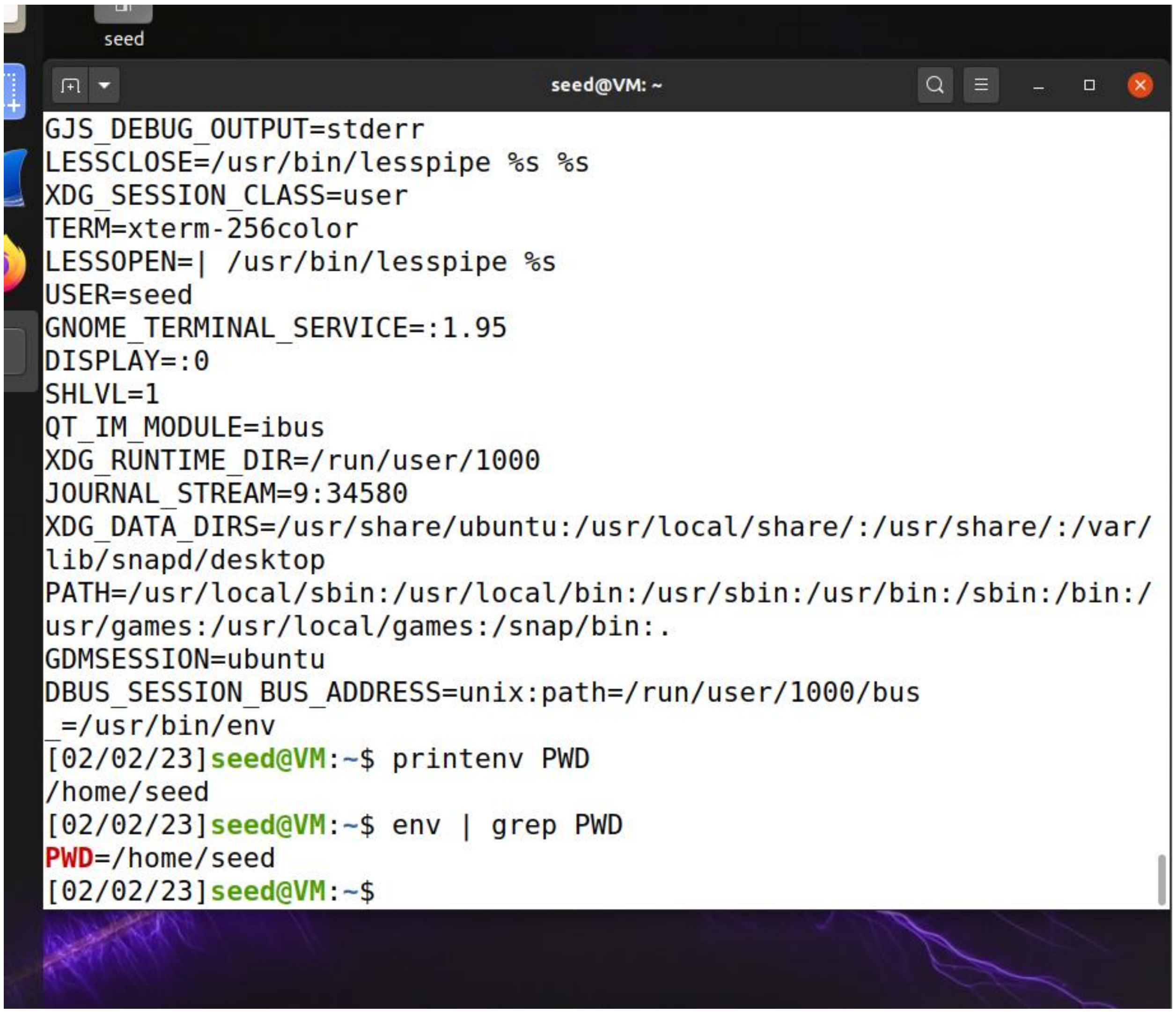
Benjamin Haedt

Lab 1

CSCI 476 – Security

Task 1.1



```
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=:1.95
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:34580
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/
lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/
usr/games:/usr/local/games:/snap/bin:.
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
_=/usr/bin/env
[02/02/23]seed@VM:~$ printenv PWD
/home/seed
[02/02/23]seed@VM:~$ env | grep PWD
PWD=/home/seed
[02/02/23]seed@VM:~$
```

Task 1.2

[02/02/23]seed@VM:~$ env

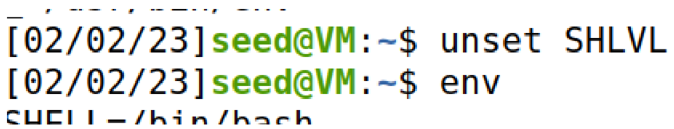
USER=seed

GNOME_TERMINAL_SERVICE=:1.95

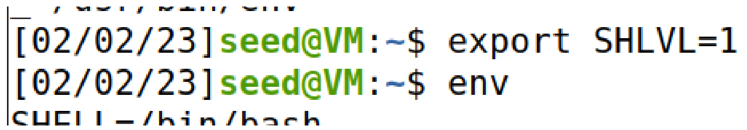
DISPLAY=:0

SHLVL=1

```
QT_IM_MODULE=ibus
[02/02/23]seed@VM:~$ unset SHLVL
[02/02/23]seed@VM:~$ env
GNOME_TERMINAL_SERVICE=:1.95
DISPLAY=:0
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:34580
[02/02/23]seed@VM:~$ export SHLVL=1
[02/02/23]seed@VM:~$ env
GNOME_TERMINAL_SERVICE=:1.95
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:34580
```



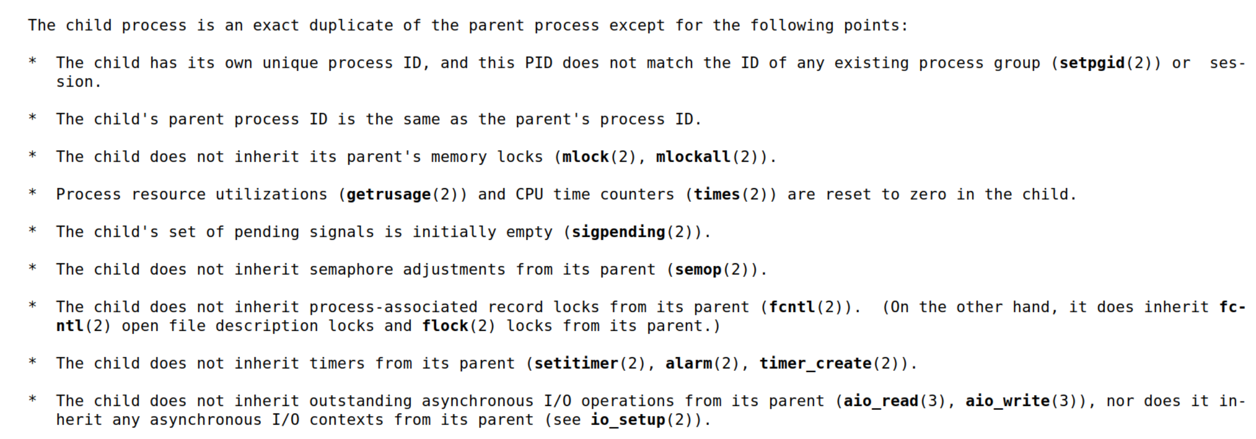
```
[02/02/23]seed@VM:~$ unset SHLVL
[02/02/23]seed@VM:~$ env
```



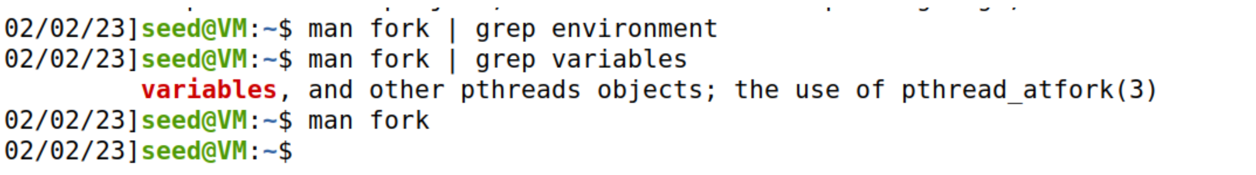
```
[02/02/23]seed@VM:~$ export SHLVL=1
[02/02/23]seed@VM:~$ env
```

Task 2

Yes, the child inherits the environment variables from the parent.

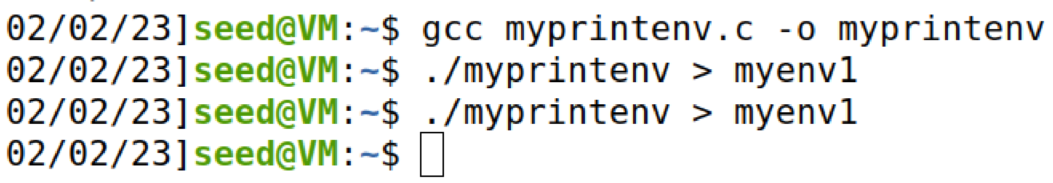


```
The child process is an exact duplicate of the parent process except for the following points:

*  The child has its own unique process ID, and this PID does not match the ID of any existing process group (setpgid(2)) or  ses-
   sion.

*  The child's parent process ID is the same as the parent's process ID.

*  The child does not inherit its parent's memory locks (mlock(2), mlockall(2)).

*  Process resource utilizations (getrusage(2)) and CPU time counters (times(2)) are reset to zero in the child.

*  The child's set of pending signals is initially empty (sigpending(2)).

*  The child does not inherit semaphore adjustments from its parent (semop(2)).

*  The child does not inherit process-associated record locks from its parent (fcntl(2)).  (On the other hand, it does inherit fc-
   ntl(2) open file description locks and flock(2) locks from its parent.)

*  The child does not inherit timers from its parent (setitimer(2), alarm(2), timer_create(2)).

*  The child does not inherit outstanding asynchronous I/O operations from its parent (aio_read(3), aio_write(3)), nor does it in-
   herit any asynchronous I/O contexts from its parent (see io_setup(2)).
```

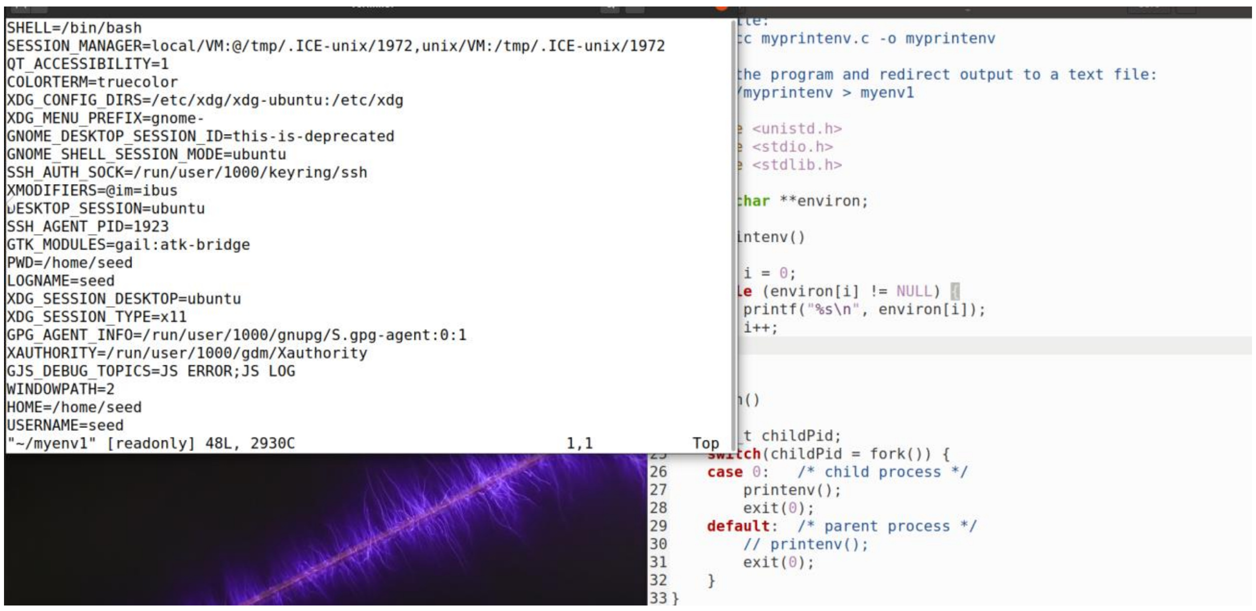


```
02/02/23]seed@VM:~$ man fork | grep environment
02/02/23]seed@VM:~$ man fork | grep variables
       variables, and other pthreads objects; the use of pthread_atfork(3)
02/02/23]seed@VM:~$ man fork
02/02/23]seed@VM:~$
```

Task 2.1



```
02/02/23]seed@VM:~$ gcc myprintenv.c -o myprintenv
02/02/23]seed@VM:~$ ./myprintenv > myenv1
02/02/23]seed@VM:~$ ./myprintenv > myenv1
02/02/23]seed@VM:~$ 
```

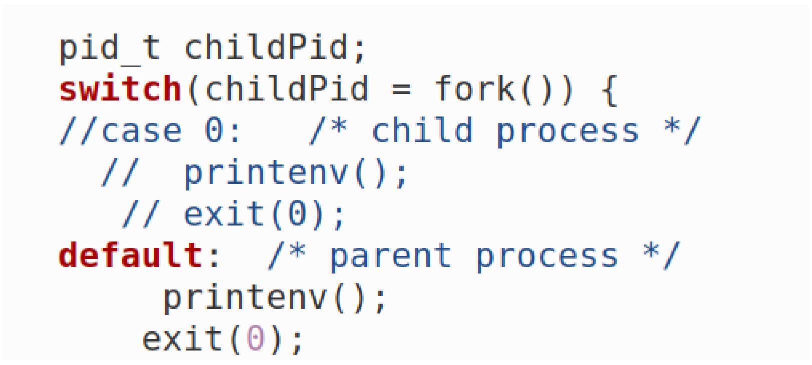


```
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1972,unix/VM:/tmp/.ICE-unix/1972
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1923
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
"~/myenv1" [readonly] 48L, 2930C                    1,1           Top
```

```
c myprintenv.c -o myprintenv

he program and redirect output to a text file:
myprintenv > myenv1

 <unistd.h>
 <stdio.h>
 <stdlib.h>

har **environ;

ntenv()

 i = 0;
e (environ[i] != NULL) {
 printf("%s\n", environ[i]);
 i++;



()

t childPid;
switch(childPid = fork()) {
  case 0:   /* child process */
      printenv();
      exit(0);
  default:  /* parent process */
      // printenv();
      exit(0);
  }
}
```

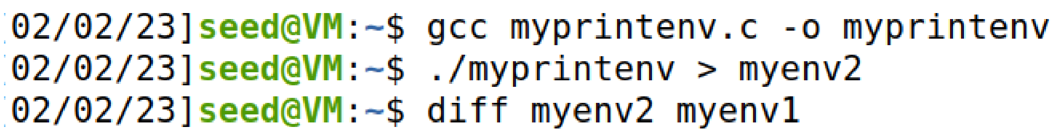
When we compile this C program and run it, it gets the PID_t info from the child process of fork(). This shows us that the environment variable is the same as the parent.

Task 2.2

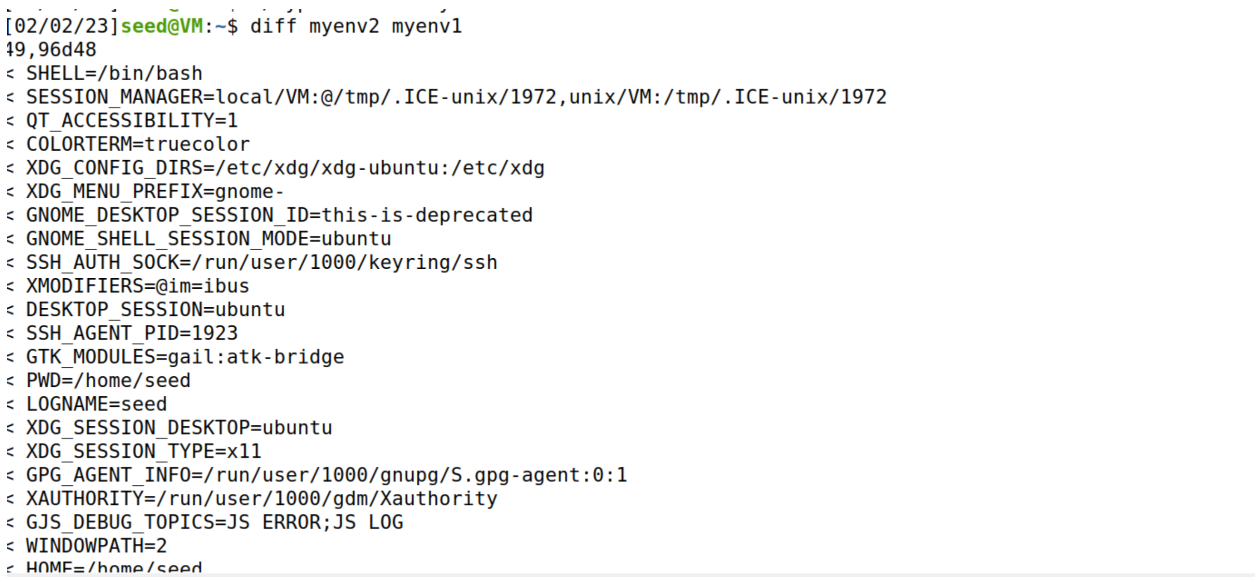


```
pid_t childPid;
switch(childPid = fork()) {
//case 0:   /* child process */
  //  printenv();
   // exit(0);
default:  /* parent process */
     printenv();
    exit(0);
```

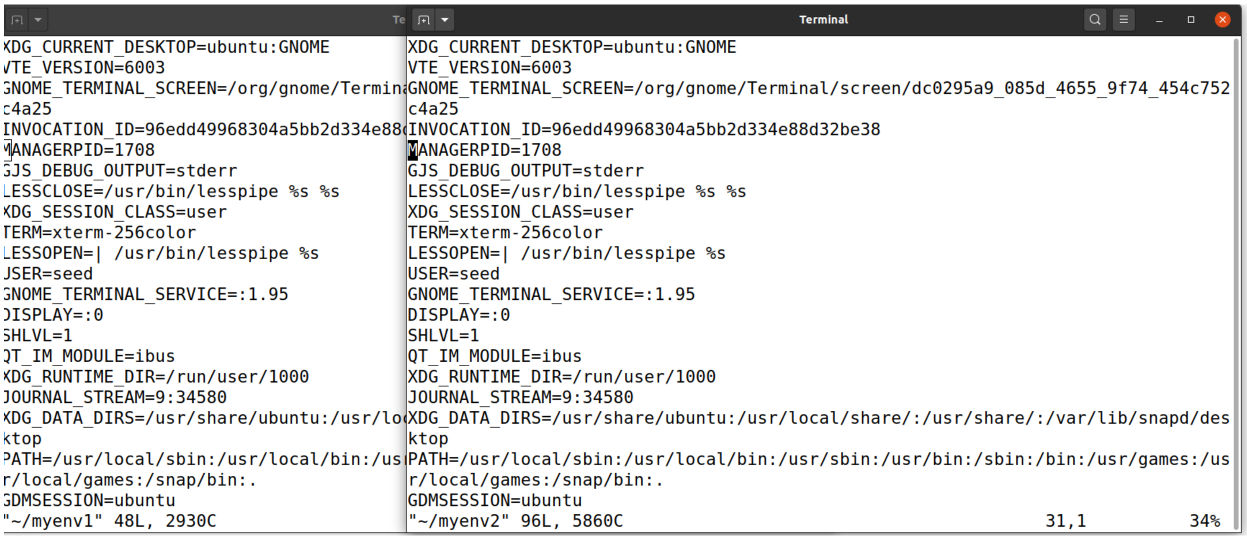
Task 2.3



```
02/02/23]seed@VM:~$ gcc myprintenv.c -o myprintenv
02/02/23]seed@VM:~$ ./myprintenv > myenv2
02/02/23]seed@VM:~$ diff myenv2 myenv1
```



```
[02/02/23]seed@VM:~$ diff myenv2 myenv1
49,96d48
< SHELL=/bin/bash
< SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1972,unix/VM:/tmp/.ICE-unix/1972
< 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=1923
< 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
```



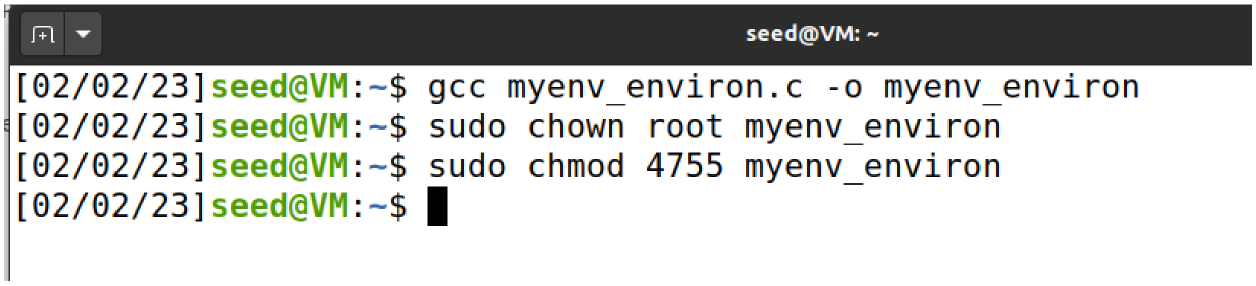
```
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Termina
c4a25
INVOCATION_ID=96edd49968304a5bb2d334e88
MANAGERPID=1708
GJS_DEBUG_OUTPUT=stderr
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.95
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:34580
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/lo
ktop
PATH=/usr/local/sbin:/usr/local/bin:/us
r/local/games:/snap/bin:.
GDMSESSION=ubuntu
"~/myenv1" 48L, 2930C
```

```
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/dc0295a9_085d_4655_9f74_454c752
c4a25
INVOCATION_ID=96edd49968304a5bb2d334e88d32be38
MANAGERPID=1708
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=:1.95
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:34580
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/des
ktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/local/games:/snap/bin:.
GDMSESSION=ubuntu
"~/myenv2" 96L, 5860C                                          31,1          34%
```

After running the diff command to compare the output off each, diff command says the entire output is different, but it shouldn't be, they are the same. I verified this by comparing them side by side.

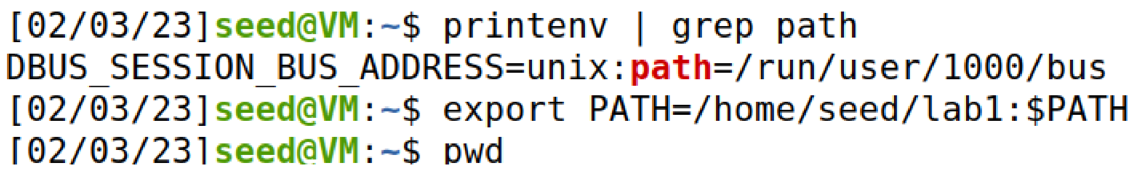
This means that the child process effectively has the same environment variables as the parent does, the child process is the same in almost every way as the parent as it is a copy. There are only a few different things it doesn't share.

Task 3.1 & Task 3.2

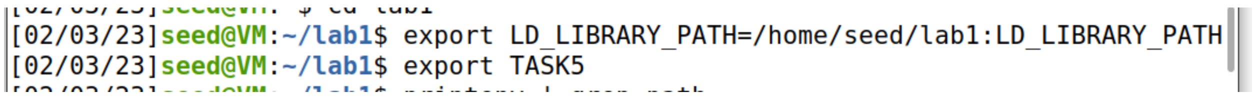


```
[02/02/23]seed@VM:~$ gcc myenv_environ.c -o myenv_environ
[02/02/23]seed@VM:~$ sudo chown root myenv_environ
[02/02/23]seed@VM:~$ sudo chmod 4755 myenv_environ
[02/02/23]seed@VM:~$
```

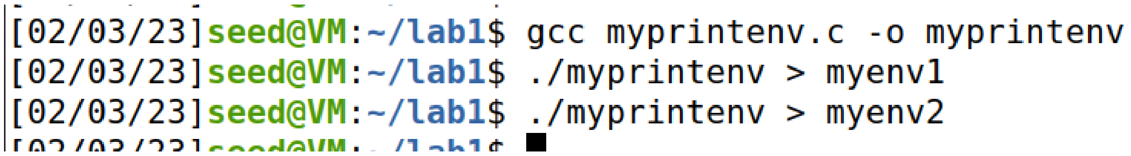
Task 3.3



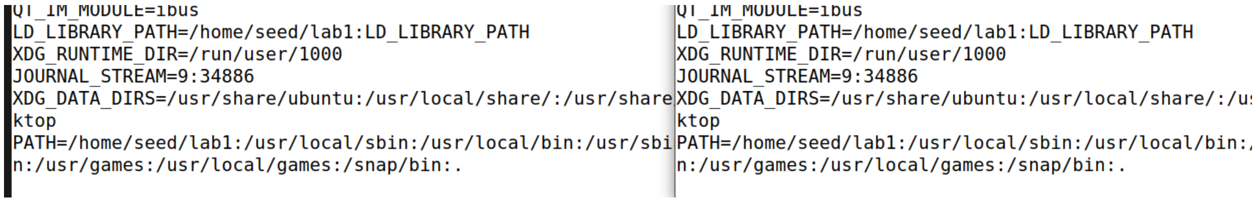
```
[02/03/23]seed@VM:~$ printenv | grep path
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
[02/03/23]seed@VM:~$ export PATH=/home/seed/lab1:$PATH
[02/03/23]seed@VM:~$ pwd
```



```
[02/03/23]seed@VM:~/lab1$ export LD_LIBRARY_PATH=/home/seed/lab1:LD_LIBRARY_PATH
[02/03/23]seed@VM:~/lab1$ export TASK5
```

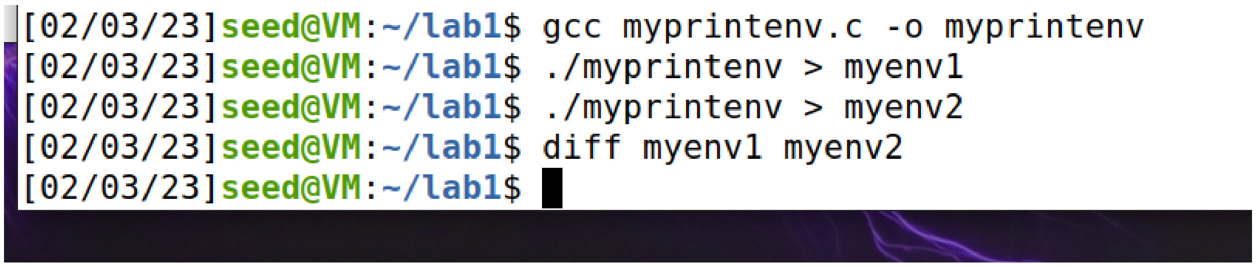


```
[02/03/23]seed@VM:~/lab1$ gcc myprintenv.c -o myprintenv
[02/03/23]seed@VM:~/lab1$ ./myprintenv > myenv1
[02/03/23]seed@VM:~/lab1$ ./myprintenv > myenv2
[02/03/23]seed@VM:~/lab1$
```



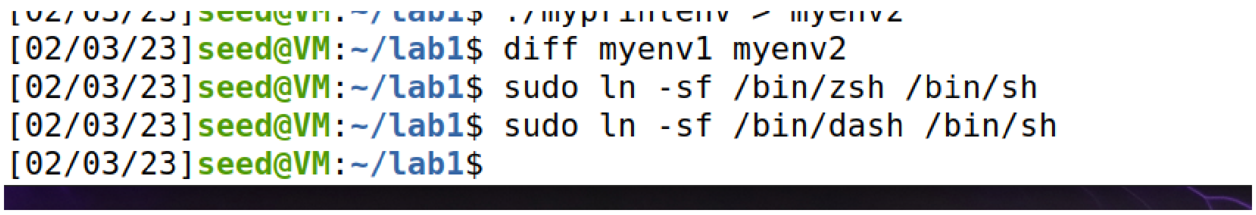
```
QT_IM_MODULE=ibus
LD_LIBRARY_PATH=/home/seed/lab1:LD_LIBRARY_PATH
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:34886
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share
ktop
PATH=/home/seed/lab1:/usr/local/sbin:/usr/local/bin:/usr/sbi
n:/usr/games:/usr/local/games:/snap/bin:.
```

```
QT_IM_MODULE=ibus
LD_LIBRARY_PATH=/home/seed/lab1:LD_LIBRARY_PATH
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:34886
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/u
ktop
PATH=/home/seed/lab1:/usr/local/sbin:/usr/local/bin:
n:/usr/games:/usr/local/games:/snap/bin:.
```



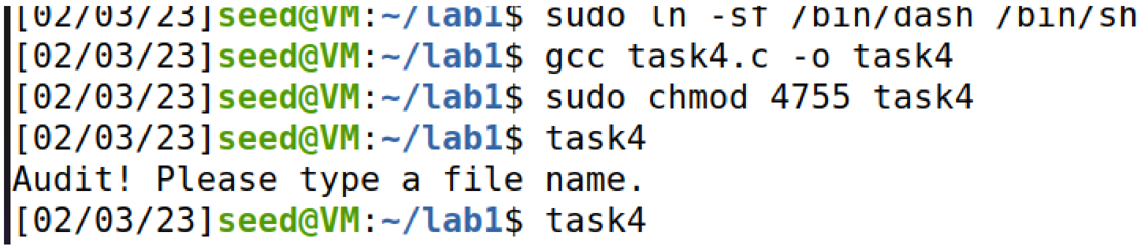
```
[02/03/23]seed@VM:~/lab1$ gcc myprintenv.c -o myprintenv
[02/03/23]seed@VM:~/lab1$ ./myprintenv > myenv1
[02/03/23]seed@VM:~/lab1$ ./myprintenv > myenv2
[02/03/23]seed@VM:~/lab1$ diff myenv1 myenv2
[02/03/23]seed@VM:~/lab1$ 
```

Now when I try diff myenv1 and myenv2, after export PATH, export LD_LIBRARY_PATH, and export TASK5, there is no difference.



```
[02/03/23]seed@VM:~/lab1$ ./myprintenv > myenv2
[02/03/23]seed@VM:~/lab1$ diff myenv1 myenv2
[02/03/23]seed@VM:~/lab1$ sudo ln -sf /bin/zsh /bin/sh
[02/03/23]seed@VM:~/lab1$ sudo ln -sf /bin/dash /bin/sh
[02/03/23]seed@VM:~/lab1$
```

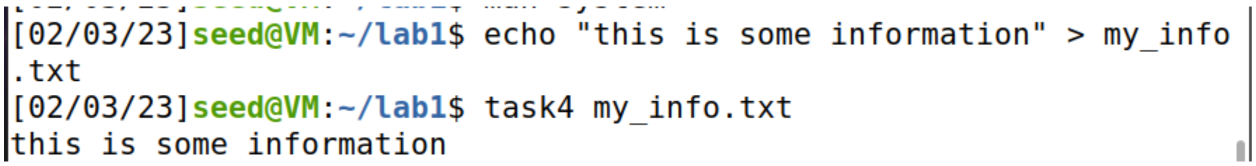
Task 4



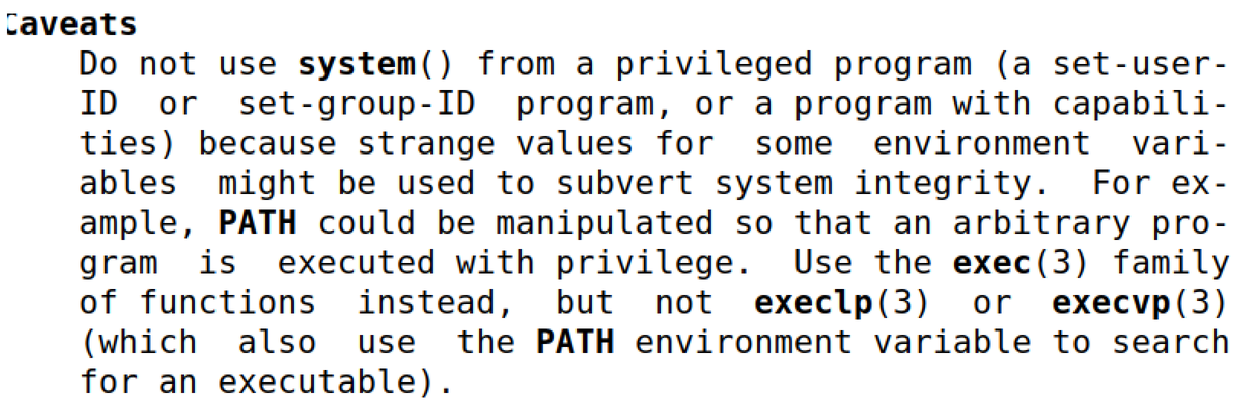
```
[02/03/23]seed@VM:~/lab1$ sudo ln -sf /bin/dash /bin/sh
[02/03/23]seed@VM:~/lab1$ gcc task4.c -o task4
[02/03/23]seed@VM:~/lab1$ sudo chmod 4755 task4
[02/03/23]seed@VM:~/lab1$ task4
Audit! Please type a file name.
[02/03/23]seed@VM:~/lab1$ task4
```

Task 4.1

Yes, if I were Bob, I could compromise the integrity of the system. "system(), because the variable affects how the shell works"



```
[02/03/23]seed@VM:~/lab1$ echo "this is some information" > my_info
.txt
[02/03/23]seed@VM:~/lab1$ task4 my_info.txt
this is some information
```

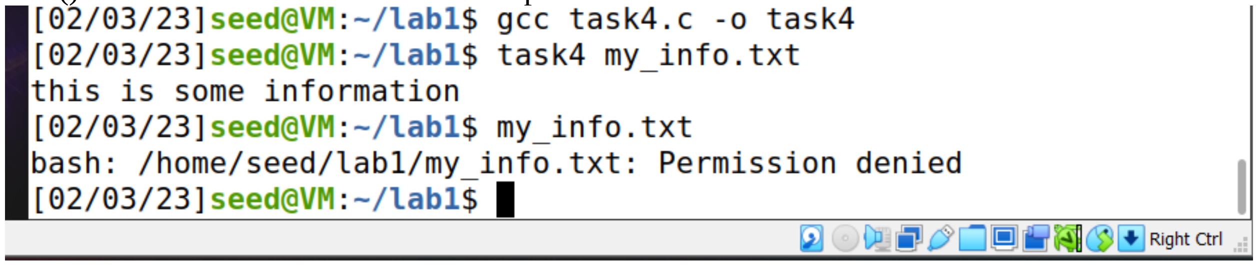


```
Caveats
    Do not use system() from a privileged program (a set-user-
    ID  or  set-group-ID  program, or a program with capabili-
    ties) because strange values for  some  environment  vari-
    ables  might be used to subvert system integrity.  For ex-
    ample, PATH could be manipulated so that an arbitrary pro-
    gram  is  executed with privilege.  Use the exec(3) family
    of functions  instead,  but  not  execlp(3)  or  execvp(3)
    (which  also  use  the PATH environment variable to search
    for an executable).
```

Task 4.2

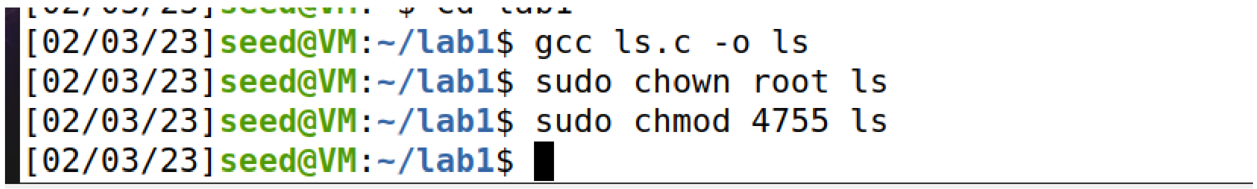
No, I changed the program to a root-owned SET-UID and ran execve statement, it can not gain full control of the system.

I don't think execve can be used to really take control of the system like system() can, but we can still read files that we aren't supposed to have access to, system() creates a child process by using fork() that executes the shell command specified.

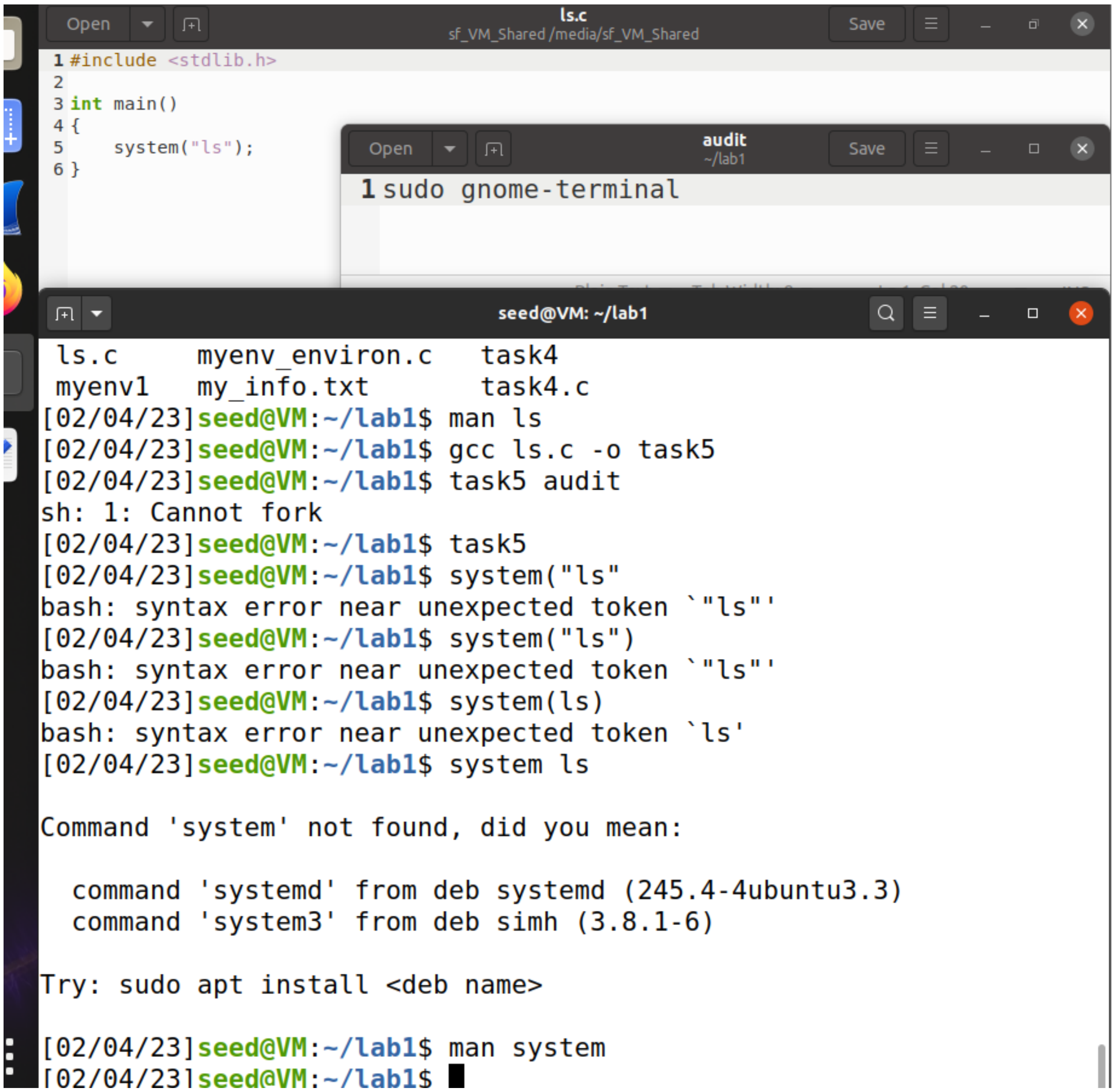


```
[02/03/23]seed@VM:~/lab1$ gcc task4.c -o task4
[02/03/23]seed@VM:~/lab1$ task4 my_info.txt
this is some information
[02/03/23]seed@VM:~/lab1$ my_info.txt
bash: /home/seed/lab1/my_info.txt: Permission denied
[02/03/23]seed@VM:~/lab1$ 
```

Task 5:



```
[02/03/23]seed@VM:~$ cd lab1
[02/03/23]seed@VM:~/lab1$ gcc ls.c -o ls
[02/03/23]seed@VM:~/lab1$ sudo chown root ls
[02/03/23]seed@VM:~/lab1$ sudo chmod 4755 ls
[02/03/23]seed@VM:~/lab1$ 
```



```
#include <stdlib.h>

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

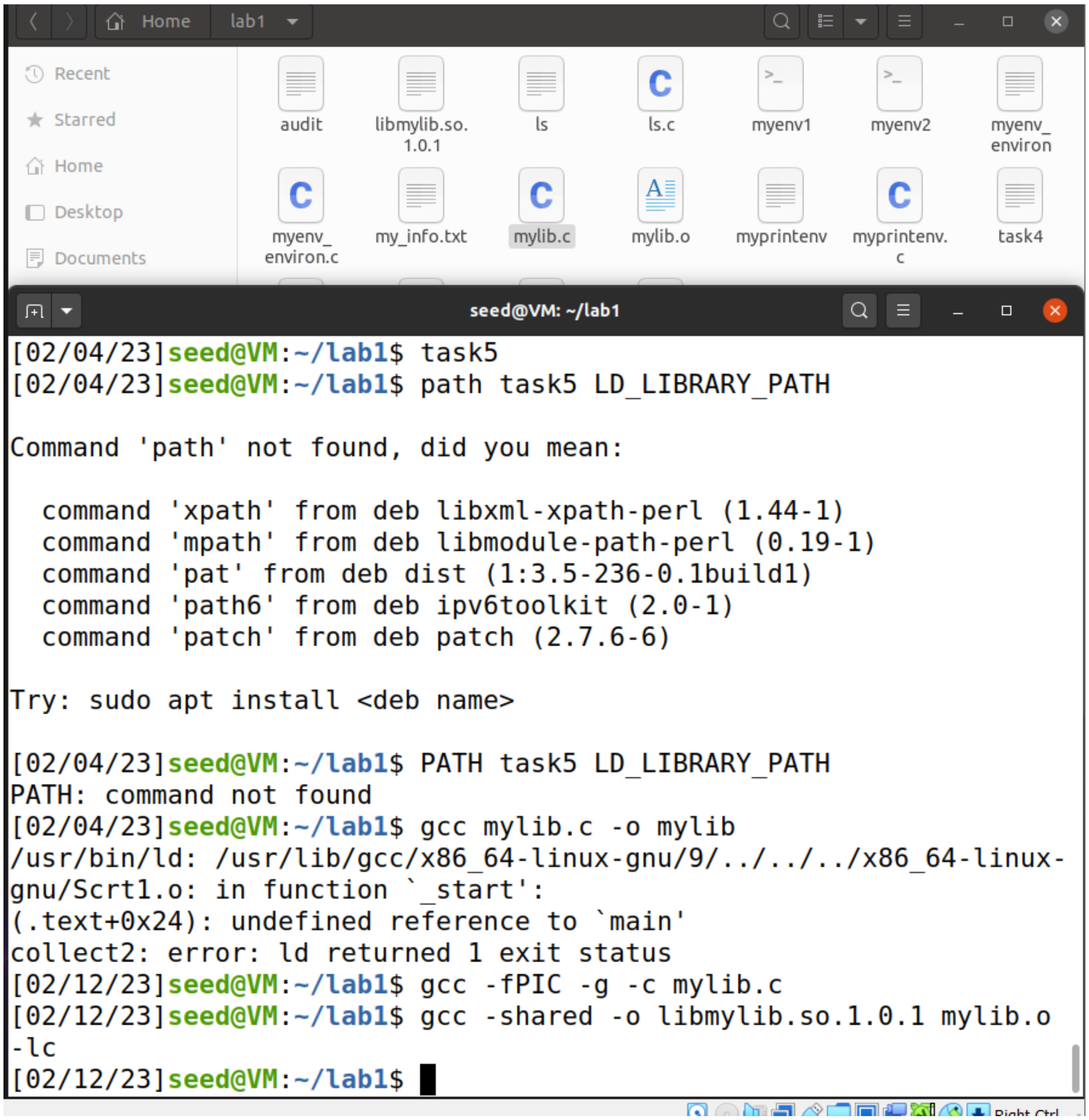
```
sudo gnome-terminal
```

```
 ls.c      myenv_environ.c    task4
 myenv1    my_info.txt        task4.c
[02/04/23]seed@VM:~/lab1$ man ls
[02/04/23]seed@VM:~/lab1$ gcc ls.c -o task5
[02/04/23]seed@VM:~/lab1$ task5 audit
sh: 1: Cannot fork
[02/04/23]seed@VM:~/lab1$ task5
[02/04/23]seed@VM:~/lab1$ system("ls"
bash: syntax error near unexpected token `"ls"'
[02/04/23]seed@VM:~/lab1$ system("ls")
bash: syntax error near unexpected token `"ls"'
[02/04/23]seed@VM:~/lab1$ system(ls)
bash: syntax error near unexpected token `ls'
[02/04/23]seed@VM:~/lab1$ system ls

Command 'system' not found, did you mean:

  command 'systemd' from deb systemd (245.4-4ubuntu3.3)
  command 'system3' from deb simh (3.8.1-6)

Try: sudo apt install <deb name>

[02/04/23]seed@VM:~/lab1$ man system
[02/04/23]seed@VM:~/lab1$ 
```

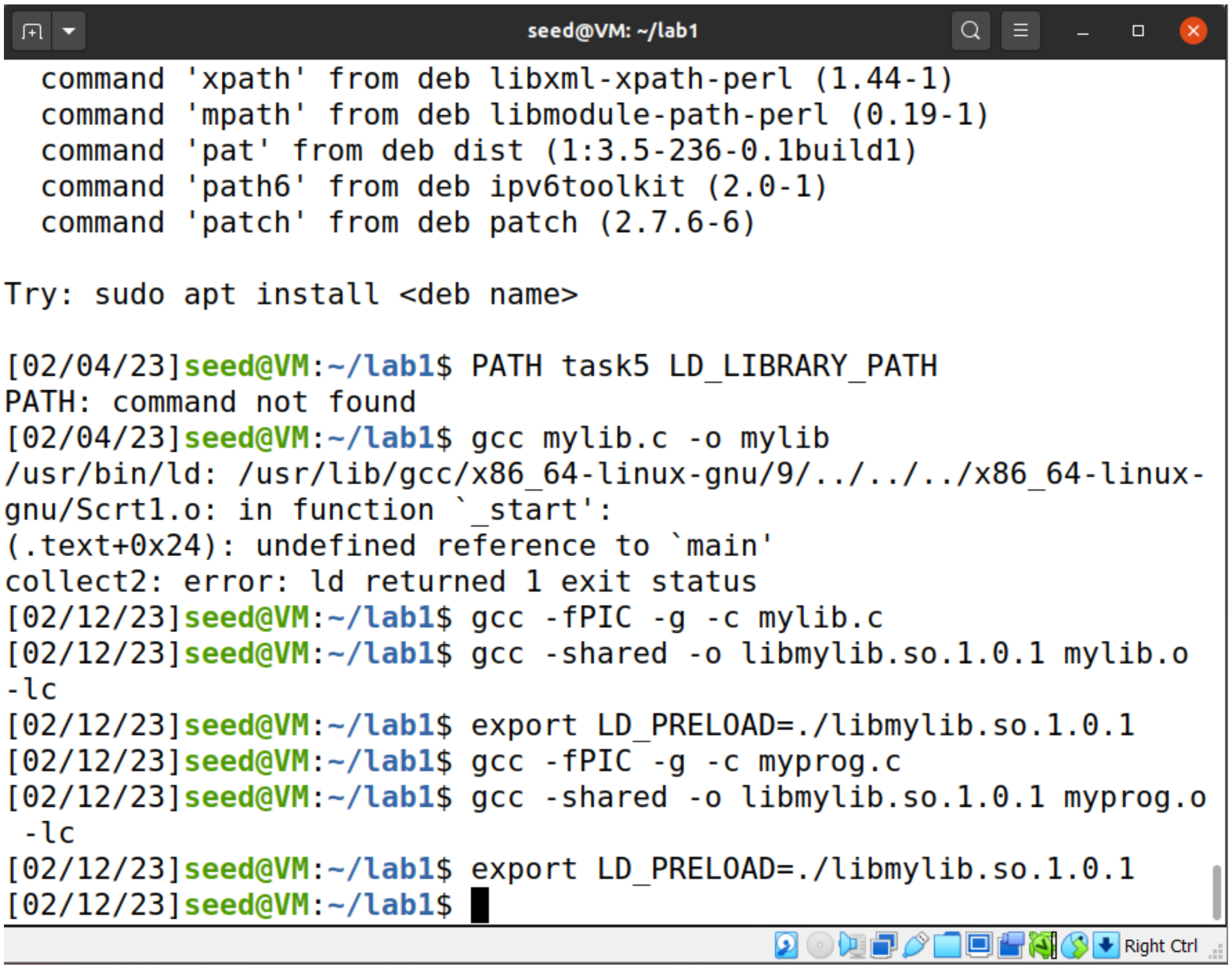
I don’t think I can have this program run anything else after it is compiled and I run it, obviously if I edit the program and then compile I can have it do whatever I want. I am sure there is a way to run any command you want after its compiled, I’m not quite sure how to make something work.

Task 6:

6.1

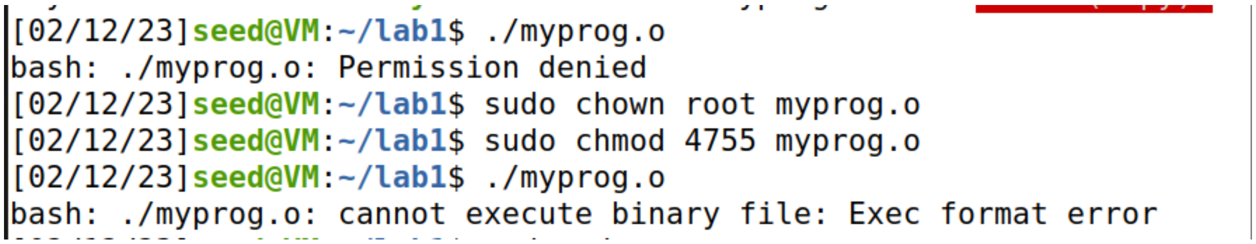


```
[02/04/23]seed@VM:~/lab1$ task5
[02/04/23]seed@VM:~/lab1$ path task5 LD_LIBRARY_PATH

Command 'path' not found, did you mean:

  command 'xpath' from deb libxml-xpath-perl (1.44-1)
  command 'mpath' from deb libmodule-path-perl (0.19-1)
  command 'pat' from deb dist (1:3.5-236-0.1build1)
  command 'path6' from deb ipv6toolkit (2.0-1)
  command 'patch' from deb patch (2.7.6-6)

Try: sudo apt install <deb name>

[02/04/23]seed@VM:~/lab1$ PATH task5 LD_LIBRARY_PATH
PATH: command not found
[02/04/23]seed@VM:~/lab1$ gcc mylib.c -o mylib
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/9/../../../x86_64-linux-
gnu/Scrt1.o: in function `_start':
(.text+0x24): undefined reference to `main'
collect2: error: ld returned 1 exit status
[02/12/23]seed@VM:~/lab1$ gcc -fPIC -g -c mylib.c
[02/12/23]seed@VM:~/lab1$ gcc -shared -o libmylib.so.1.0.1 mylib.o
-lc
[02/12/23]seed@VM:~/lab1$ 
```

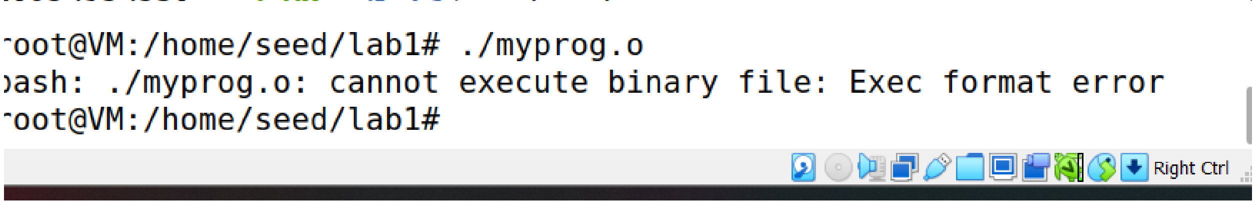


```
seed@VM: ~/lab1
  command 'xpath' from deb libxml-xpath-perl (1.44-1)
  command 'mpath' from deb libmodule-path-perl (0.19-1)
  command 'pat' from deb dist (1:3.5-236-0.1build1)
  command 'path6' from deb ipv6toolkit (2.0-1)
  command 'patch' from deb patch (2.7.6-6)

Try: sudo apt install <deb name>

[02/04/23]seed@VM:~/lab1$ PATH task5 LD_LIBRARY_PATH
PATH: command not found
[02/04/23]seed@VM:~/lab1$ gcc mylib.c -o mylib
/usr/bin/ld: /usr/lib/gcc/x86_64-linux-gnu/9/../../../x86_64-linux-
gnu/Scrt1.o: in function `_start':
(.text+0x24): undefined reference to `main'
collect2: error: ld returned 1 exit status
[02/12/23]seed@VM:~/lab1$ gcc -fPIC -g -c mylib.c
[02/12/23]seed@VM:~/lab1$ gcc -shared -o libmylib.so.1.0.1 mylib.o
-lc
[02/12/23]seed@VM:~/lab1$ export LD_PRELOAD=./libmylib.so.1.0.1
[02/12/23]seed@VM:~/lab1$ gcc -fPIC -g -c myprog.c
[02/12/23]seed@VM:~/lab1$ gcc -shared -o libmylib.so.1.0.1 myprog.o
 -lc
[02/12/23]seed@VM:~/lab1$ export LD_PRELOAD=./libmylib.so.1.0.1
[02/12/23]seed@VM:~/lab1$ 
```

6.2



```
[02/12/23]seed@VM:~/lab1$ ./myprog.o
bash: ./myprog.o: Permission denied
[02/12/23]seed@VM:~/lab1$ sudo chown root myprog.o
[02/12/23]seed@VM:~/lab1$ sudo chmod 4755 myprog.o
[02/12/23]seed@VM:~/lab1$ ./myprog.o
bash: ./myprog.o: cannot execute binary file: Exec format error
```



```
oot@VM:/home/seed/lab1# ./myprog.o
ash: ./myprog.o: cannot execute binary file: Exec format error
oot@VM:/home/seed/lab1#
```

6.3

After coming back to this lab after over a week, I left off on the end of part 5, and I am completely lost as to what I was even doing. I have no clue what differences I see honestly. All I remember is something about how we can run a program that will create a new thread of the parent process with the same properties and permissions, but we can essentially inject new commands in. I never got it to work, but I said that in part 5, this last part I have no clue whats going on though. Looking at both mylib and myprog, I see that we have two “programs” (I say “programs” because one is technically not a stand alone program, it is just a function), so when we load each program into a library, and run myprog, I would look in that local library first to see if it can satisfy the sleep function, since it sees that in the library it was loaded into has a ‘sleep’ function it can read from, it is loading that into memory to be executed, since we can essentially run anything we want to, maybe we can use that I some way to earlier in the lab, like I said, its been over a week, and I have no clue whats going on anymore ¯_(ツ)_/¯…

```
#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");
 }


/* myprog.c */
#include <unistd.h>
int main()
{
  sleep(1);
  return 0;
}
```