

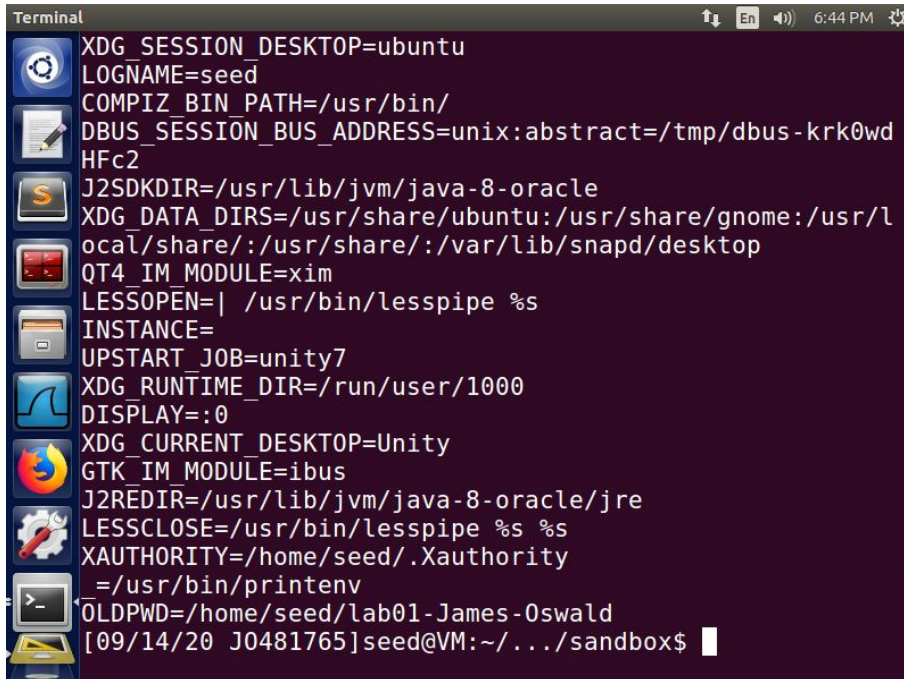
James Oswald

Lab 01

## Task #1: Manipulating Environment Variables

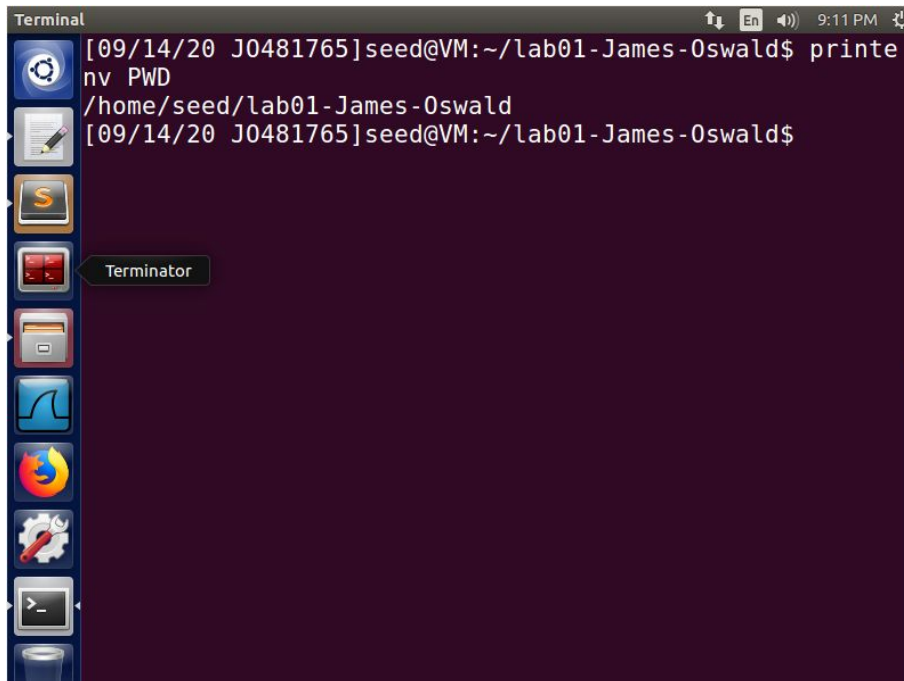
Bullet #1:

Running printenv to get a list of all the environmental variables.

A terminal window titled "Terminal" with a dark purple background and a sidebar of application icons on the left. The terminal displays the output of the 'printenv' command, listing various environment variables such as XDG\_SESSION\_DESKTOP, LOGNAME, COMPIZ\_BIN\_PATH, DBUS\_SESSION\_BUS\_ADDRESS, J2SDKDIR, XDG\_DATA\_DIRS, QT4\_IM\_MODULE, LESSOPEN, INSTANCE, UPSTART\_JOB, XDG\_RUNTIME\_DIR, DISPLAY, XDG\_CURRENT\_DESKTOP, GTK\_IM\_MODULE, J2REDIR, LESSCLOSE, XAUTHORITY, OLDPWD, and the current directory. The prompt shows the user is 'seed' at a VM named 'lab01-James-Oswald' in the directory '~/.../sandbox' at 6:44 PM.

```
Terminal
XDG_SESSION_DESKTOP=ubuntu
LOGNAME=seed
COMPIZ_BIN_PATH=/usr/bin/
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-krk0wdHFc2
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=
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
_=/usr/bin/printenv
OLDPWD=/home/seed/lab01-James-Oswald
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

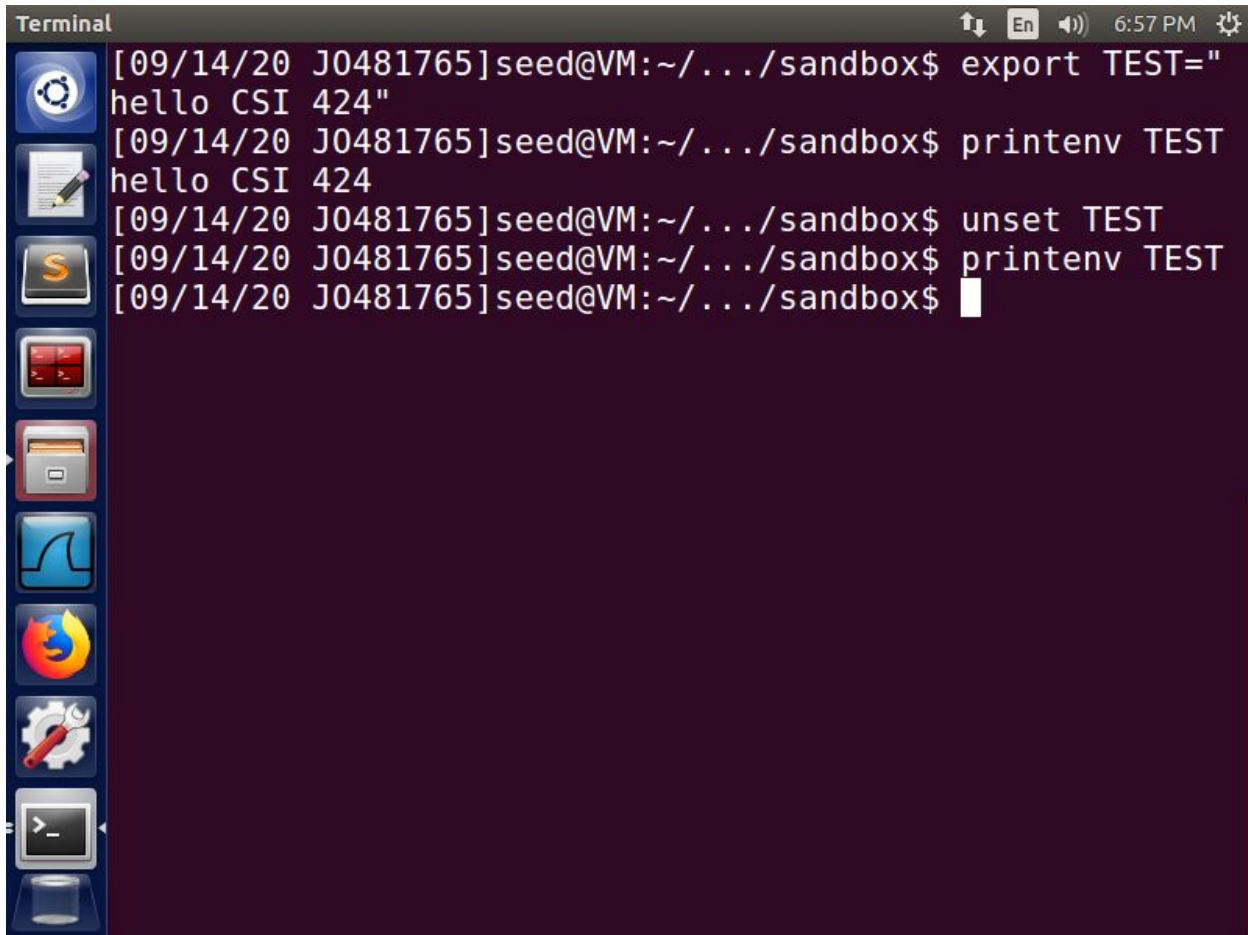
Running printenv PWD to see the value of a single environmental variable

A terminal window titled "Terminal" with a dark purple background and a sidebar of application icons on the left. The terminal shows the user running 'printenv PWD', which outputs the path '/home/seed/lab01-James-Oswald'. The prompt indicates the user is 'seed' at a VM named 'lab01-James-Oswald' in the directory '~/lab01-James-Oswald' at 9:11 PM. A "Terminator" button is visible in the sidebar.

```
Terminal
[09/14/20 J0481765]seed@VM:~/lab01-James-Oswald$ printenv PWD
/home/seed/lab01-James-Oswald
[09/14/20 J0481765]seed@VM:~/lab01-James-Oswald$
```

## Bullet #2:

Testing out using export and unset to set and unset a single custom environmental variable

A terminal window titled "Terminal" with a dark background and a light blue sidebar containing icons for various applications. The terminal shows a series of commands and their outputs. The first command is "export TEST='hello CSI 424'", followed by "printenv TEST" which outputs "hello CSI 424". The third command is "unset TEST", followed by "printenv TEST" which outputs an empty line. The terminal window has a status bar at the top right showing "6:57 PM" and a settings icon.

```
Terminal [09/14/20 J0481765]seed@VM:~/.../sandbox$ export TEST="
hello CSI 424"
[09/14/20 J0481765]seed@VM:~/.../sandbox$ printenv TEST
hello CSI 424
[09/14/20 J0481765]seed@VM:~/.../sandbox$ unset TEST
[09/14/20 J0481765]seed@VM:~/.../sandbox$ printenv TEST
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

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

### Step #1

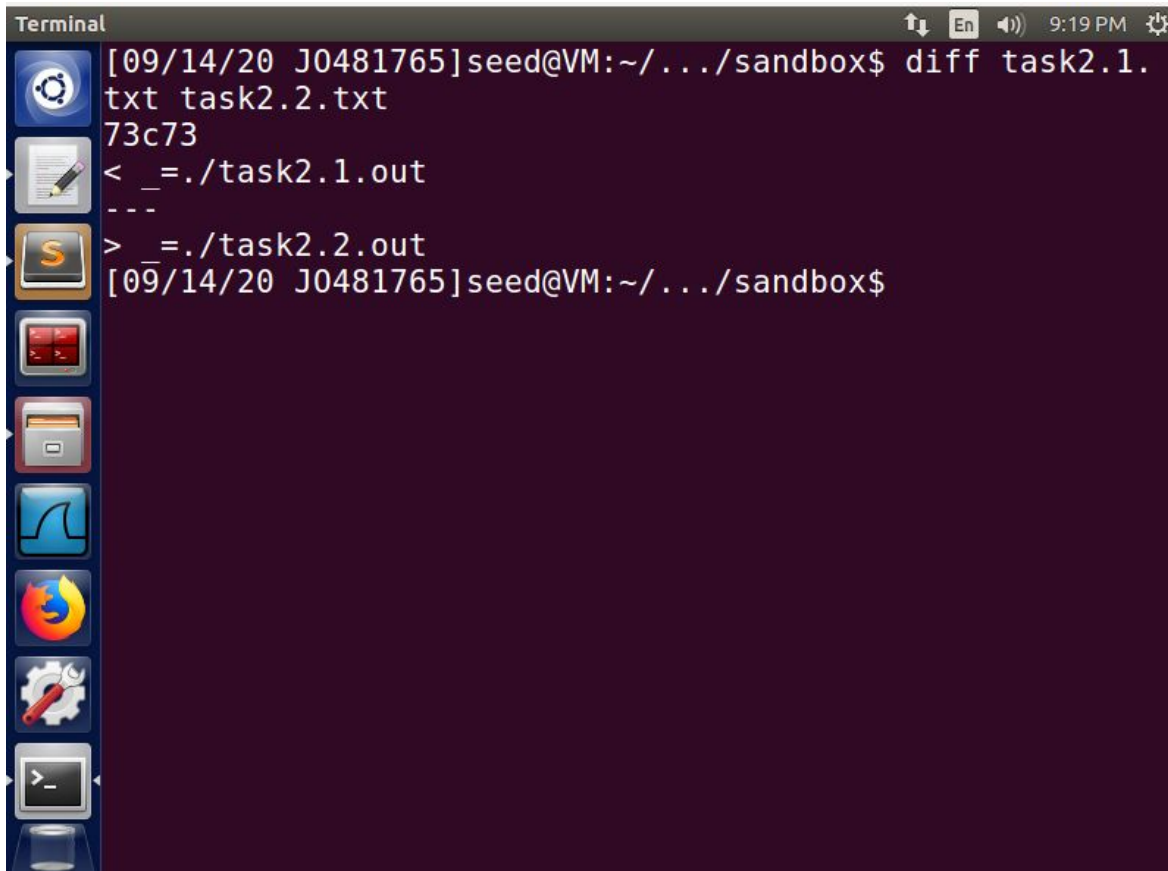
Running the program and looking at the strings produced by the child process in task2.1.txt, I see that the file contains the same things printed by printenv with the exception of \$\_ being the name of the program.

### Step #2

After changing the program and seeing the and looking at the strings produced by the child process in task2.2.txt, I see the same environmental variables with the exception of \$\_.

### Step#3

Running diff on the two output files I see that the only difference is the \_ environmental variable.

A terminal window titled "Terminal" with a dark purple background. The terminal shows the command `diff task2.1.txt task2.2.txt` and its output. The output indicates a difference in the first line, showing the parent process's environment variable `_` pointing to `./task2.1.out` and the child process's `_` pointing to `./task2.2.out`. The terminal also shows the user's prompt and the current directory path `~/.../sandbox`.

```
Terminal [09/14/20  J0481765]seed@VM:~/.../sandbox$ diff task2.1.txt task2.2.txt
73c73
< _=./task2.1.out
---
> _=./task2.2.out
[09/14/20  J0481765]seed@VM:~/.../sandbox$
```

This allows me to draw the conclusion that a child process created with `fork()` will share the same environment variables as its parent.

### Task #3: Environment Variables and `execve()`

#### Step #1:

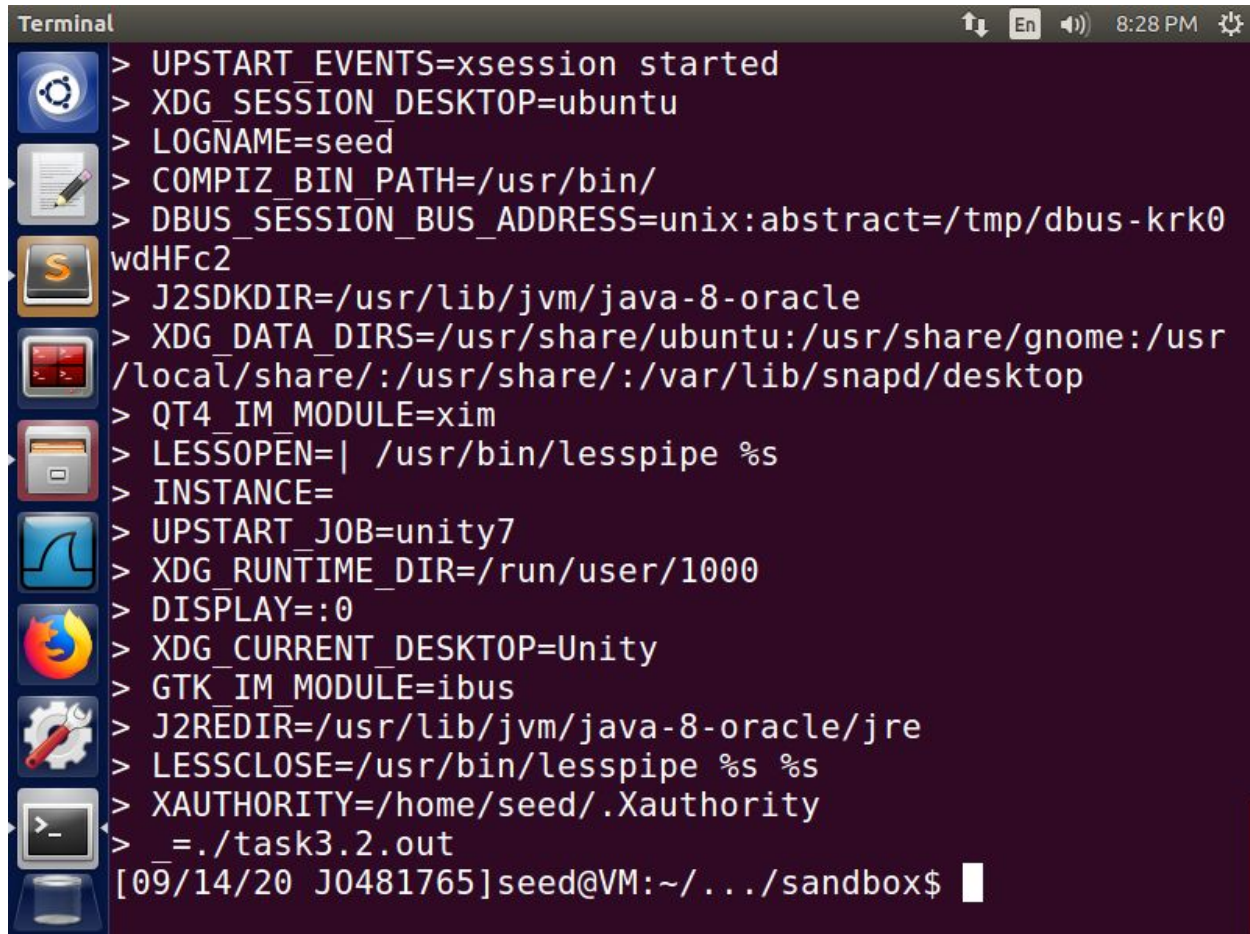
I compiled and ran `task3.1.c` piping the output to `task3.1.txt`. The file was empty, no environmental variables were printed.

#### Step #2:

I made the change to `task3.1.c` passing in `environ` as a parameter to `execve`. I piped the result into `task3.2.txt`. The file was full and had all the environmental variables present.

Step #3:

I ran diff on the output files to confirm and see that indeed only the program from task 2 produced the environmental variables as output.

A terminal window with a dark purple background and a sidebar of application icons on the left. The terminal displays the output of a program, listing various environment variables. The output starts with a prompt character '>' and ends with a shell prompt '[09/14/20 J0481765]seed@VM:~/.../sandbox\$'.

```
> UPSTART_EVENTS=xsession started
> XDG_SESSION_DESKTOP=ubuntu
> LOGNAME=seed
> COMPIZ_BIN_PATH=/usr/bin/
> DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-krk0
wdHFc2
> 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=
> 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
> _=./task3.2.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

From this result the conclusion can be drawn that the new program gets its environmental variables via the environ parameter being passed to it as the third parameter in execve().

#### Task #4: Environment Variables and system()

We easily verify that the system passes in environmental variables by compiling the given program and running it. I took the liberty of piping the result into task4.txt, which contained all the proper environmental variables. But have also ran the program to show this to be the case with a screenshot:



```
Terminal
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
UPSTART_EVENTS=xsession started
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1414
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed/lab01-James-Oswald/sandbox
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=unity-settings-daemon
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

## Task #5: Environment Variable and Set-UID Programs

Step #1:

I wrote the program and saved it as task5.c compiling it to task5.out

Step #2:

I performed the necessary modifications with chown and chmod to make it a Set-UID program

```
Terminal
[09/14/20 J0481765]seed@VM:~/.../sandbox$ gcc -o task5.out task5.c
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chown root task5.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chmod 4755 task5.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

Step#3:

I set the new environmental variables using export and then ran task5.out

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ gcc -o task5.out task5.c
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chown root .
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chmod 4755 .
[09/14/20 J0481765]seed@VM:~/.../sandbox$ export PATH=$PATH:/fake/path/addition
[09/14/20 J0481765]seed@VM:~/.../sandbox$ export LD_LIBRARY_PATH=/fake/path
[09/14/20 J0481765]seed@VM:~/.../sandbox$ export MEMEVAR="This is a joke var"
```

The result was rather surprising, despite being a Set-UID root program, its environmental variables were the ones I had just set using export.

```
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
WINDOWID=60817497
OLDPWD=/home/seed/lab01-James-Oswald
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1414
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
MEMEVAR=This is a joke var
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;44:ex=
```

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

First setting up the code and making its executable a Set-UID program

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ gcc -o task6.out task6.c
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chown root task6.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chmod 4755 task6.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

Now adding my working directory to the first spot in the path using

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ export PATH=$PWD:$PATH
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

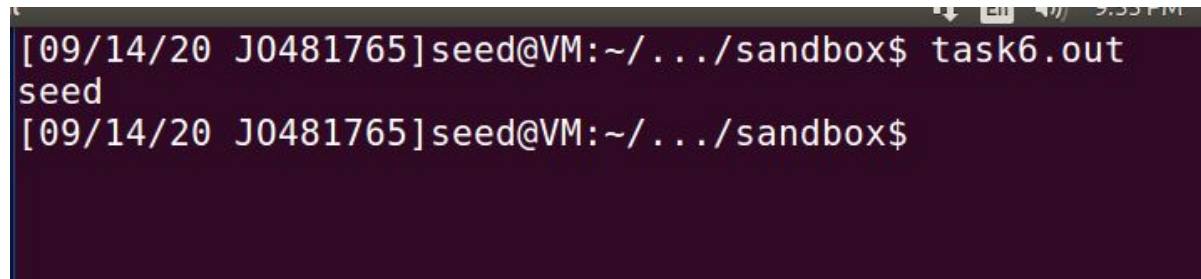
To test if I can run my own program and whether it has root privileges, I write my own program to replace ls



```
task3.1.c x task3.2.c x task4.c x task5.c x task6.c x ls.c x 2.
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main()
5 {
6     system("whoami");
7     return 0;
8 }
9
```

And I compile it as ls

Now running task6.out



```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ task6.out
seed
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

We see that I have successfully run my own code and replaced ls, but it's not running as root, rather it's running as seed and is without root privileges.



## Task 7: The LD\_PRELOAD Environment Variable and Set-UID Programs

### Step #1

Setting LD\_PRELOAD for the seed, saving and compiling the program

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ gcc -fPIC -g -c mylib.c
[09/14/20 J0481765]seed@VM:~/.../sandbox$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/14/20 J0481765]seed@VM:~/.../sandbox$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

### Step #2:

Testing with different users and setUID states. As a normal user with LD\_PRELOAD set, our code runs, when changed to a setUID program our code doesn't run.

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ myprog.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/14/20 J0481765]seed@VM:~/.../sandbox$ myprog.out
I am not sleeping!
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chown root myprog.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chmod 4755 myprog.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ myprog.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

Switching to root and setting LD\_PRELOAD will make our code run.

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo -i
root@VM:~# cd /home/seed/lab01-James-Oswald/sandbox/
root@VM:/home/seed/lab01-James-Oswald/sandbox# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed/lab01-James-Oswald/sandbox# ./myprog.out
I am not sleeping!
root@VM:/home/seed/lab01-James-Oswald/sandbox#
```



Switching the program to be owned by user1 and setting LD\_PRELOAD will not make our code run.

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo -i
root@VM:~# cd /home/seed/lab01-James-Oswald/sandbox/
root@VM:/home/seed/lab01-James-Oswald/sandbox# su user1
-c "export LD_PRELOAD=./libmylib.so.1.0.1"
root@VM:/home/seed/lab01-James-Oswald/sandbox# su user1
-c "myprog.out"
root@VM:/home/seed/lab01-James-Oswald/sandbox#
```

Step #3:

To figure out the cause, i can set up an experiment by printing out whether LD\_PRELOAD has been set. The conclusion of this experiment is that when trying to run code with elevated privileges not as a user whose LD\_PRELOAD has been set, the code won't run.

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

Step #1

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ gcc -o task8.out task8.c
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chmod 4755 task8.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chown root task8.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

If i was bob I would not be able to compromise the system via PATH since the path to /bin/cat is absolute (I tried doing this, the code is in /sandbox/bin/cat.c), however I may be able to take advantage of LD\_PRELOAD to redefine sprintf to store my own custom input in command.

No matter what you do, the child process will inherit the environmental variables via being invoked with system

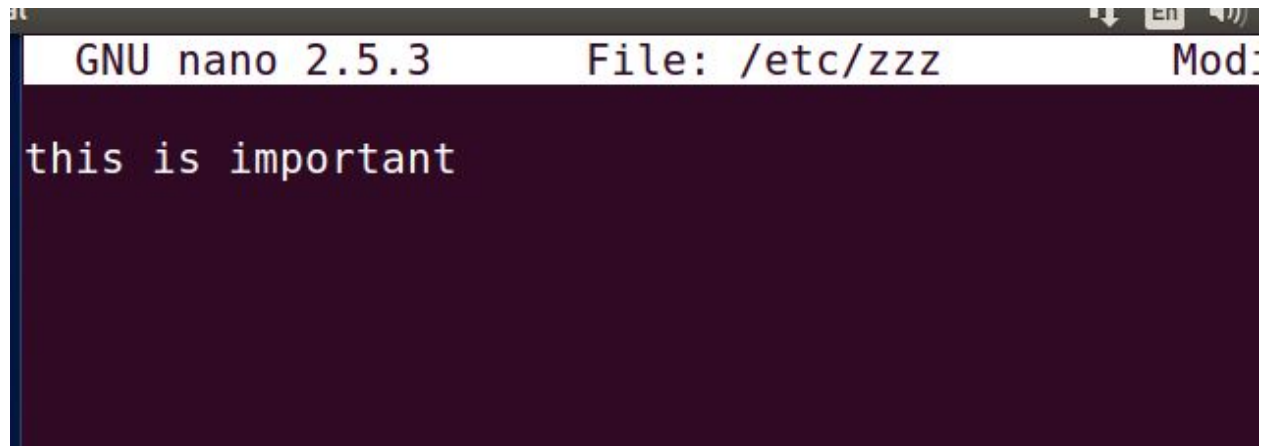
Step 2:

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ gcc -o task8.2.out task8.2.c
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chown root task8.2.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chmod 4755 task8.2.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

Changing the line to be `execve()` with `NULL` passed in as its environmental vars ensures that we won't be able to use them maliciously in the subprocess even if we can change the behavior of the task 8 using them.

Task 9:

Creating the file `/etc/zzz`

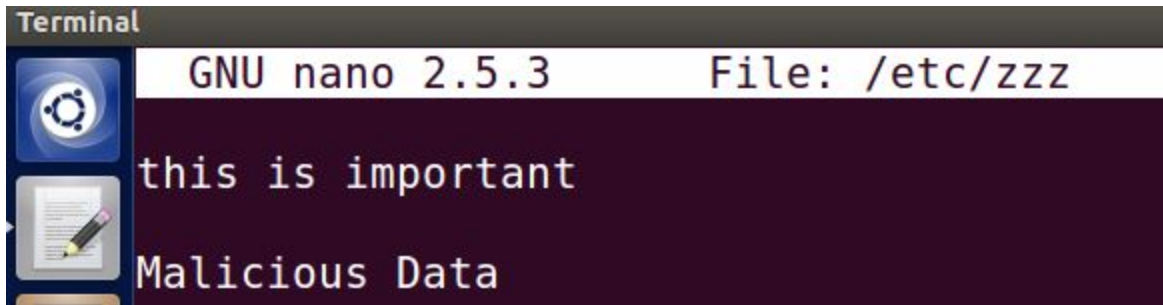


The screenshot shows a terminal window with the GNU nano 2.5.3 text editor open. The title bar indicates the file being edited is `/etc/zzz`. The editor's content area displays the text `this is important` on a single line. The status bar at the bottom shows the file's permissions as `-rwxr-xr-x`.

Making it owned by root and changing it to a Set UID program

```
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chown root task9.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ sudo chmod 4755 task9.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$ task9.out
[09/14/20 J0481765]seed@VM:~/.../sandbox$
```

Sure enough the program was able to insert malicious data into the file.

A terminal window titled "Terminal" showing the GNU nano 2.5.3 text editor. The editor is editing the file "/etc/zzz". The text "this is important" is on the first line, and "Malicious Data" is on the second line. The terminal has a dark background and a light-colored border.

```
Terminal
GNU nano 2.5.3      File: /etc/zzz
this is important
Malicious Data
```

This result is very interesting and rather unexpected. Despite relinquishing root privileges, the process still retains the ability to write to a file it should no longer be able to write to. I suspect this is due to the file descriptor being created while the program still had root privileges.

All code can be found in the sandbox folder