Dhanashree Srinivasa

SUID: 393473169

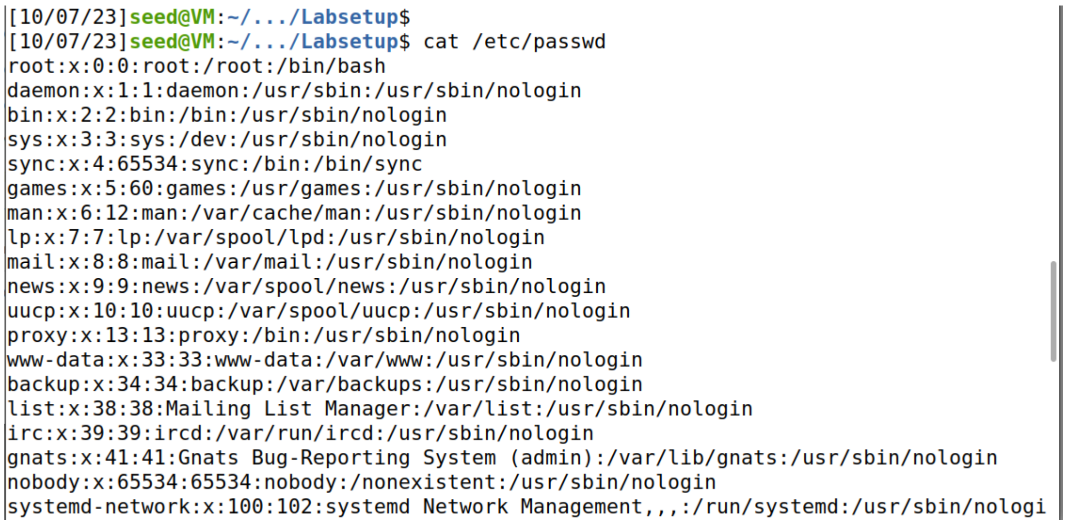
Course: Computer Security - CSE 643

Environment Variable and SetUID Lab

Task 1: Manipulating Environment Variables

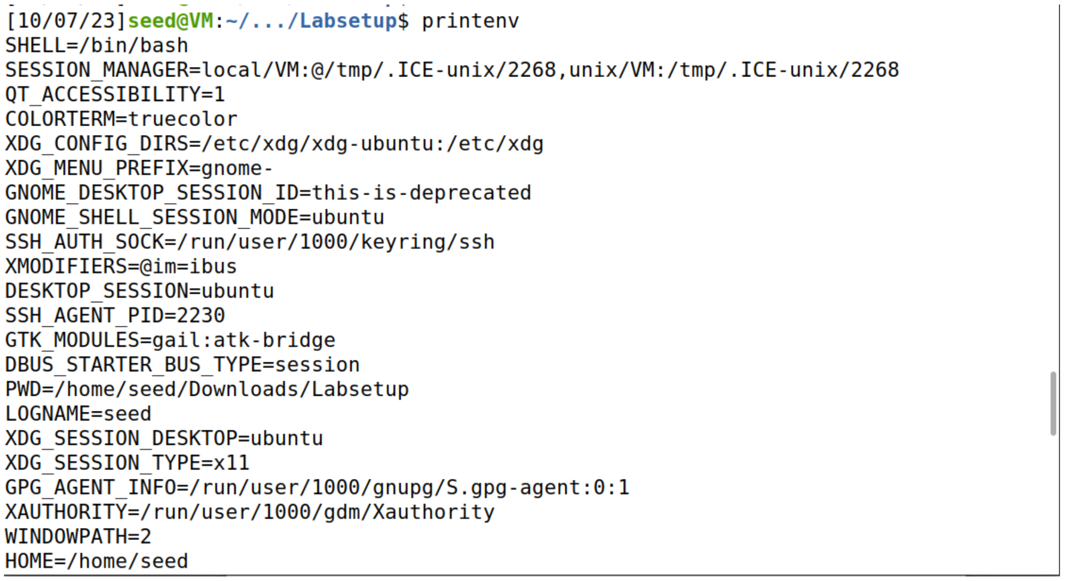
Before starting, to check for the default shell configured for my account,

command: cat /etc/passwd and the default shell was /bin/bash.



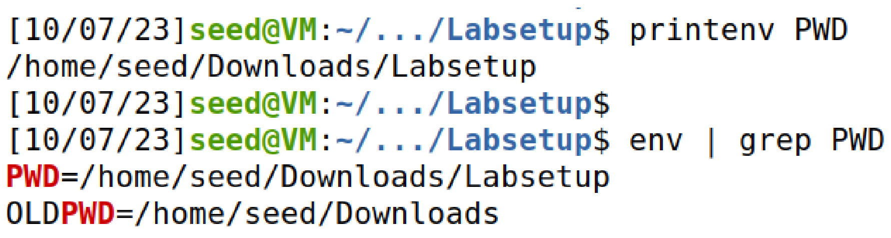
```
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
www-data:x:33:33:www-data:/var/www:/usr/sbin/nologin
backup:x:34:34:backup:/var/backups:/usr/sbin/nologin
list:x:38:38:Mailing List Manager:/var/list:/usr/sbin/nologin
irc:x:39:39:ircd:/var/run/ircd:/usr/sbin/nologin
gnats:x:41:41:Gnats Bug-Reporting System (admin):/var/lib/gnats:/usr/sbin/nologin
nobody:x:65534:65534:nobody:/nonexistent:/usr/sbin/nologin
systemd-network:x:100:102:systemd Network Management,,,:/run/systemd:/usr/sbin/nologi
```

Use the printenv command to check the environment variables.



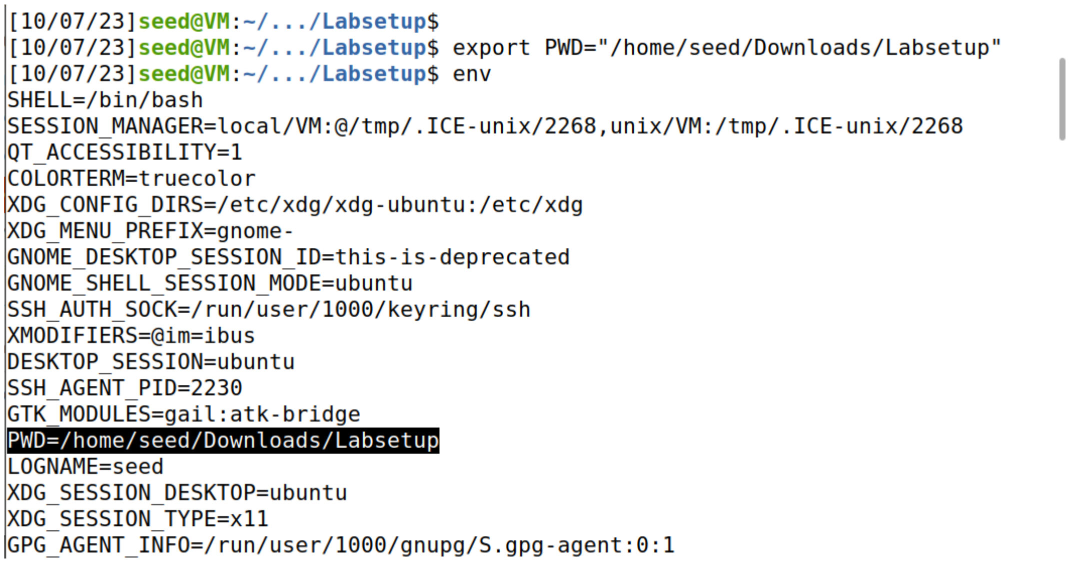
```
[10/07/23]seed@VM:~/.../Labsetup$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2268,unix/VM:/tmp/.ICE-unix/2268
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2230
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Downloads/Labsetup
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
WINDOWPATH=2
HOME=/home/seed
```

Use the env | grep PWD to filter out the PWD env variables



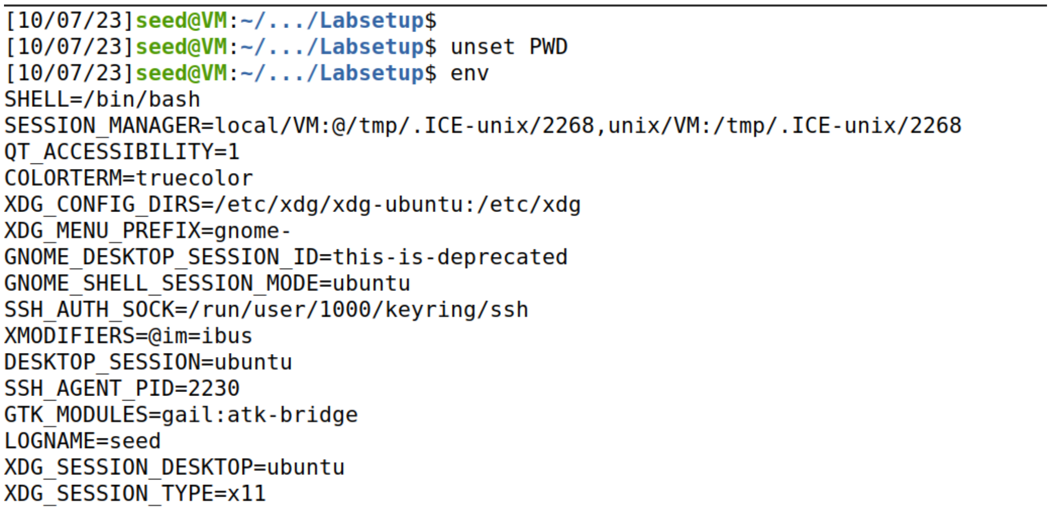
```
[10/07/23]seed@VM:~/.../Labsetup$ printenv PWD
/home/seed/Downloads/Labsetup
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ env | grep PWD
PWD=/home/seed/Downloads/Labsetup
OLDPWD=/home/seed/Downloads
```

In order to add the environment variable use the export command, and when we check the env variable we can see PWD is added.



```
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ export PWD="/home/seed/Downloads/Labsetup"
[10/07/23]seed@VM:~/.../Labsetup$ env
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2268,unix/VM:/tmp/.ICE-unix/2268
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=2230
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Downloads/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
```

To delete the environment variable, use the unset command. We can see that the PWD var is deleted.

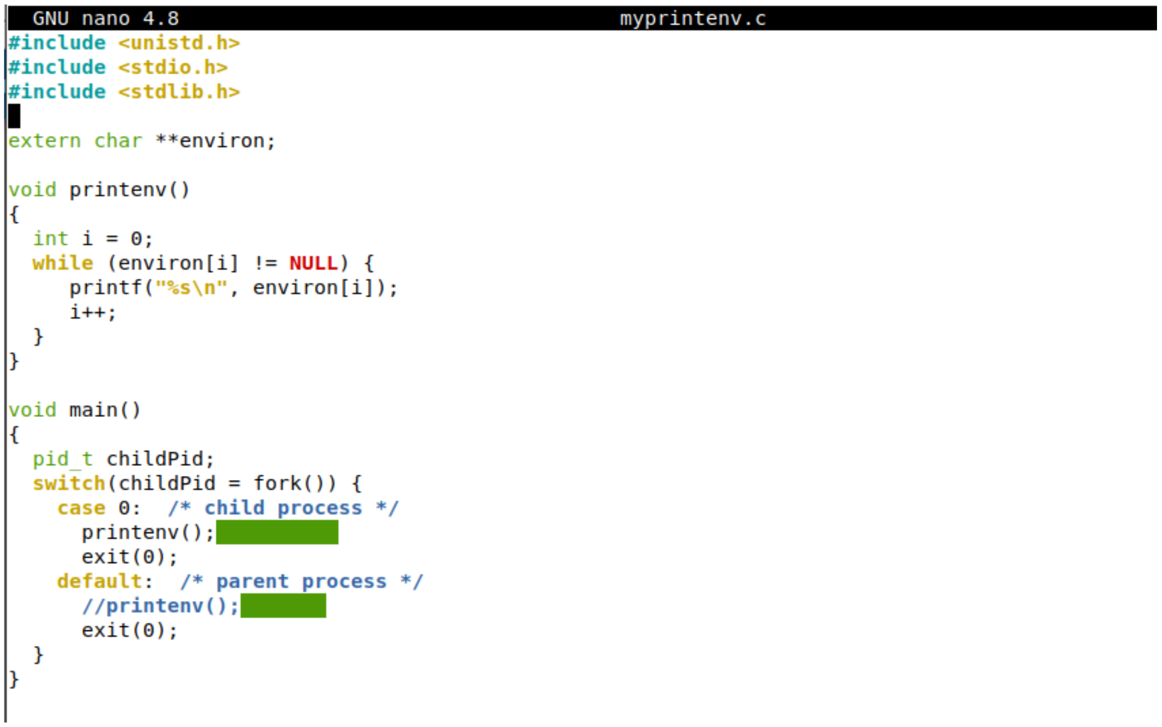


```
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ unset PWD
[10/07/23]seed@VM:~/.../Labsetup$ env
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2268,unix/VM:/tmp/.ICE-unix/2268
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
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=2230
GTK_MODULES=gail:atk-bridge
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
```

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

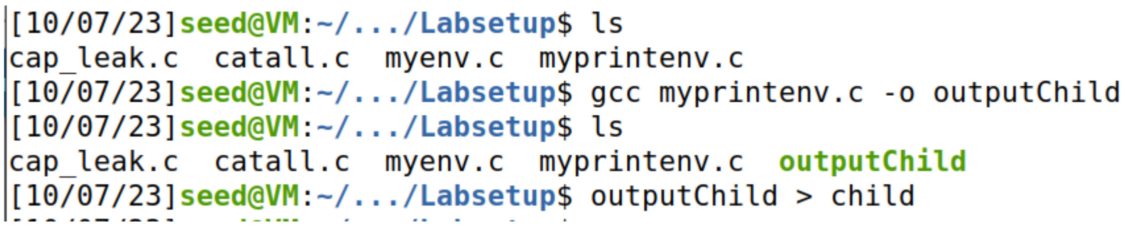
Step 1: Please compile and run the following program, and describe your observation. The program can be found in the Labsetup folder; it can be compiled using "gcc myprintenv.c", which will generate a binary called a.out. Let's run it and save the output into a file using "a.out > file".

Edit and compile the myprintenv.c file. Initially the printenv of the parent process is commented out.



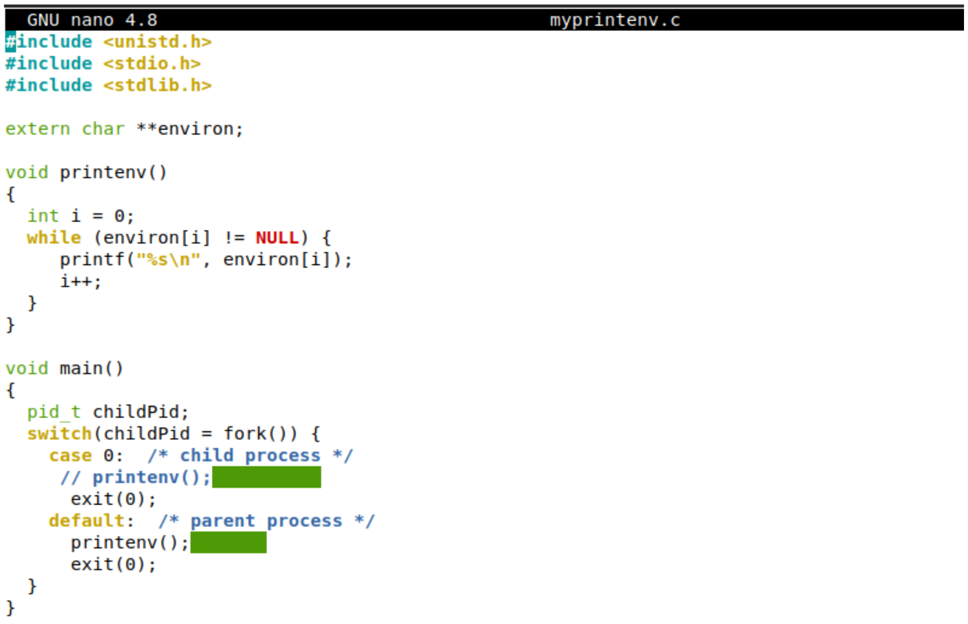
```
  GNU nano 4.8                                    myprintenv.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

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);
  }
}
```

Store the output of the file to child file.



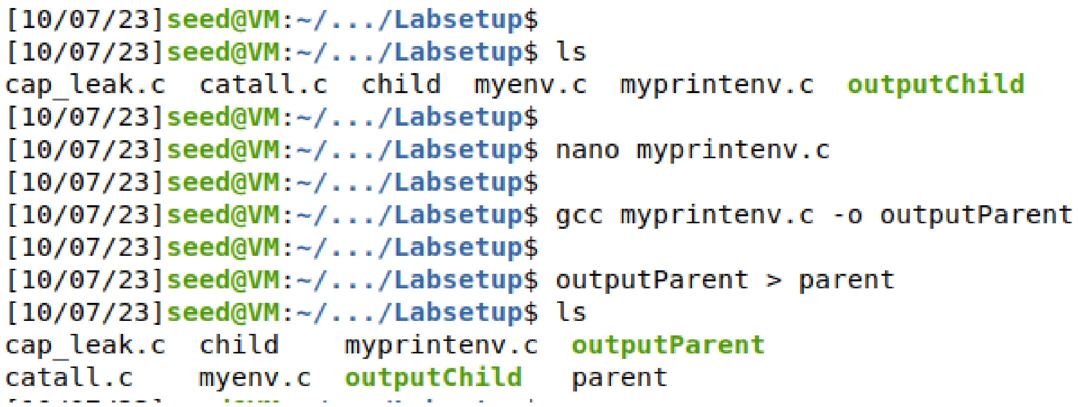
```
[10/07/23]seed@VM:~/.../Labsetup$ ls
cap_leak.c  catall.c  myenv.c  myprintenv.c
[10/07/23]seed@VM:~/.../Labsetup$ gcc myprintenv.c -o outputChild
[10/07/23]seed@VM:~/.../Labsetup$ ls
cap_leak.c  catall.c  myenv.c  myprintenv.c  outputChild
[10/07/23]seed@VM:~/.../Labsetup$ outputChild > child
```

Step 2: Now comment out the printenv() statement in the child process case (Line ①), and uncomment the printenv() statement in the parent process case (Line ②). Compile and run the code again, and describe your observation. Save the output in another file.



```
  GNU nano 4.8                                    myprintenv.c
#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);
  }
}
```

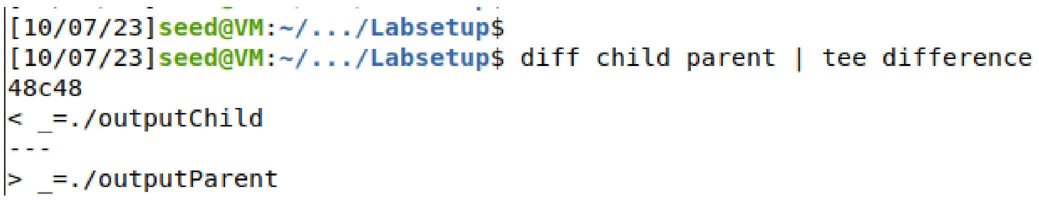
Edit the myprintenv.c file and comment the child printenv statement. And store the output in parent file.



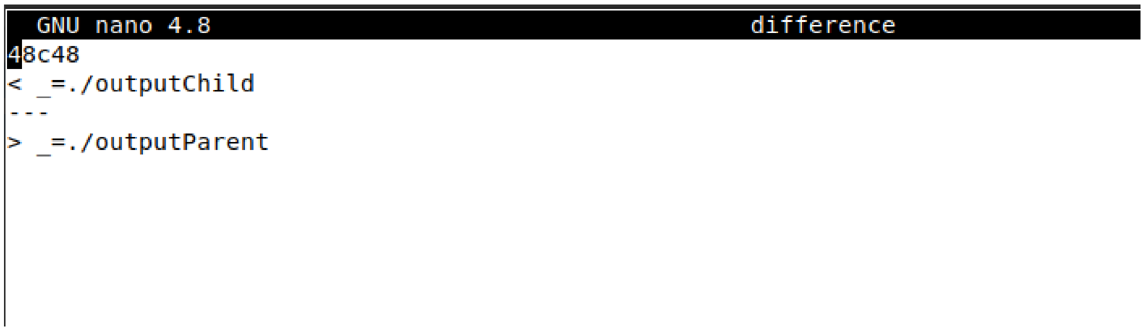
```
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ ls
cap_leak.c  catall.c  child  myenv.c  myprintenv.c  outputChild
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ nano myprintenv.c
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ gcc myprintenv.c -o outputParent
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ outputParent > parent
[10/07/23]seed@VM:~/.../Labsetup$ ls
cap_leak.c  child    myprintenv.c  outputParent
catall.c    myenv.c  outputChild   parent
```

Step 3. Compare the difference of these two files using the diff command. Please draw your conclusion.

Use the diff command and compare the output of child and parent file. We can see that the child process inherits all the environment variables of the parent.



```
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ diff child parent | tee difference
48c48
< _=./outputChild
---
> _=./outputParent
```

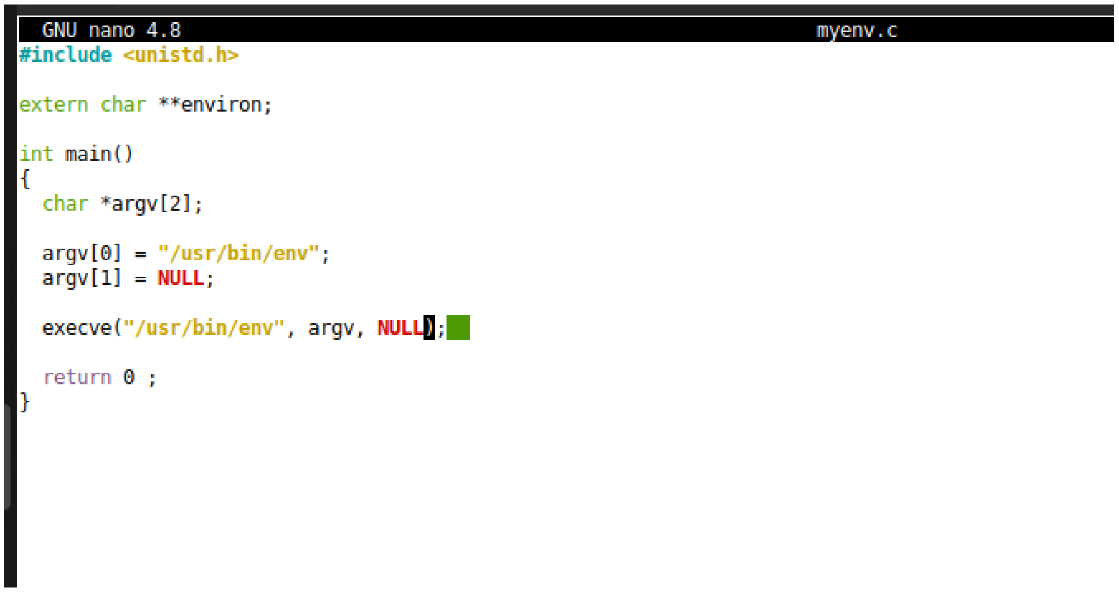


```
  GNU nano 4.8                                  difference
48c48
< _=./outputChild
---
> _=./outputParent
```

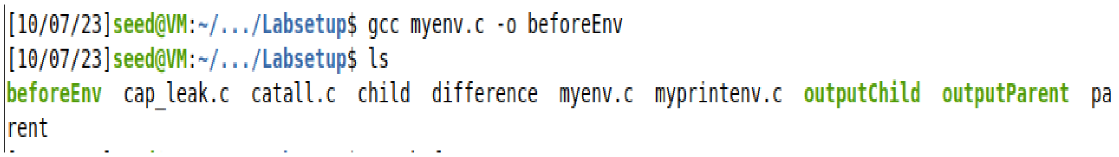
Task 3: Environment Variables and execve()

Step 1. Please compile and run the following program and describe your observation. This program simply executes a program called /usr/bin/env, which prints out the environment variables of the current process.

Edit the program myenv.c and compile the program. The program compiled successfully but we can see that there is no outout since the execve function is NULL.

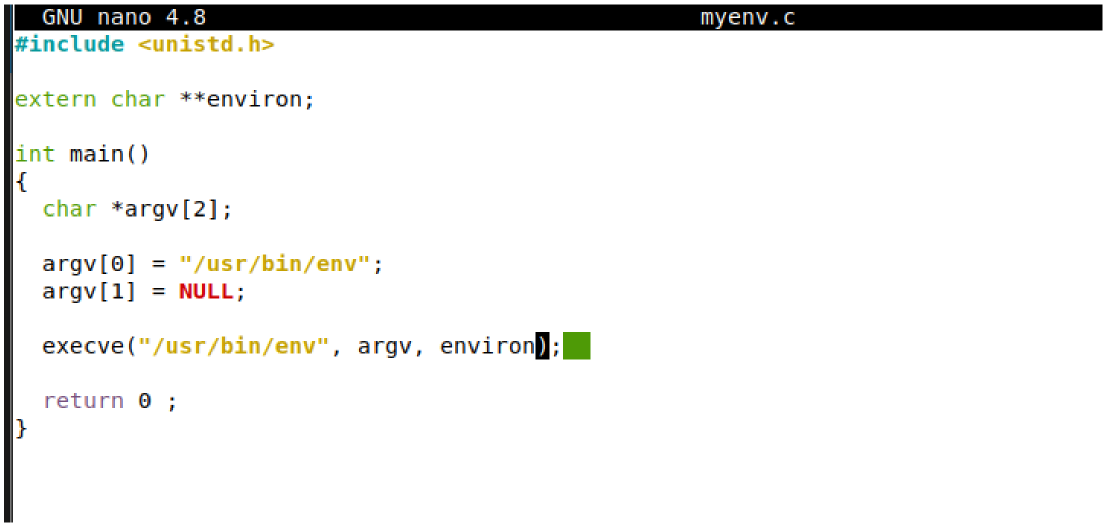


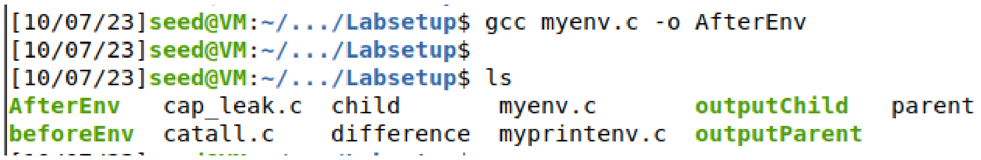
```
  GNU nano 4.8                                  myenv.c
#include <unistd.h>

extern char **environ;

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

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

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

  return 0 ;
}
```


```
[10/07/23]seed@VM:~/.../Labsetup$ gcc myenv.c -o beforeEnv
[10/07/23]seed@VM:~/.../Labsetup$ ls
beforeEnv  cap_leak.c  catall.c  child  difference  myenv.c  myprintenv.c  outputChild  outputParent  pa
rent
```

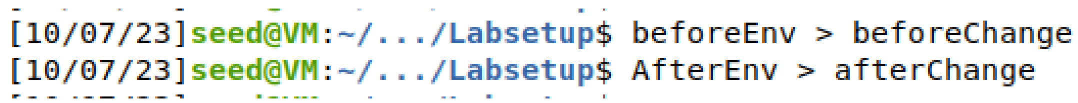
Step 2. Change the invocation of execve() in Line ① to the following; describe your observation.

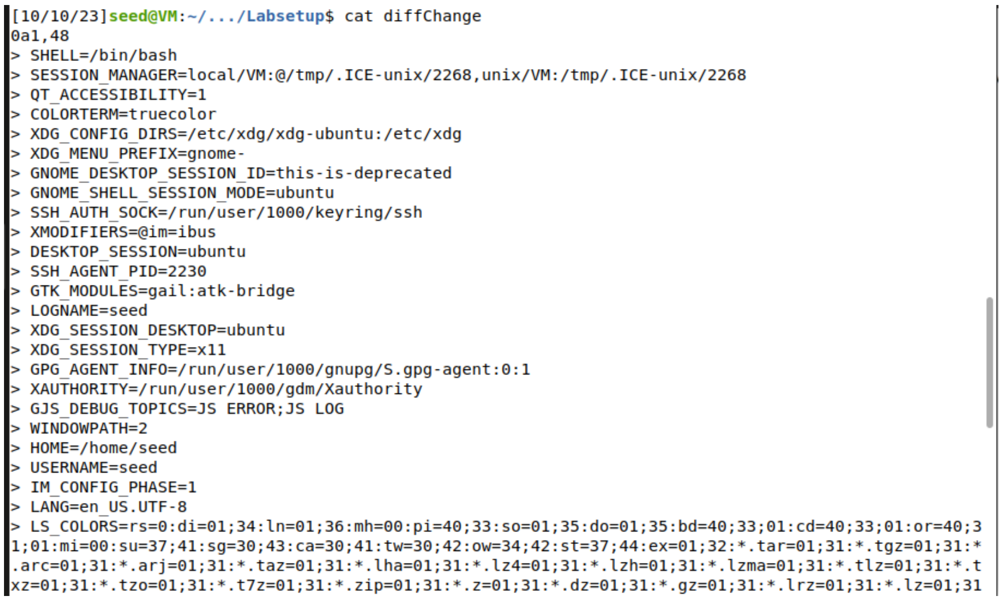
Edit the myenv.c file and add the environ as the argument.


```
  GNU nano 4.8                                        myenv.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 ;
}
```

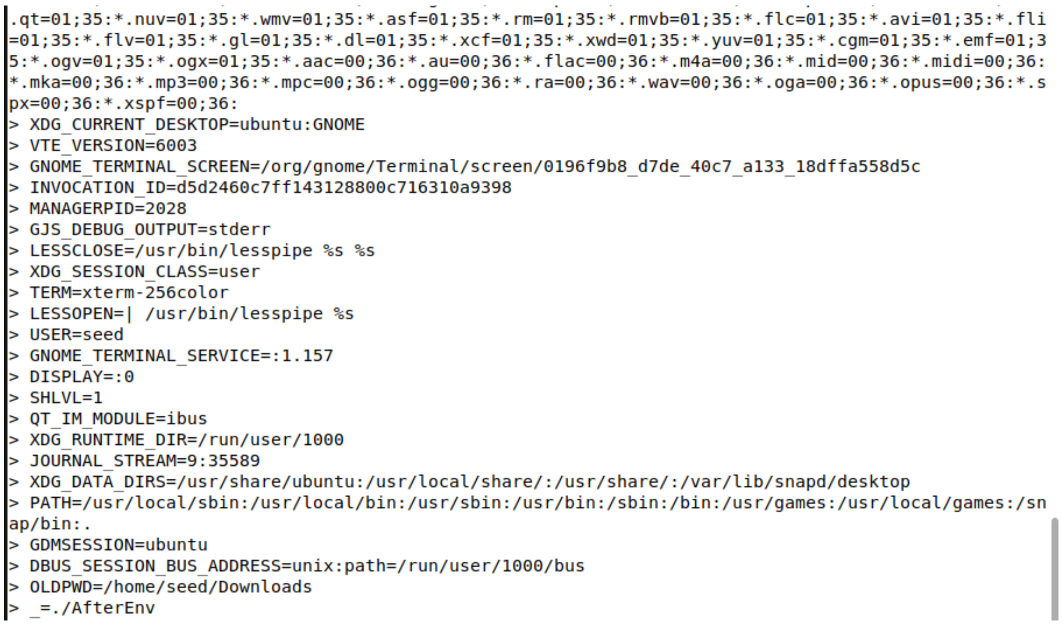

```
[10/07/23]seed@VM:~/.../Labsetup$ gcc myenv.c -o AfterEnv
[10/07/23]seed@VM:~/.../Labsetup$
[10/07/23]seed@VM:~/.../Labsetup$ ls
AfterEnv   cap_leak.c  child       myenv.c       outputChild   parent
beforeEnv  catall.c    difference  myprintenv.c  outputParent
```

Store the output to rtwo different files and print the output:


```
[10/07/23]seed@VM:~/.../Labsetup$ beforeEnv > beforeChange
[10/07/23]seed@VM:~/.../Labsetup$ AfterEnv > afterChange
```



```
[10/10/23]seed@VM:~/.../Labsetup$ cat diffChange
0a1,48
> SHELL=/bin/bash
> SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2268,unix/VM:/tmp/.ICE-unix/2268
> 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=2230
> GTK_MODULES=gail:atk-bridge
> 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
> IM_CONFIG_PHASE=1
> LANG=en_US.UTF-8
> 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;3
1;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=01;31:*.tlz=01;31:*.t
xz=01;31:*.tzo=01;31:*.t7z=01;31:*.zip=01;31:*.z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31
```

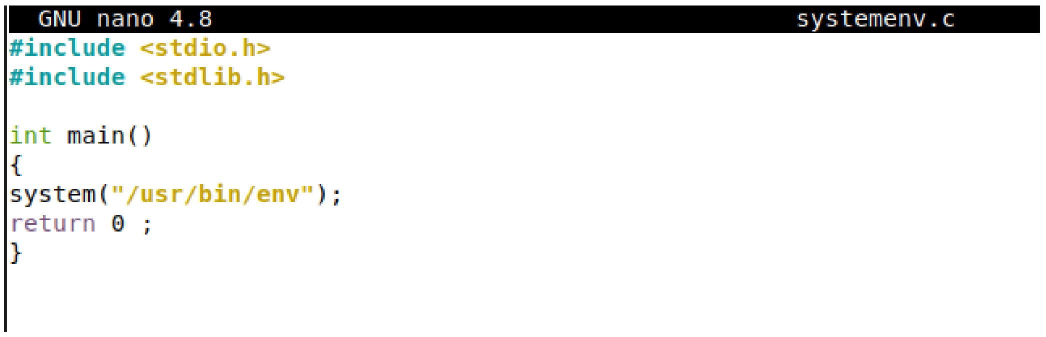


```
.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=01;35:*.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli
=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;3
5:*.ogv=01;35:*.ogx=01;35:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:
*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.s
px=00;36:*.xspf=00;36:
> XDG_CURRENT_DESKTOP=ubuntu:GNOME
> VTE_VERSION=6003
> GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/0196f9b8_d7de_40c7_a133_18dffa558d5c
> INVOCATION_ID=d5d2460c7ff143128800c716310a9398
> MANAGERPID=2028
> 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=:1.157
> DISPLAY=:0
> SHLVL=1
> QT_IM_MODULE=ibus
> XDG_RUNTIME_DIR=/run/user/1000
> JOURNAL_STREAM=9:35589
> 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:/sn
ap/bin:.
> GDMSESSION=ubuntu
> DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
> OLDPWD=/home/seed/Downloads
> _=./AfterEnv
```

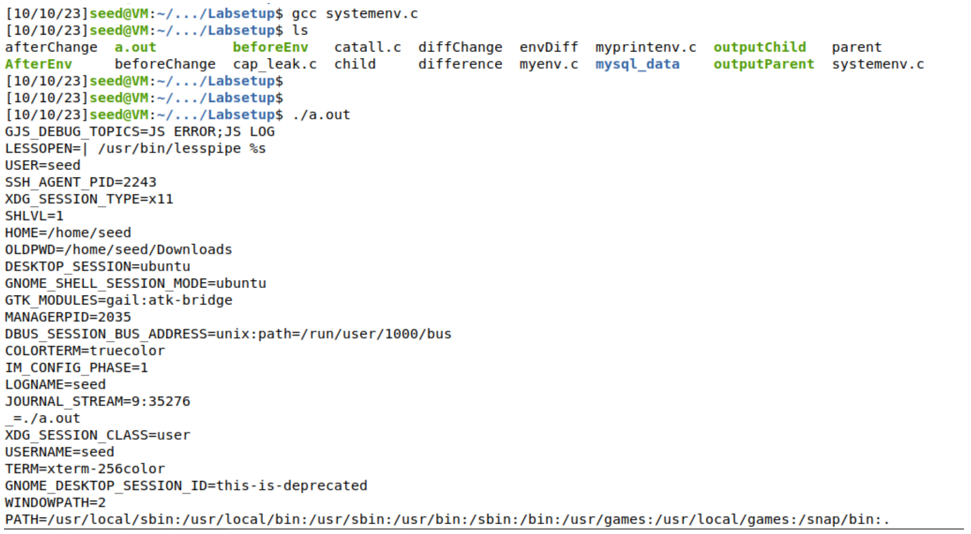
The third input to the execve() function gives the environment variable of the current process. The output was null since no environment variables were associated with this new process because the environ variable was not supplied in the initial program. However, after making changes to the program, we gave the environ variable—which held all the environment variables for the current process—as the third parameter to execve, and the program's output contained all the environment variables as expected. In conclusion, the program's environment variables are provided by the third parameter of the execve() command.

Task 4: Environment Variables and system()

Compile and execute systemenv.c



```
 GNU nano 4.8                                              systemenv.c
#include <stdio.h>
#include <stdlib.h>

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



```
[10/10/23]seed@VM:~/.../Labsetup$ gcc systemenv.c
[10/10/23]seed@VM:~/.../Labsetup$ ls
afterChange  a.out         beforeEnv   catall.c  diffChange  envDiff  myprintenv.c  outputChild   parent
AfterEnv     beforeChange  cap_leak.c  child     difference  myenv.c  mysql_data    outputParent  systemenv.c
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ ./a.out
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=2243
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
OLDPWD=/home/seed/Downloads
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=2035
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
IM_CONFIG_PHASE=1
LOGNAME=seed
JOURNAL_STREAM=9:35276
_=./a.out
XDG_SESSION_CLASS=user
USERNAME=seed
TERM=xterm-256color
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
WINDOWPATH=2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.
```

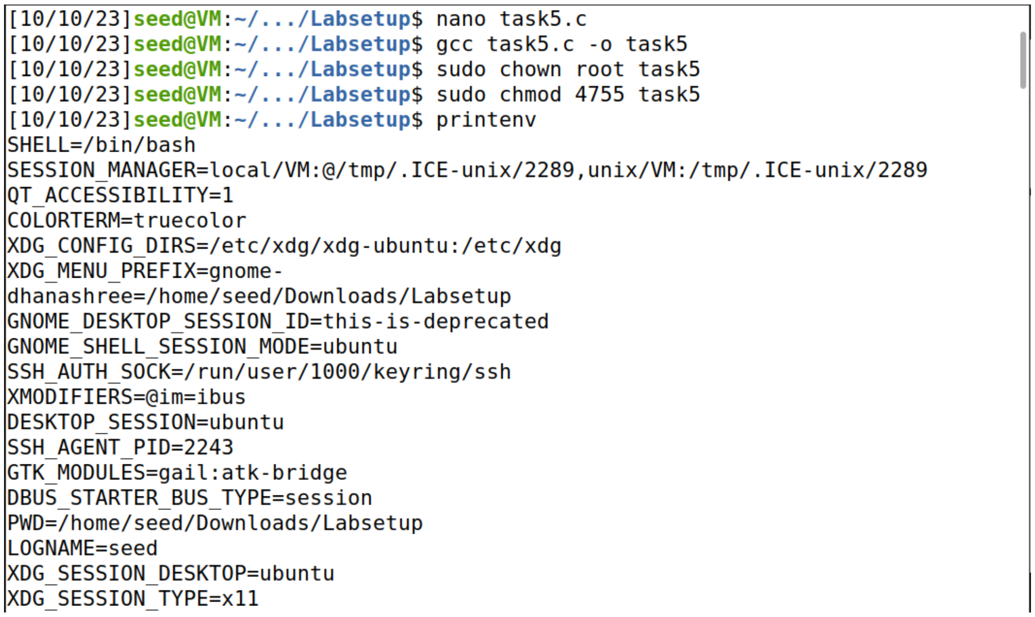
The program is compiled and run, and as can be seen, the output displays the environment variable of the current process even if we don't explicitly send any environment variables in the program. This occurs because the called function /bin/sh receives the environment variables indirectly from the system function.

Task 5: Environment Variable and Set-UID Programs

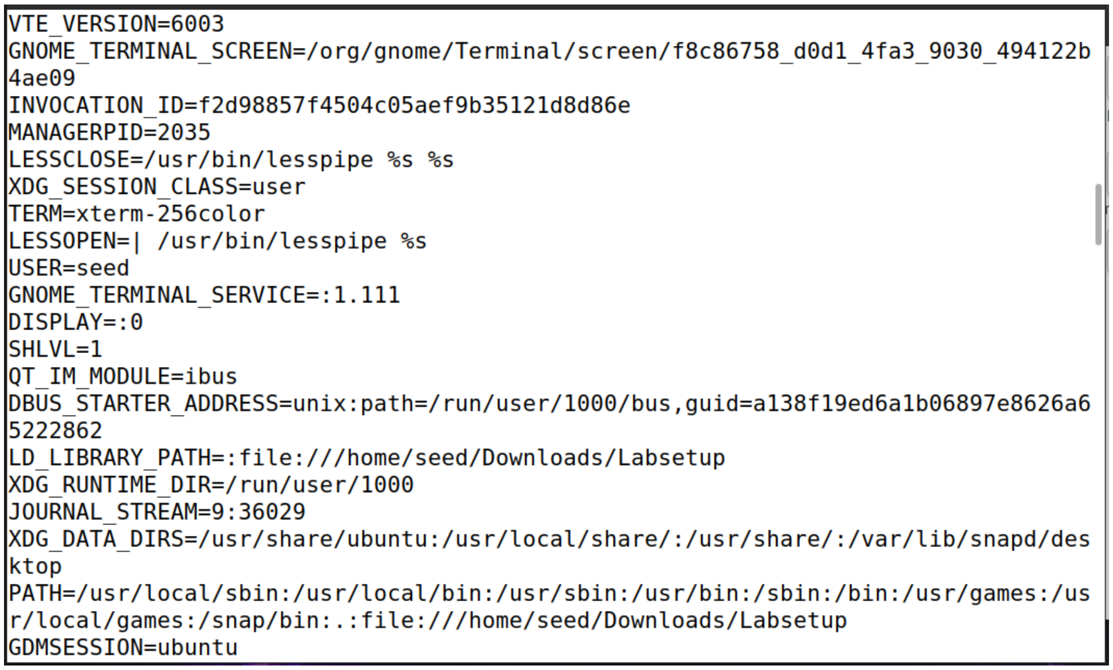
We use the following commands to modify the file's ownership and permissions after building the provided program:

sudo chown root filename (making the root as the owner of filename)

sudo chmod 4755 filename (making the program a SET-UID program by setting set-uid bit)

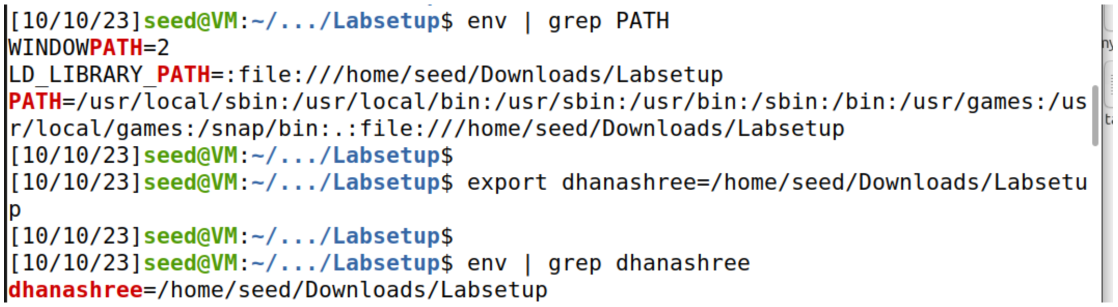


```
[10/10/23]seed@VM:~/.../Labsetup$ nano task5.c
[10/10/23]seed@VM:~/.../Labsetup$ gcc task5.c -o task5
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown root task5
[10/10/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 task5
[10/10/23]seed@VM:~/.../Labsetup$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2289,unix/VM:/tmp/.ICE-unix/2289
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
dhanashree=/home/seed/Downloads/Labsetup
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=2243
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Downloads/Labsetup
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
```



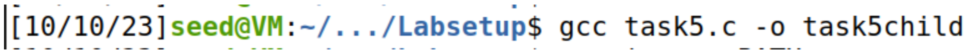
```
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/f8c86758_d0d1_4fa3_9030_494122b
4ae09
INVOCATION_ID=f2d98857f4504c05aef9b35121d8d86e
MANAGERPID=2035
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.111
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=a138f19ed6a1b06897e8626a6
5222862
LD_LIBRARY_PATH=:file:///home/seed/Downloads/Labsetup
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:36029
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:.:file:///home/seed/Downloads/Labsetup
GDMSESSION=ubuntu
```

The software is now a SET-UID root program as a result. When checking for environment variables, I just initialize one new variable with the names dhanashree and /home/seed/Downloads/Labsetup using the export command, leaving the other environment values alone because PATH and LD_LIBRARY_PATH are already there.

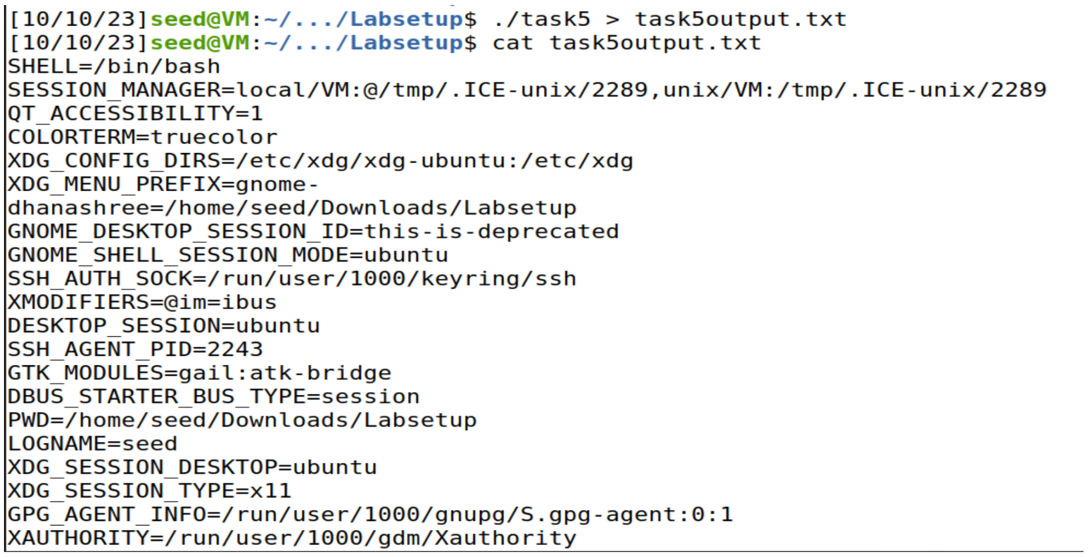


```
[10/10/23]seed@VM:~/.../Labsetup$ env | grep PATH
WINDOWPATH=2
LD_LIBRARY_PATH=:file:///home/seed/Downloads/Labsetup
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/local/games:/snap/bin:.:file:///home/seed/Downloads/Labsetup
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ export dhanashree=/home/seed/Downloads/Labsetu
p
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ env | grep dhanashree
dhanashree=/home/seed/Downloads/Labsetup
```

Compile the program task5.c and save the file in task5output.txt

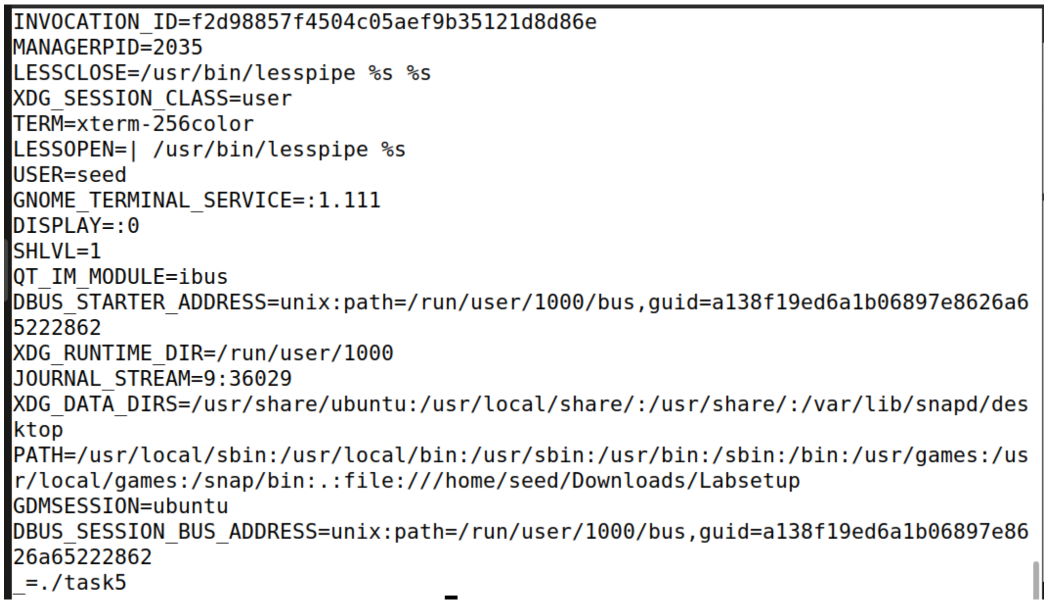


```
[10/10/23]seed@VM:~/.../Labsetup$ gcc task5.c -o task5child
```



```
[10/10/23]seed@VM:~/.../Labsetup$ ./task5 > task5output.txt
[10/10/23]seed@VM:~/.../Labsetup$ cat task5output.txt
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2289,unix/VM:/tmp/.ICE-unix/2289
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
dhanashree=/home/seed/Downloads/Labsetup
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=2243
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Downloads/Labsetup
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
```

As can be observed in the screenshot (when searching for LD in the file, it returns no data), the child process inherits the PATH and dhanashree environment variables, but there is no LD environment variable:

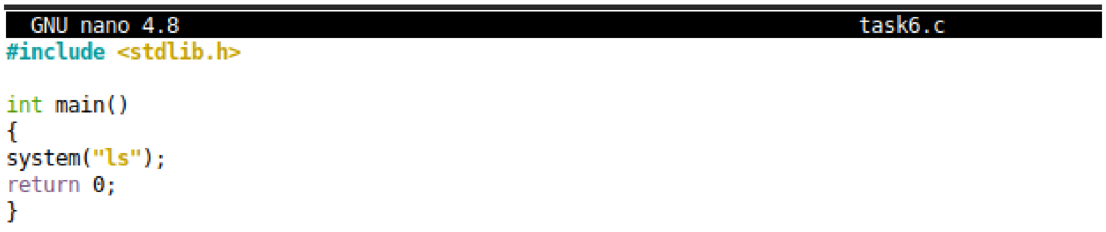


```
INVOCATION_ID=f2d98857f4504c05aef9b35121d8d86e
MANAGERPID=2035
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.111
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=a138f19ed6a1b06897e8626a6
5222862
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:36029
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:.:file:///home/seed/Downloads/Labsetup
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=a138f19ed6a1b06897e86
26a65222862
_=./task5
```

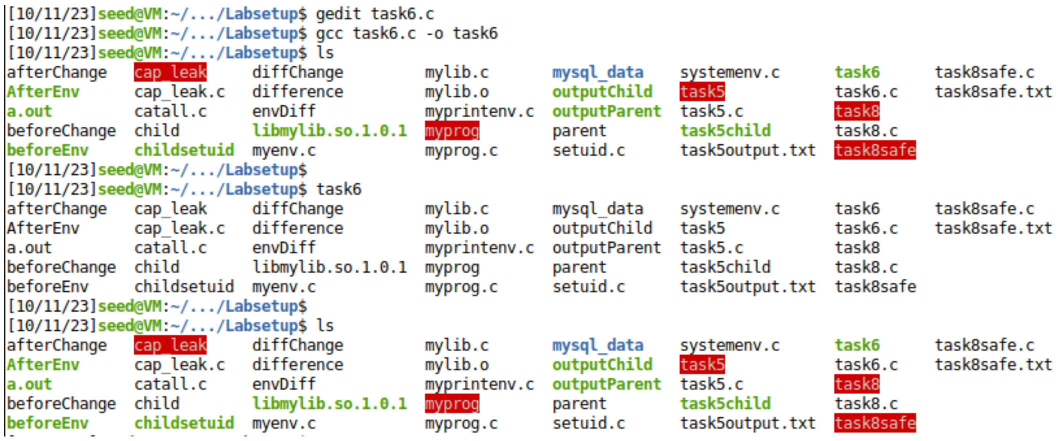
This shows that the SET-UID program's child process may not inherit all the environment variables of the parent process, LD_LIBRARY_PATH being one of them. The LD_LIBRARY_PATH is ignored here because the real user id and effective user id are different.

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

Edit and compile the file task6.c and save the compiled file in task6.

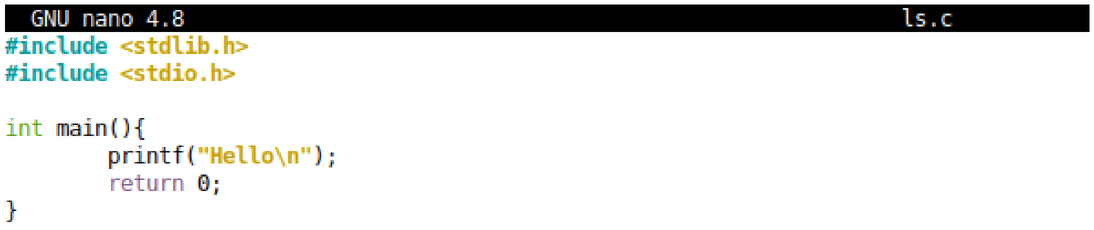


```
  GNU nano 4.8                                                    task6.c
#include <stdlib.h>

int main()
{
system("ls");
return 0;
}
```

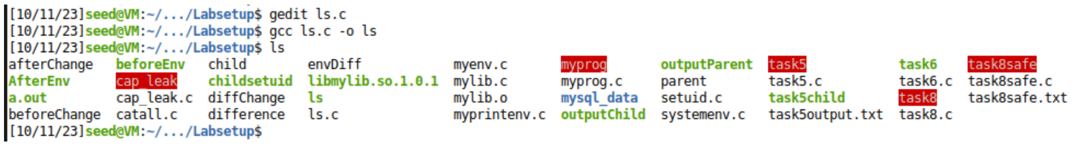


```
[10/11/23]seed@VM:~/.../Labsetup$ gedit task6.c
[10/11/23]seed@VM:~/.../Labsetup$ gcc task6.c -o task6
[10/11/23]seed@VM:~/.../Labsetup$ ls
afterChange   cap_leak     diffChange        mylib.c       mysql_data    systemenv.c      task6      task8safe.c
AfterEnv      cap_leak.c   difference        mylib.o       outputChild   task5            task6.c    task8safe.txt
a.out         catall.c     envDiff           myprintenv.c  outputParent  task5.c          task8
beforeChange  child        libmylib.so.1.0.1 myprog        parent        task5child       task8.c
beforeEnv     childsetuid  myenv.c           myprog.c      setuid.c      task5output.txt  task8safe
[10/11/23]seed@VM:~/.../Labsetup$
[10/11/23]seed@VM:~/.../Labsetup$ task6
afterChange   cap_leak     diffChange        mylib.c       mysql_data    systemenv.c      task6      task8safe.c
AfterEnv      cap_leak.c   difference        mylib.o       outputChild   task5            task6.c    task8safe.txt
a.out         catall.c     envDiff           myprintenv.c  outputParent  task5.c          task8
beforeChange  child        libmylib.so.1.0.1 myprog        parent        task5child       task8.c
beforeEnv     childsetuid  myenv.c           myprog.c      setuid.c      task5output.txt  task8safe
[10/11/23]seed@VM:~/.../Labsetup$
[10/11/23]seed@VM:~/.../Labsetup$ ls
afterChange   cap_leak     diffChange        mylib.c       mysql_data    systemenv.c      task6      task8safe.c
AfterEnv      cap_leak.c   difference        mylib.o       outputChild   task5            task6.c    task8safe.txt
a.out         catall.c     envDiff           myprintenv.c  outputParent  task5.c          task8
beforeChange  child        libmylib.so.1.0.1 myprog        parent        task5child       task8.c
beforeEnv     childsetuid  myenv.c           myprog.c      setuid.c      task5output.txt  task8safe
```

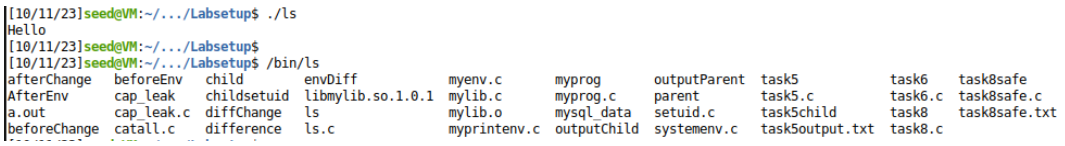
Edit and compile the file ls.c and the compiled file ls.



```
  GNU nano 4.8                                                  ls.c
#include <stdlib.h>
#include <stdio.h>

int main(){
        printf("Hello\n");
        return 0;
}
```



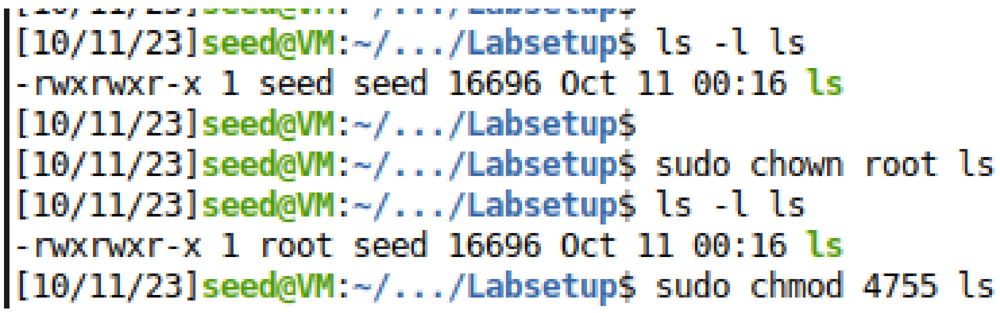
```
[10/11/23]seed@VM:~/.../Labsetup$ gedit ls.c
[10/11/23]seed@VM:~/.../Labsetup$ gcc ls.c -o ls
[10/11/23]seed@VM:~/.../Labsetup$ ls
afterChange  beforeEnv  child        envDiff             myenv.c       myprog       outputParent  task5            task6    task8safe
AfterEnv     cap_leak   childsetuid  libmylib.so.1.0.1  mylib.c       myprog.c     parent        task5.c          task6.c  task8safe.c
a.out        cap_leak.c diffChange   ls                 mylib.o       mysql_data   setuid.c      task5child       task8    task8safe.txt
beforeChange catall.c   difference   ls.c               myprintenv.c  outputChild  systemenv.c   task5output.txt  task8.c
[10/11/23]seed@VM:~/.../Labsetup$
```



```
[10/11/23]seed@VM:~/.../Labsetup$ ./ls
Hello
[10/11/23]seed@VM:~/.../Labsetup$
[10/11/23]seed@VM:~/.../Labsetup$ /bin/ls
afterChange  beforeEnv  child        envDiff             myenv.c       myprog       outputParent  task5            task6    task8safe
AfterEnv     cap_leak   childsetuid  libmylib.so.1.0.1  mylib.c       myprog.c     parent        task5.c          task6.c  task8safe.c
a.out        cap_leak.c diffChange   ls                 mylib.o       mysql_data   setuid.c      task5child       task8    task8safe.txt
beforeChange catall.c   difference   ls.c               myprintenv.c  outputChild  systemenv.c   task5output.txt  task8.c
```

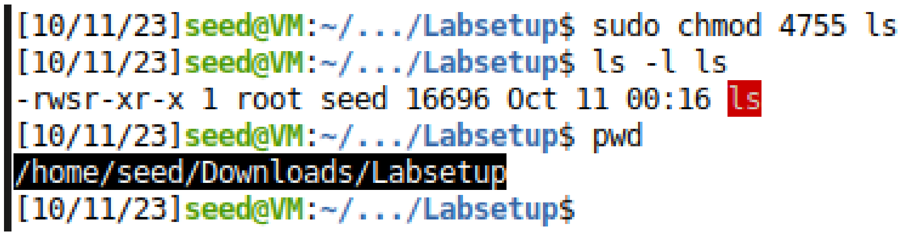
The owner of the compiled program is changed to root, and it is transformed into a SET-UID program.

Next, we check the present working directory of the program as well as the environment variable PATH's current value.



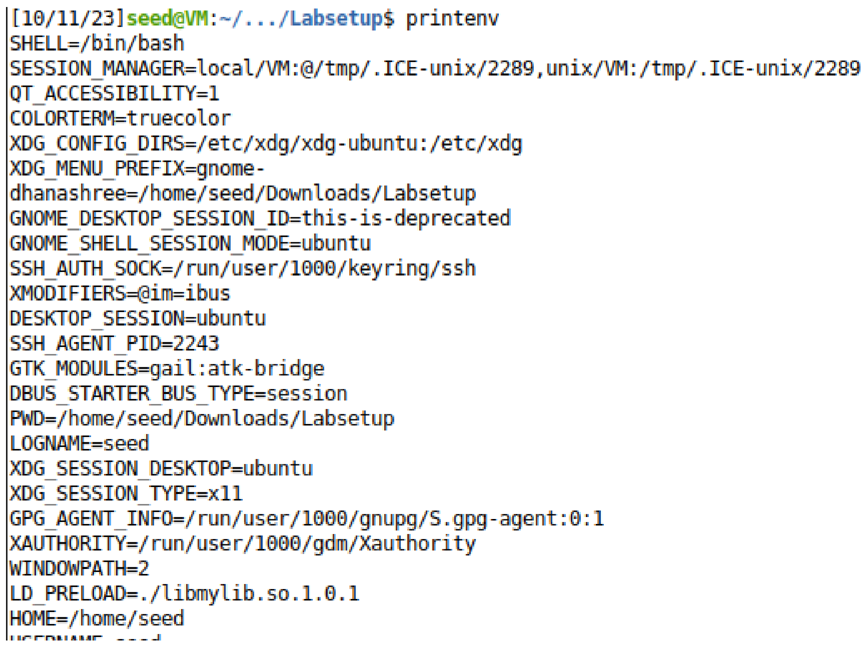
```
[10/11/23]seed@VM:~/.../Labsetup$ ls -l ls
-rwxrwxr-x 1 seed seed 16696 Oct 11 00:16 ls
[10/11/23]seed@VM:~/.../Labsetup$
[10/11/23]seed@VM:~/.../Labsetup$ sudo chown root ls
[10/11/23]seed@VM:~/.../Labsetup$ ls -l ls
-rwxrwxr-x 1 root seed 16696 Oct 11 00:16 ls
[10/11/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 ls
```

Confirming that task6compiled is a SET-UID program with root as the owner:

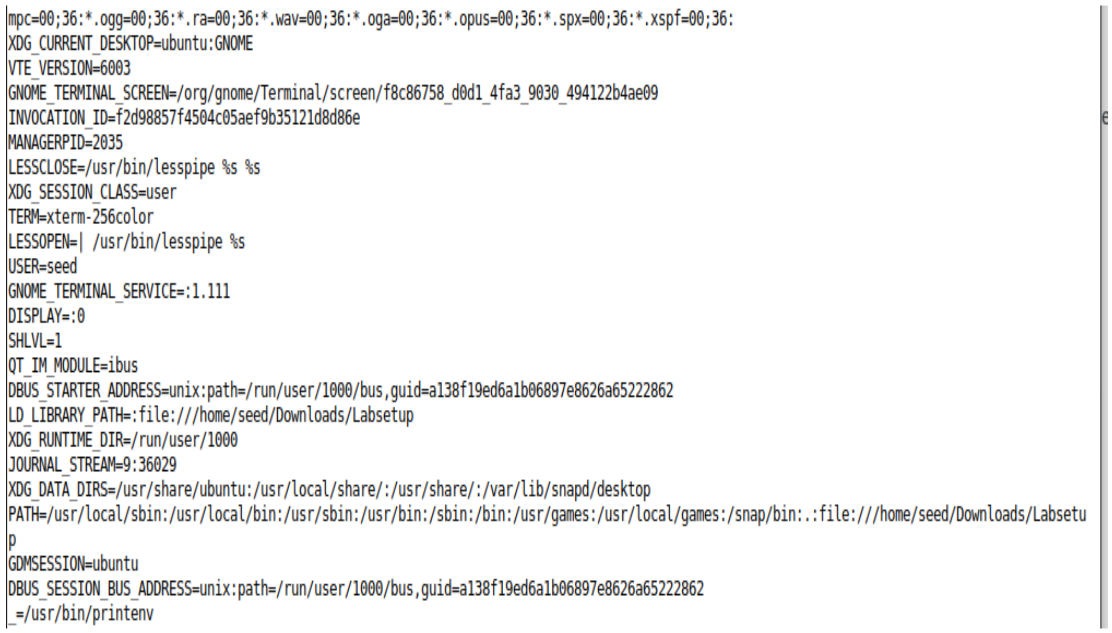


```
[10/11/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 ls
[10/11/23]seed@VM:~/.../Labsetup$ ls -l ls
-rwsr-xr-x 1 root seed 16696 Oct 11 00:16 ls
[10/11/23]seed@VM:~/.../Labsetup$ pwd
/home/seed/Downloads/Labsetup
[10/11/23]seed@VM:~/.../Labsetup$
```

Print the environment variables.



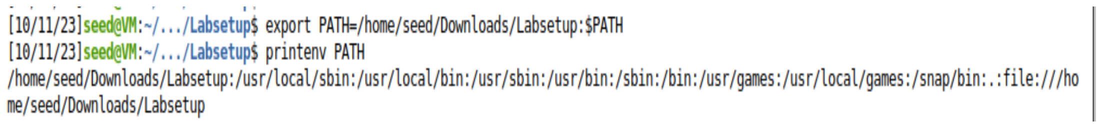
```
[10/11/23]seed@VM:~/.../Labsetup$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/2289,unix/VM:/tmp/.ICE-unix/2289
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
dhanashree=/home/seed/Downloads/Labsetup
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2243
GTK_MODULES=gail:atk-bridge
DBUS_STARTER_BUS_TYPE=session
PWD=/home/seed/Downloads/Labsetup
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
WINDOWPATH=2
LD_PRELOAD=./libmylib.so.1.0.1
HOME=/home/seed
```



```
mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.opus=00;36:*.spx=00;36:*.xspf=00;36:
XDG_CURRENT_DESKTOP=ubuntu:GNOME
VTE_VERSION=6003
GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/f8c86758_d0d1_4fa3_9030_494122b4ae09
INVOCATION_ID=f2d98857f4504c05aef9b35121d8d86e
MANAGERPID=2035
LESSCLOSE=/usr/bin/lesspipe %s %s
XDG_SESSION_CLASS=user
TERM=xterm-256color
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
GNOME_TERMINAL_SERVICE=:1.111
DISPLAY=:0
SHLVL=1
QT_IM_MODULE=ibus
DBUS_STARTER_ADDRESS=unix:path=/run/user/1000/bus,guid=a138f19ed6a1b06897e8626a65222862
LD_LIBRARY_PATH=:file:///home/seed/Downloads/Labsetup
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:36029
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:.:file:///home/seed/Downloads/Labsetu
p
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus,guid=a138f19ed6a1b06897e8626a65222862
_=/usr/bin/printenv
```

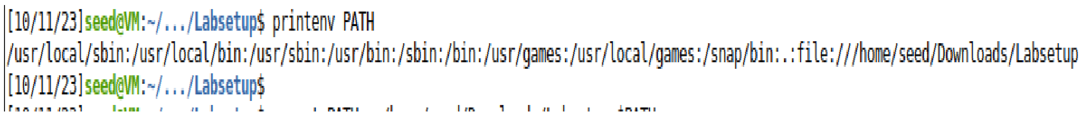
Modify the value of the environment variable PATH and supplied the path to my file as the variable's end value in order to execute my program instead of the default ls program.

This instructs the software to look in my directory before any other directories to find the file, and because I have a file with the same name as ls, the current program will run my program.



```
[10/11/23]seed@VM:~/.../Labsetup$ export PATH=/home/seed/Downloads/Labsetup:$PATH
[10/11/23]seed@VM:~/.../Labsetup$ printenv PATH
/home/seed/Downloads/Labsetup:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.:file:///ho
me/seed/Downloads/Labsetup
```

Print the env Path variable.



```
[10/11/23]seed@VM:~/.../Labsetup$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:.:file:///home/seed/Downloads/Labsetup
[10/11/23]seed@VM:~/.../Labsetup$
```

This demonstrates how to modify the PATH environment variable to point to a desired folder and run user-defined applications that might be harmful. Because we are using system(), there is a chance for harm because the shell and environment variables are present.

Additionally, we have supplied the relative path of the process rather than its absolute path.

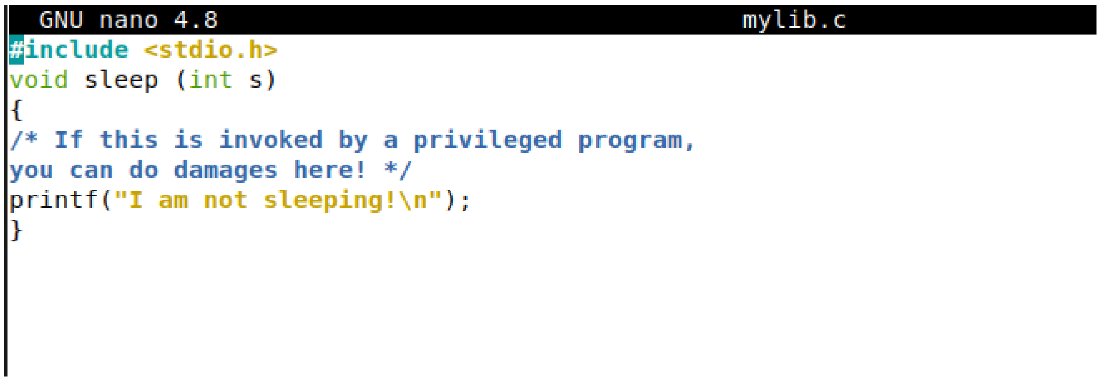
As a result, the system() function will launch a shell that will search the PATH environment variable's given location for the ls program. Therefore, as it is a root-owned SET-UID program, the attacker can run malicious code with root privileges by altering the PATH value to a folder containing a malicious file with the same name as that supplied in the program. Therefore, employing relative path and system function in SET-UID software may expose users to dangerous attacks.

Task 7: The LD PRELOAD Environment Variable and Set-UID Programs

Step 1. First, we will see how these environment variables influence the behavior of dynamic loader/linker when running a normal program. Please follow these steps:

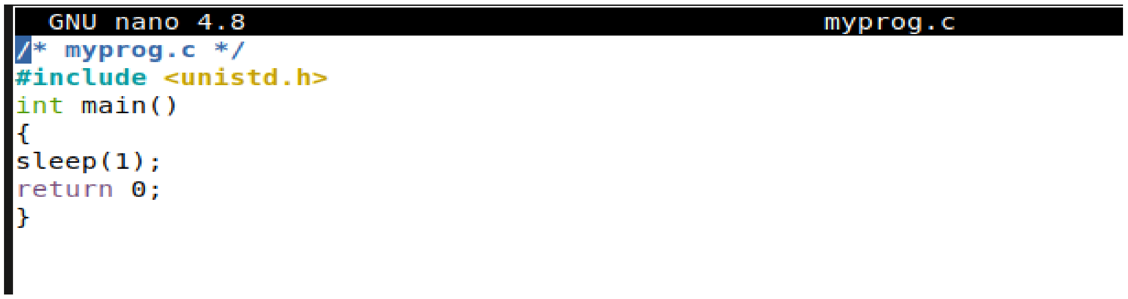
Regular Program, Normal User:

Edit and compile mylib.c



```
  GNU nano 4.8                                   mylib.c
#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");
}
```

Edit and compile myprog.c

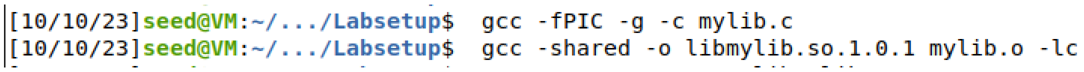


```
  GNU nano 4.8                                         myprog.c
/* myprog.c */
#include <unistd.h>
int main()
{
sleep(1);
return 0;
}
```

Compile the file using:

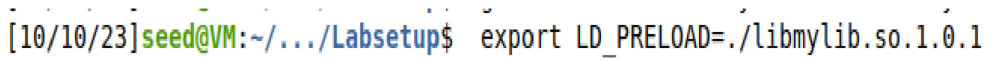
gcc -fPIC -g -c mylib.c (where -fPIC is emit position-independent code, suitable for dynamic linking and avoiding any limit on the size of the global offset table, -g is producing debugging information and -c iscompiling the file but not linking it.)

gcc -shared -o filename mylib.o -lc (where -shared produces a shared object that can be linked to other objects to form a executable, -o file stores the output in file.)



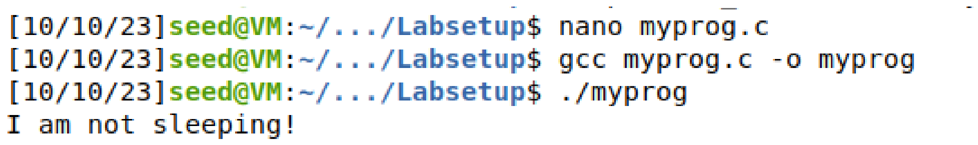
```
[10/10/23]seed@VM:~/.../Labsetup$  gcc -fPIC -g -c mylib.c
[10/10/23]seed@VM:~/.../Labsetup$  gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
```

This executable output file is then mentioned as the value of the LD_PRELOAD variable. This forces all programs to load this library before running.



```
[10/10/23]seed@VM:~/.../Labsetup$  export LD_PRELOAD=./libmylib.so.1.0.1
```

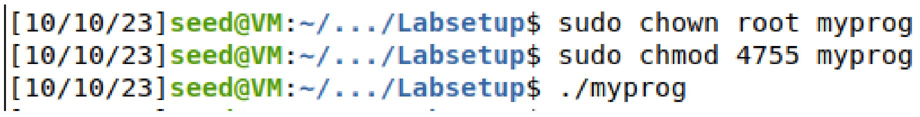
When this program is executed as a normal user, we can see that it runs the sleep function we defined and outputs the statement we defined in that function:



```
[10/10/23]seed@VM:~/.../Labsetup$ nano myprog.c
[10/10/23]seed@VM:~/.../Labsetup$ gcc myprog.c -o myprog
[10/10/23]seed@VM:~/.../Labsetup$ ./myprog
I am not sleeping!
```

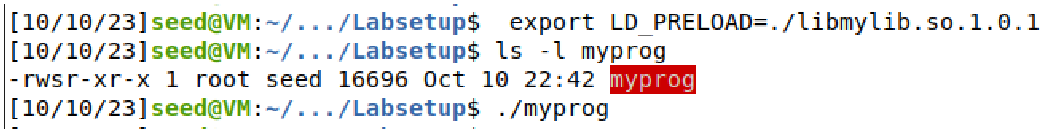
Step 2. After you have done the above, please run myprog under the following conditions, and observe what happens.

Change the owner effective user of the myprog program to root and set it to the SET_UID program.



```
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown root myprog
[10/10/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 myprog
[10/10/23]seed@VM:~/.../Labsetup$ ./myprog
```

Run the program for about 1 second, but there is no output. The system-defined sleep function was used instead of the library containing my sleep function.



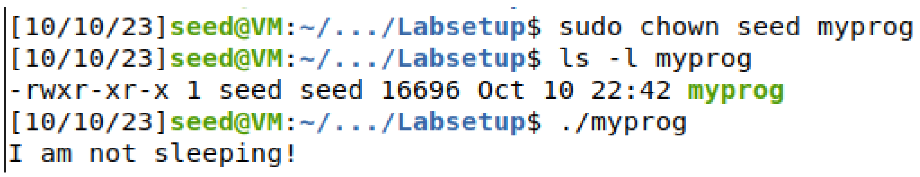
```
[10/10/23]seed@VM:~/.../Labsetup$  export LD_PRELOAD=./libmylib.so.1.0.1
[10/10/23]seed@VM:~/.../Labsetup$ ls -l myprog
-rwsr-xr-x 1 root seed 16696 Oct 10 22:42 myprog
[10/10/23]seed@VM:~/.../Labsetup$ ./myprog
```

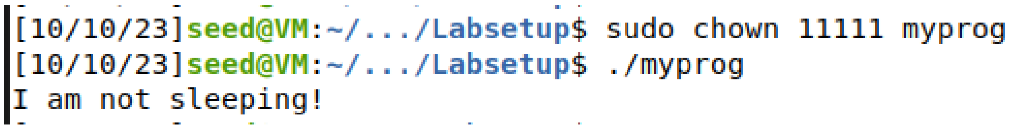
I didn't need to log in as the root user account because the application is already a SET-UID root program; I merely did that to set the LD_PRELOAD variable. When we run the program, we can see that the LD_PRELOAD variable is there and that the user-defined sleep function has been called. This occurs as a result of the fact that we are logged in as root and that root is also the function's owner. The LD_PRELOAD variable is kept because the process now has the same real ID and effective ID.

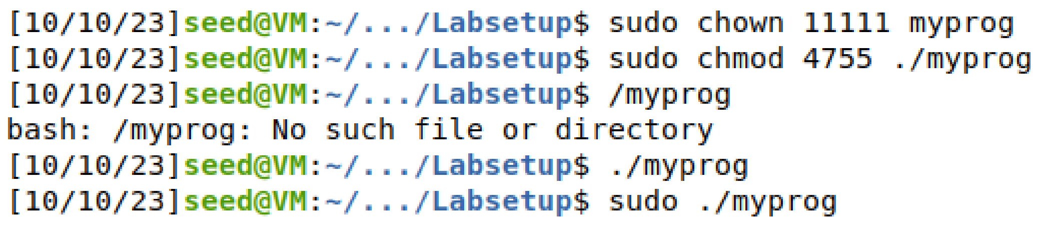
Step 3. You should be able to observe different behaviors in the scenarios described above, even though you are running the same program. You need to figure out what causes the difference. Environment variables play a role here. Please design an experiment to figure out the main causes, and explain why the behaviors in Step 2 are different. (Hint: the child process may not inherit the LD * environment variables).

Set-UID root program and export LD_PRELOAD env and root account.

The owner of this file will now be seed (a user account other than root), and it will be a SET-UID program. We next enter the seed's account and modify the LD_PRELOAD variable once more. Rerunning the application reveals that the LD_PRELOAD variable is present in the current process and that the user-defined sleep function is also called.


```
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown seed myprog
[10/10/23]seed@VM:~/.../Labsetup$ ls -l myprog
-rwxr-xr-x 1 seed seed 16696 Oct 10 22:42 myprog
[10/10/23]seed@VM:~/.../Labsetup$ ./myprog
I am not sleeping!
```


```
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown 11111 myprog
[10/10/23]seed@VM:~/.../Labsetup$ ./myprog
I am not sleeping!
```

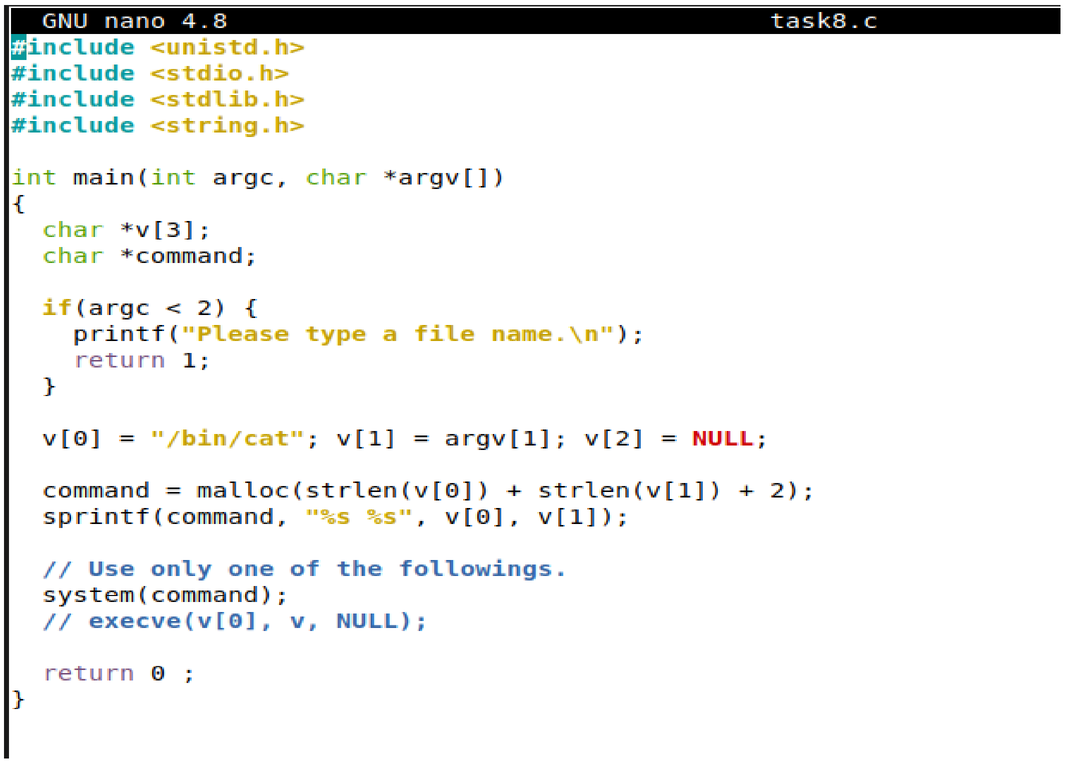

```
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown 11111 myprog
[10/10/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 ./myprog
[10/10/23]seed@VM:~/.../Labsetup$ /myprog
bash: /myprog: No such file or directory
[10/10/23]seed@VM:~/.../Labsetup$ ./myprog
[10/10/23]seed@VM:~/.../Labsetup$ sudo ./myprog
```

According to this behavior, the LD_PRELOAD variable is present while the effective ID and real ID are the same and is removed when they are different. This is because of the security feature of the SET-UID application. In the first, third, and fourth cases, the user-defined library was preloaded since the LD_PRELOAD variable was always present because the owner and the account executing the file were the same. While the LD_PRELOAD variable was discarded and the system-defined sleep function was called in the second scenario, where the effective ID was that of root and the real ID was that of seed.

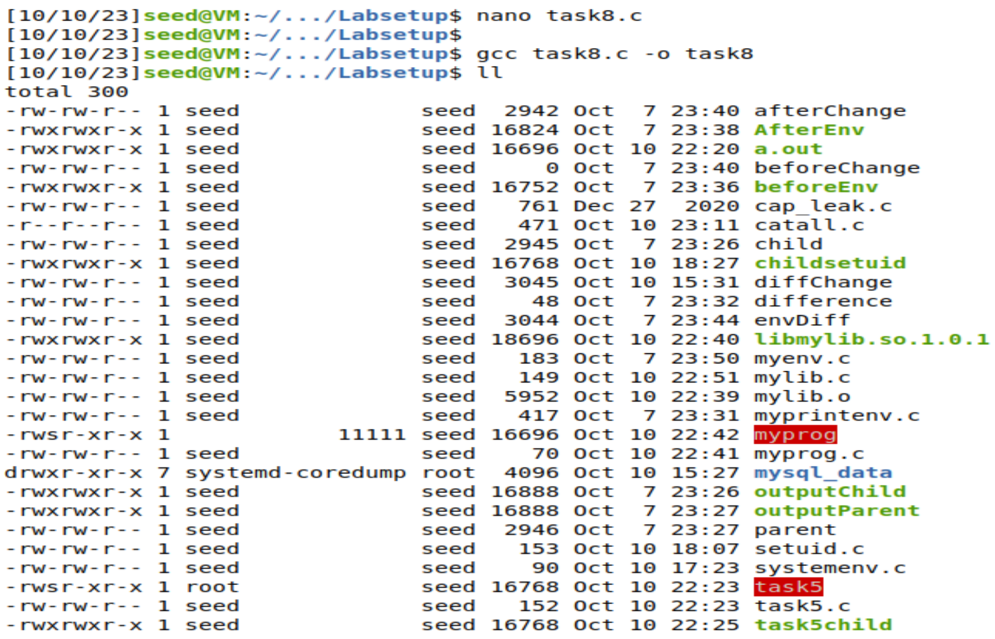
Task 8: Invoking External Programs Using system() versus execve()

Step 1: Compile the above program, make it a root-owned Set-UID program. The program will use system() to invoke the command. If you were Bob, can you compromise the integrity of the system? For example, can you remove a file that is not writable to you?

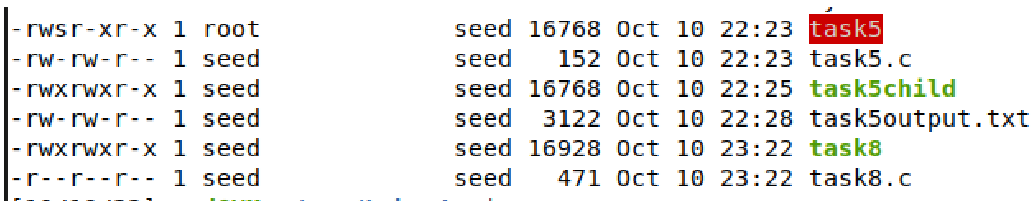
Edit and compile the file task8.c prograrm. The file is converted into a root-owned SET-UID program with executable permission to other users:



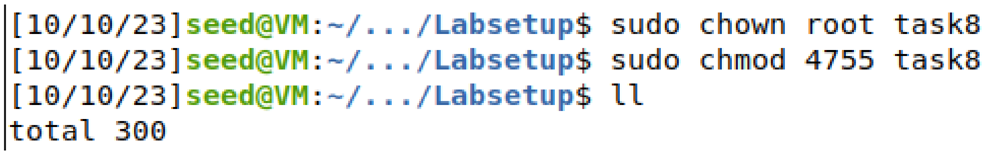
```
  GNU nano 4.8                                              task8.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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 ;
}
```



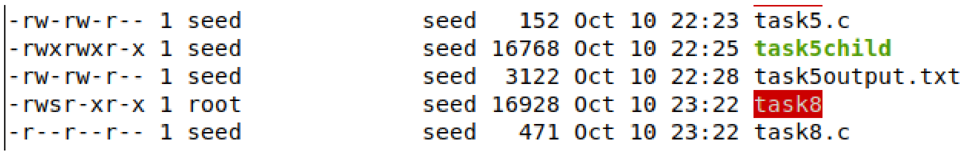
```
[10/10/23]seed@VM:~/.../Labsetup$ nano task8.c
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ gcc task8.c -o task8
[10/10/23]seed@VM:~/.../Labsetup$ ll
total 300
-rw-rw-r-- 1 seed                seed  2942 Oct  7 23:40 afterChange
-rwxrwxr-x 1 seed                seed 16824 Oct  7 23:38 AfterEnv
-rwxrwxr-x 1 seed                seed 16696 Oct 10 22:20 a.out
-rw-rw-r-- 1 seed                seed     0 Oct  7 23:40 beforeChange
-rwxrwxr-x 1 seed                seed 16752 Oct  7 23:36 beforeEnv
-rw-rw-r-- 1 seed                seed   761 Dec 27  2020 cap_leak.c
-r--r--r-- 1 seed                seed   471 Oct 10 23:11 catall.c
-rw-rw-r-- 1 seed                seed  2945 Oct  7 23:26 child
-rwxrwxr-x 1 seed                seed 16768 Oct 10 18:27 childsetuid
-rw-rw-r-- 1 seed                seed  3045 Oct 10 15:31 diffChange
-rw-rw-r-- 1 seed                seed    48 Oct  7 23:32 difference
-rw-rw-r-- 1 seed                seed  3044 Oct  7 23:44 envDiff
-rwxrwxr-x 1 seed                seed 18696 Oct 10 22:40 libmylib.so.1.0.1
-rw-rw-r-- 1 seed                seed   183 Oct  7 23:50 myenv.c
-rw-rw-r-- 1 seed                seed   149 Oct 10 22:51 mylib.c
-rw-rw-r-- 1 seed                seed  5952 Oct 10 22:39 mylib.o
-rw-rw-r-- 1 seed                seed   417 Oct  7 23:31 myprintenv.c
-rwsr-xr-x 1               11111 seed 16696 Oct 10 22:42 myprog
-rw-rw-r-- 1 seed                seed    70 Oct 10 22:41 myprog.c
drwxr-xr-x 7 systemd-coredump    root  4096 Oct 10 15:27 mysql_data
-rwxrwxr-x 1 seed                seed 16888 Oct  7 23:26 outputChild
-rwxrwxr-x 1 seed                seed 16888 Oct  7 23:27 outputParent
-rw-rw-r-- 1 seed                seed  2946 Oct  7 23:27 parent
-rw-rw-r-- 1 seed                seed   153 Oct 10 18:07 setuid.c
-rw-rw-r-- 1 seed                seed    90 Oct 10 17:23 systemenv.c
-rwsr-xr-x 1 root                seed 16768 Oct 10 22:23 task5
-rw-rw-r-- 1 seed                seed   152 Oct 10 22:23 task5.c
-rwxrwxr-x 1 seed                seed 16768 Oct 10 22:25 task5child
```



```
-rwsr-xr-x 1 root          seed 16768 Oct 10 22:23 task5
-rw-rw-r-- 1 seed          seed   152 Oct 10 22:23 task5.c
-rwxrwxr-x 1 seed          seed 16768 Oct 10 22:25 task5child
-rw-rw-r-- 1 seed          seed  3122 Oct 10 22:28 task5output.txt
-rwxrwxr-x 1 seed          seed 16928 Oct 10 23:22 task8
-r--r--r-- 1 seed          seed   471 Oct 10 23:22 task8.c
```

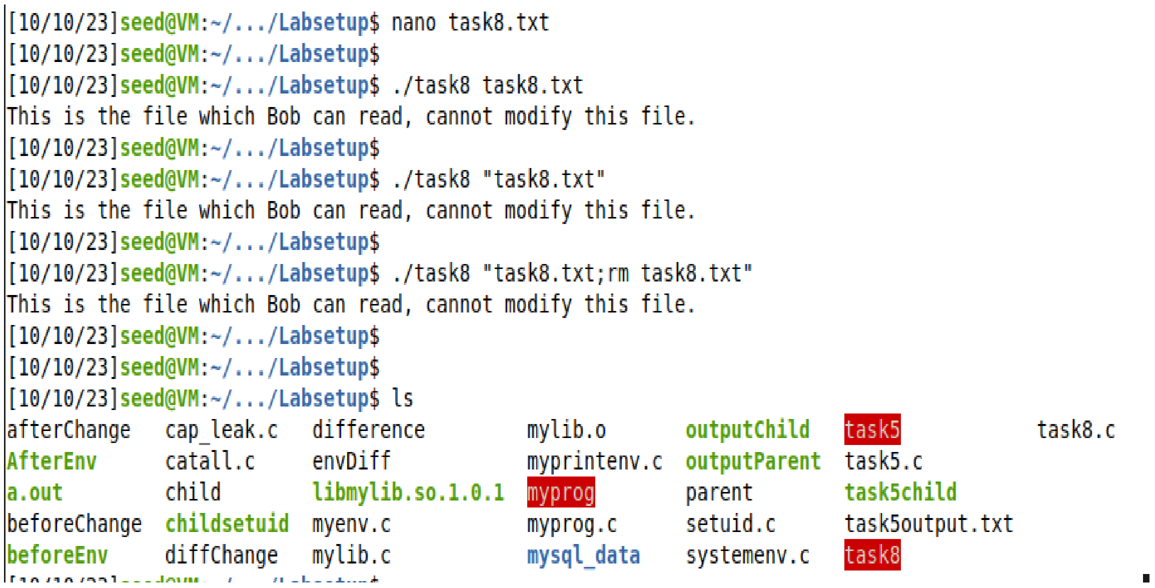


```
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown root task8
[10/10/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 task8
[10/10/23]seed@VM:~/.../Labsetup$ ll
total 300
```



```
-rw-rw-r-- 1 seed          seed   152 Oct 10 22:23 task5.c
-rwxrwxr-x 1 seed          seed 16768 Oct 10 22:25 task5child
-rw-rw-r-- 1 seed          seed  3122 Oct 10 22:28 task5output.txt
-rwsr-xr-x 1 root          seed 16928 Oct 10 23:22 task8
-r--r--r-- 1 seed          seed   471 Oct 10 23:22 task8.c
```

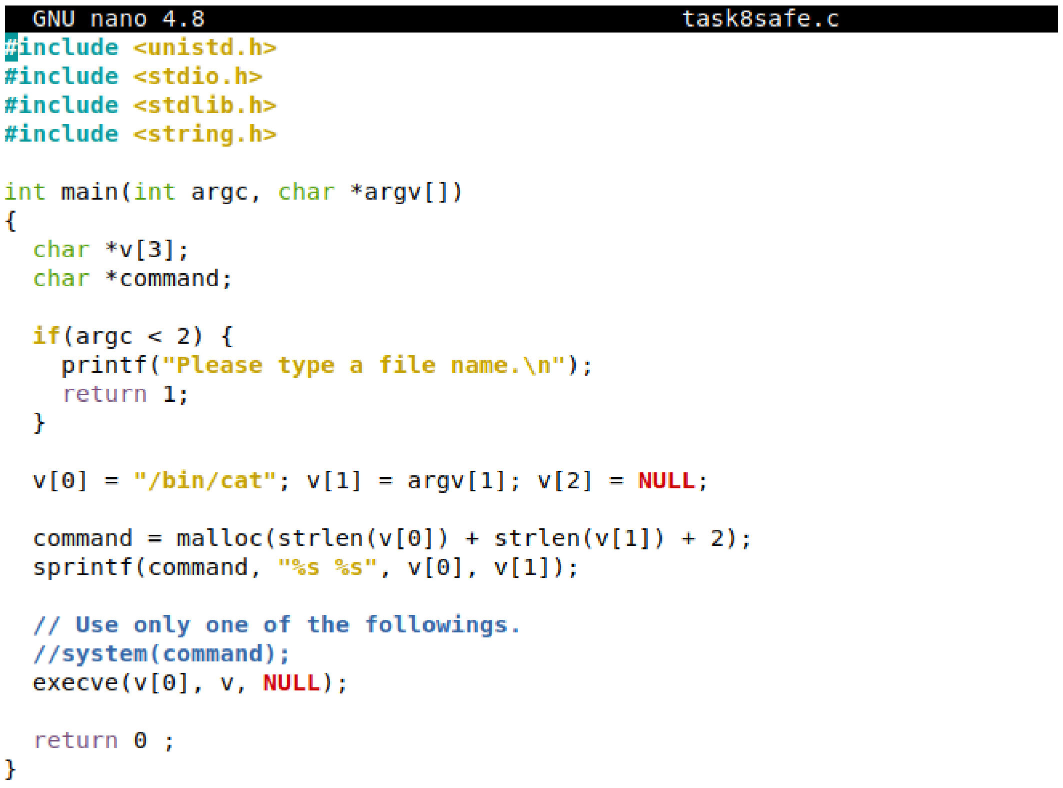
On executing the file, the normal functionality will output the file to task8.txt.



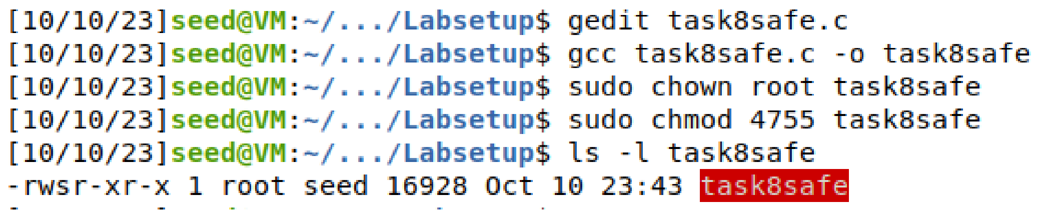
```
[10/10/23]seed@VM:~/.../Labsetup$ nano task8.txt
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ ./task8 task8.txt
This is the file which Bob can read, cannot modify this file.
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ ./task8 "task8.txt"
This is the file which Bob can read, cannot modify this file.
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ ./task8 "task8.txt;rm task8.txt"
This is the file which Bob can read, cannot modify this file.
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ ls
afterChange   cap_leak.c   difference        mylib.o       outputChild    task5            task8.c
AfterEnv      catall.c     envDiff           myprintenv.c  outputParent   task5.c
a.out         child        libmylib.so.1.0.1 myprog        parent         task5child
beforeChange  childsetuid  myenv.c           myprog.c      setuid.c       task5output.txt
beforeEnv     diffChange   mylib.c           mysql_data    systemenv.c    task8
```

Step 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. Do your attacks in Step 1 still work? Please describe and explain your observations.

Edit the file task8safe.c and comment the system statement and uncomment the execve statement.

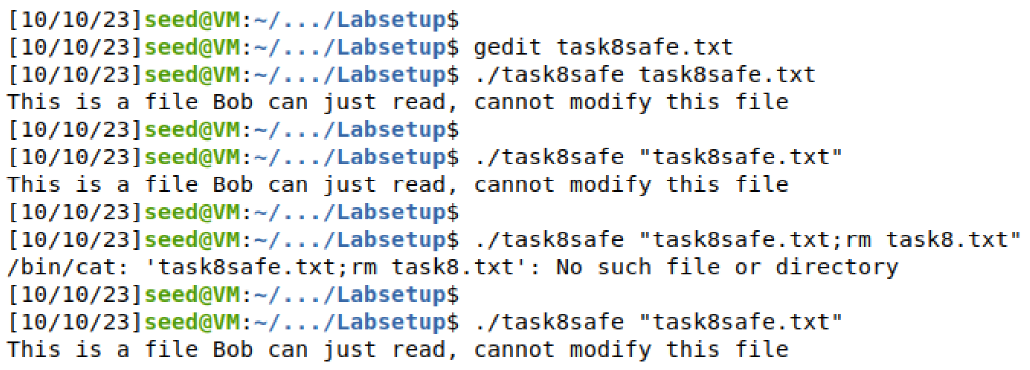


```
  GNU nano 4.8                              task8safe.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

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 ;
}
```



```
[10/10/23]seed@VM:~/.../Labsetup$ gedit task8safe.c
[10/10/23]seed@VM:~/.../Labsetup$ gcc task8safe.c -o task8safe
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown root task8safe
[10/10/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 task8safe
[10/10/23]seed@VM:~/.../Labsetup$ ls -l task8safe
-rwsr-xr-x 1 root seed 16928 Oct 10 23:43 task8safe
```

Run the program change its effective user, and set task8safe.c to set the SET_UID program. Run the program and find that it can be executed normally.

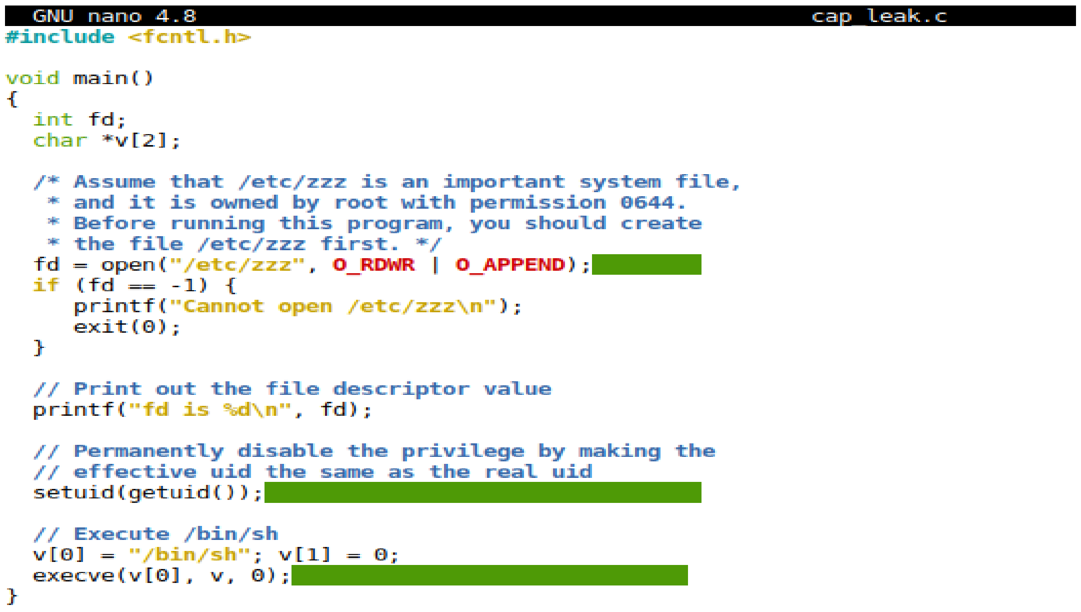


```
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ gedit task8safe.txt
[10/10/23]seed@VM:~/.../Labsetup$ ./task8safe task8safe.txt
This is a file Bob can just read, cannot modify this file
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ ./task8safe "task8safe.txt"
This is a file Bob can just read, cannot modify this file
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ ./task8safe "task8safe.txt;rm task8.txt"
/bin/cat: 'task8safe.txt;rm task8.txt': No such file or directory
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ ./task8safe "task8safe.txt"
This is a file Bob can just read, cannot modify this file
```

Use the method Step 1 to delete the file. It is found that it cannot be deleted successfully and it shows that the file or directory does not exist.

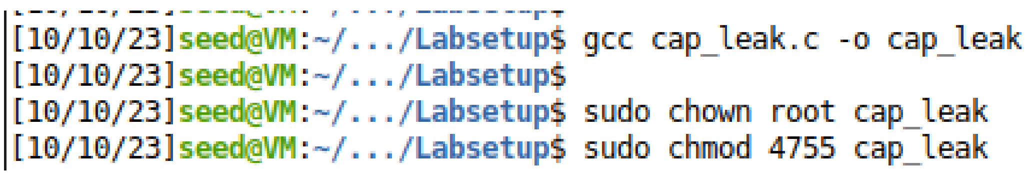
The execve() function essentially executes a command. At this time only one process can be executed as a command. This is the reason failing if the deletion. The code and the date are isolated.

Task 9: Capability Leaking

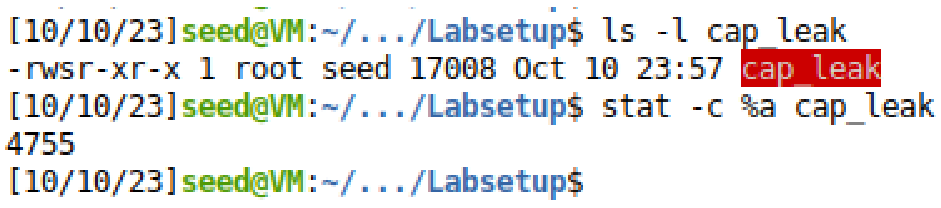


```
GNU nano 4.8                                                    cap_leak.c
#include <fcntl.h>

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

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

Edit and compile the cap_leak file.

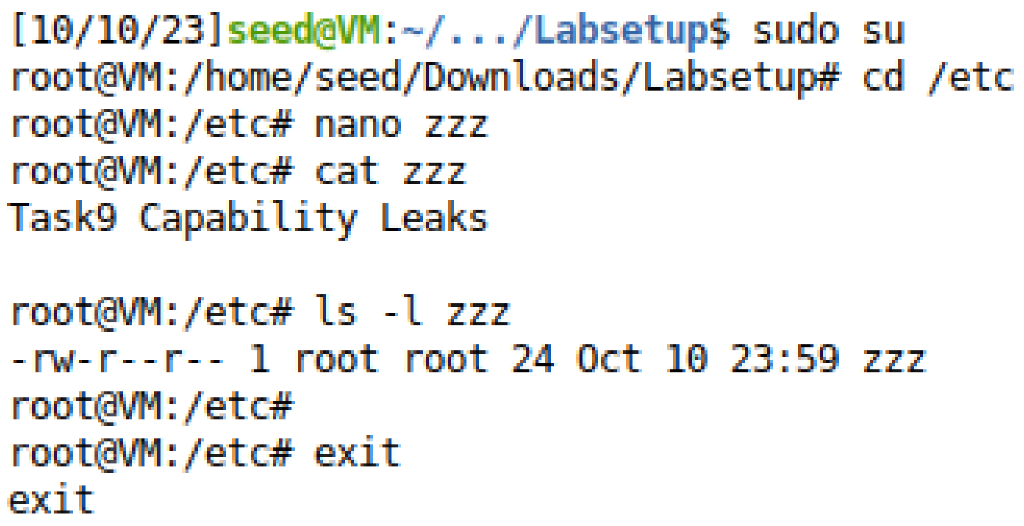


```
[10/10/23]seed@VM:~/.../Labsetup$ gcc cap_leak.c -o cap_leak
[10/10/23]seed@VM:~/.../Labsetup$
[10/10/23]seed@VM:~/.../Labsetup$ sudo chown root cap_leak
[10/10/23]seed@VM:~/.../Labsetup$ sudo chmod 4755 cap_leak
```



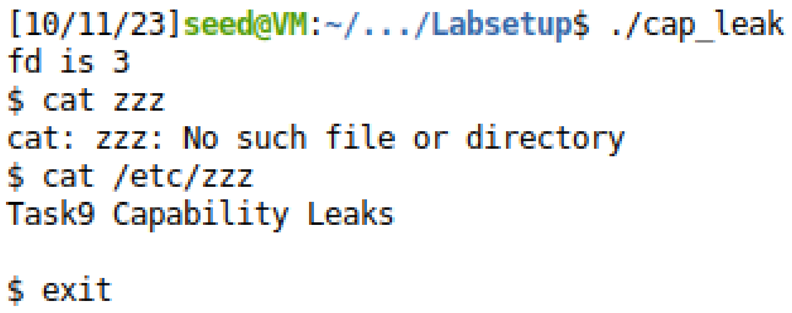
```
[10/10/23]seed@VM:~/.../Labsetup$ ls -l cap_leak
-rwsr-xr-x 1 root seed 17008 Oct 10 23:57 cap_leak
[10/10/23]seed@VM:~/.../Labsetup$ stat -c %a cap_leak
4755
[10/10/23]seed@VM:~/.../Labsetup$
```

Here, we create a file named zzz in the /etc folder containing a print statement in the main:



```
[10/10/23]seed@VM:~/.../Labsetup$ sudo su
root@VM:/home/seed/Downloads/Labsetup# cd /etc
root@VM:/etc# nano zzz
root@VM:/etc# cat zzz
Task9 Capability Leaks

root@VM:/etc# ls -l zzz
-rw-r--r-- 1 root root 24 Oct 10 23:59 zzz
root@VM:/etc#
root@VM:/etc# exit
exit
```

Then we run the application again to view the zzz file's content, and this time we can see that it has been altered. This occurs because, despite the fact that we lowered the rights in the program, we failed to close the file at the appropriate time. As a result, the file continued to run with privileged permissions, allowing data to be edited even without the proper permissions. Here, fork is called, control is then transferred to the child process, allowing the malicious user to successfully change the contents of a protected file. This demonstrates the significance of closing the file descriptor after removing privileges so that it has the proper permissions.



```
[10/11/23]seed@VM:~/.../Labsetup$ ./cap_leak
fd is 3
$ cat zzz
cat: zzz: No such file or directory
$ cat /etc/zzz
Task9 Capability Leaks

$ exit
```