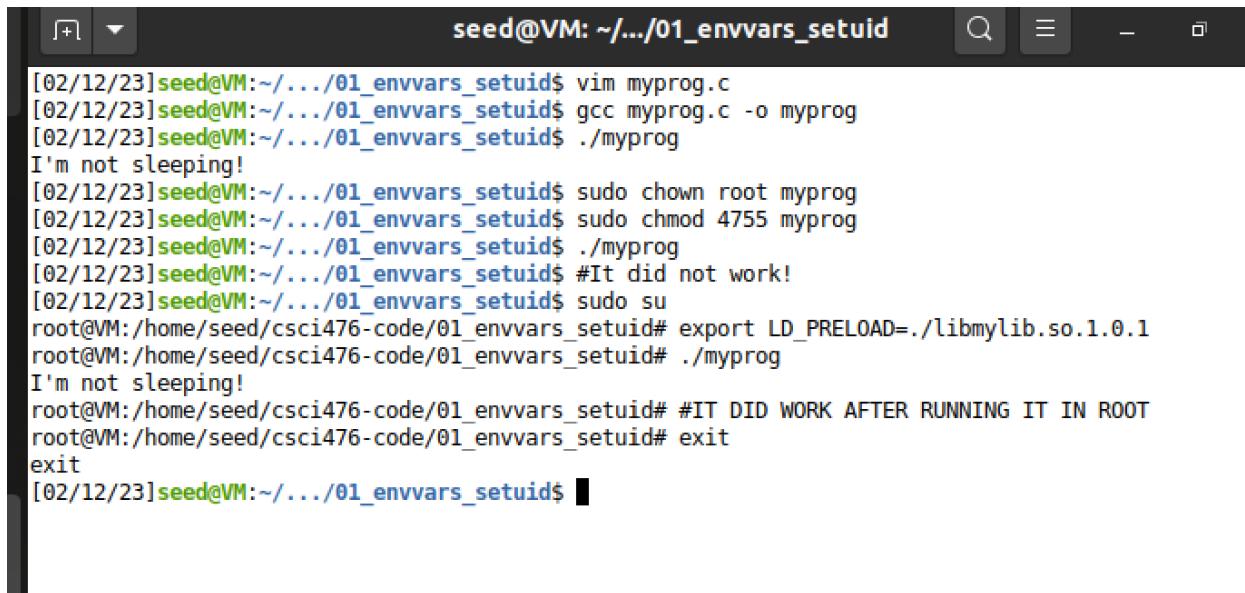
Task 6.1 : LS_PRELOAD and Set-UID Programs



```
seed@VM: ~/.../01_envvars_setuid$ vim mylib.c
[02/12/23]seed@VM:~/.../01_envvars_setuid$ gcc -fPIC -g -c mylib.c
[02/12/23]seed@VM:~/.../01_envvars_setuid$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[02/12/23]seed@VM:~/.../01_envvars_setuid$ export LD_PRELOAD=./libmylib.so.1.0.1

[02/12/23]seed@VM:~/.../01_envvars_setuid$ vim myprog.c
[02/12/23]seed@VM:~/.../01_envvars_setuid$ gcc myprog.c -o myprog
[02/12/23]seed@VM:~/.../01_envvars_setuid$
```

Tasks 6.2: LS_PRELOAD cont'



```
seed@VM: ~/.../01_envvars_setuid$ vim myprog.c
[02/12/23]seed@VM:~/.../01_envvars_setuid$ gcc myprog.c -o myprog
[02/12/23]seed@VM:~/.../01_envvars_setuid$ ./myprog
I'm not sleeping!
[02/12/23]seed@VM:~/.../01_envvars_setuid$ sudo chown root myprog
[02/12/23]seed@VM:~/.../01_envvars_setuid$ sudo chmod 4755 myprog
[02/12/23]seed@VM:~/.../01_envvars_setuid$ ./myprog
[02/12/23]seed@VM:~/.../01_envvars_setuid$ #It did not work!
[02/12/23]seed@VM:~/.../01_envvars_setuid$ sudo su
root@VM:/home/seed/csci476-code/01_envvars_setuid# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/csci476-code/01_envvars_setuid# ./myprog
I'm not sleeping!
root@VM:/home/seed/csci476-code/01_envvars_setuid# #IT DID WORK AFTER RUNNING IT IN ROOT
root@VM:/home/seed/csci476-code/01_envvars_setuid# exit
exit
[02/12/23]seed@VM:~/.../01_envvars_setuid$
```

Task 6.3:

When we ran the **myprog** program, it worked initially. However, when we chained the owner to root and made it a Set-UID program, it no longer worked! Finally, when we entered a higher-privileged shell and *then* ran it again, our **myprog** program worked again.

We can refer back to this slide with similar-yielding results:

| Process Owner | Set-UID program? | Success? |
|---------------|------------------|----------|
| seed | ✗ | ✓ |
| seed | ✓ | ✗ |
| root | ✓ | ✓ |

We can say our failure in the seed-process, Set-UID program was due to the child no longer inheriting the LD_PRELOAD environment variable that we previously edited.