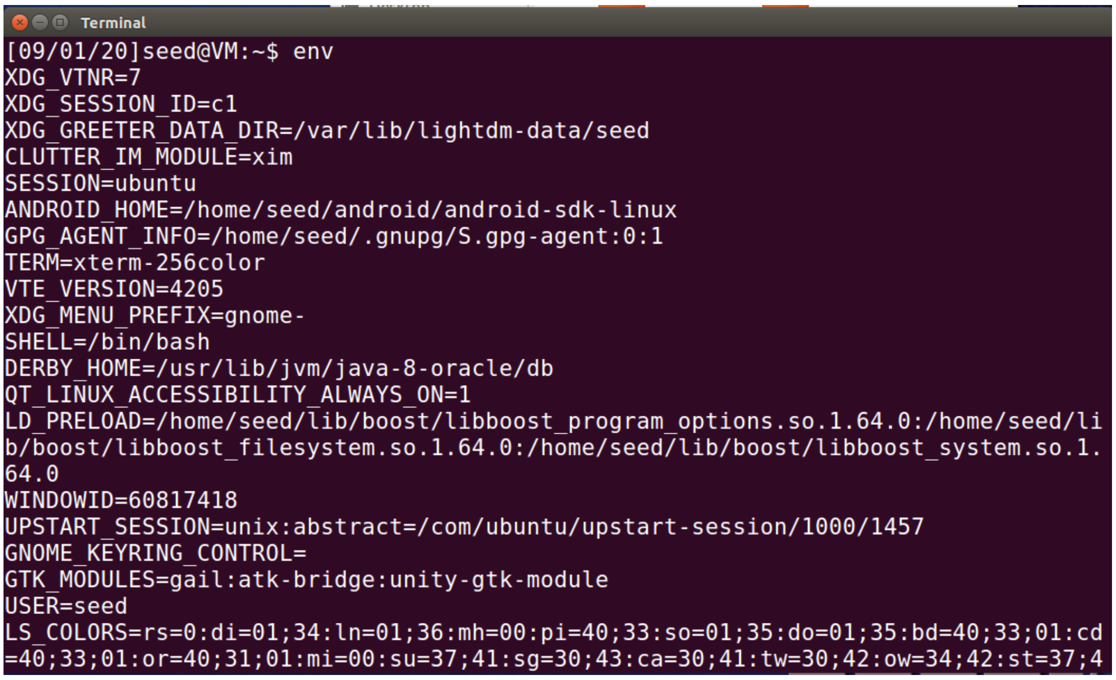
网安综合课程设计实验报告 1

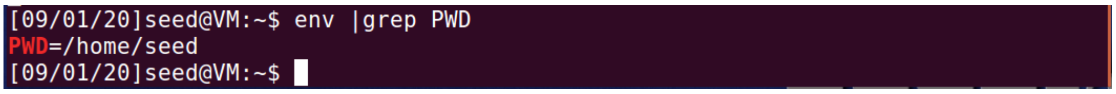
实验 1：对环境变量的基本操作

1> env 显示全部环境变量，或是 printenv，效果相同



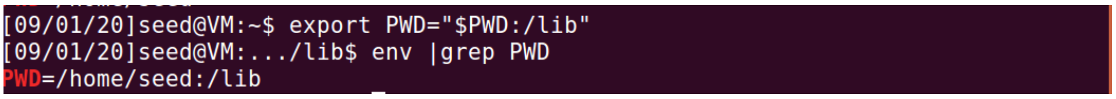
```
[09/01/20]seed@VM:~$ env
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/li
b/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.
64.0
WINDOWID=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1457
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
```

2> 筛选某些字符：env |grep PWD



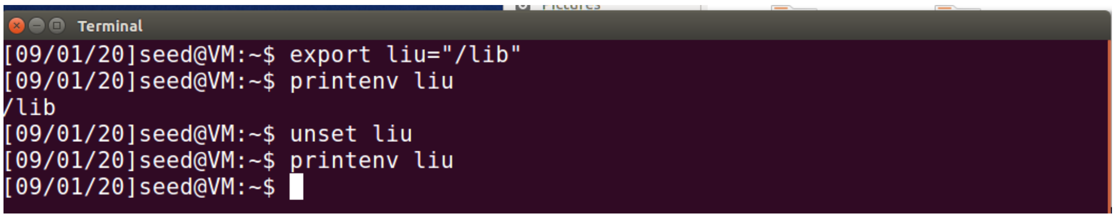
```
[09/01/20]seed@VM:~$ env |grep PWD
PWD=/home/seed
[09/01/20]seed@VM:~$
```

3> export 增加环境变量



```
[09/01/20]seed@VM:~$ export PWD="$PWD:/lib"
[09/01/20]seed@VM:.../lib$ env |grep PWD
PWD=/home/seed:/lib
```

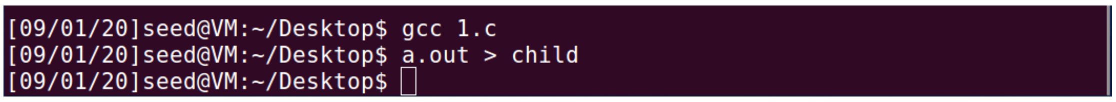
4> unset 取消环境变量



```
[09/01/20]seed@VM:~$ export liu="/lib"
[09/01/20]seed@VM:~$ printenv liu
/lib
[09/01/20]seed@VM:~$ unset liu
[09/01/20]seed@VM:~$ printenv liu
[09/01/20]seed@VM:~$
```

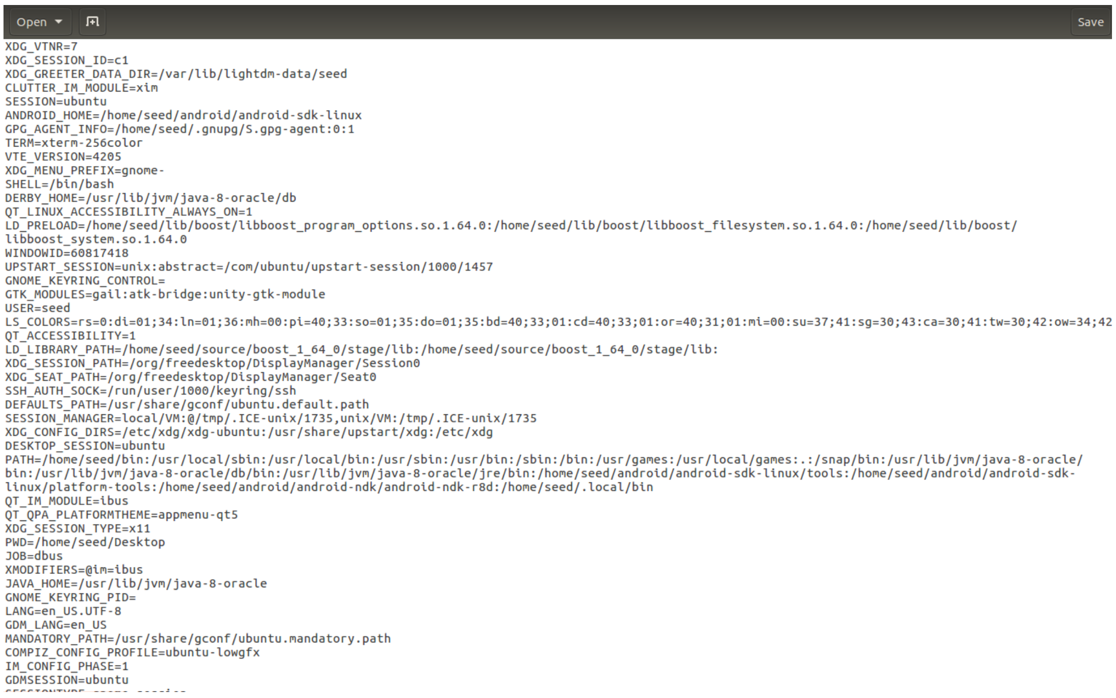
实验 2：父进程到子进程传递环境变量

1> 子进程的输出情况



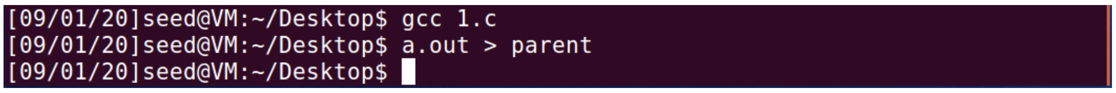
```
[09/01/20]seed@VM:~/Desktop$ gcc 1.c
[09/01/20]seed@VM:~/Desktop$ a.out > child
[09/01/20]seed@VM:~/Desktop$
```

Child 中的信息如下：



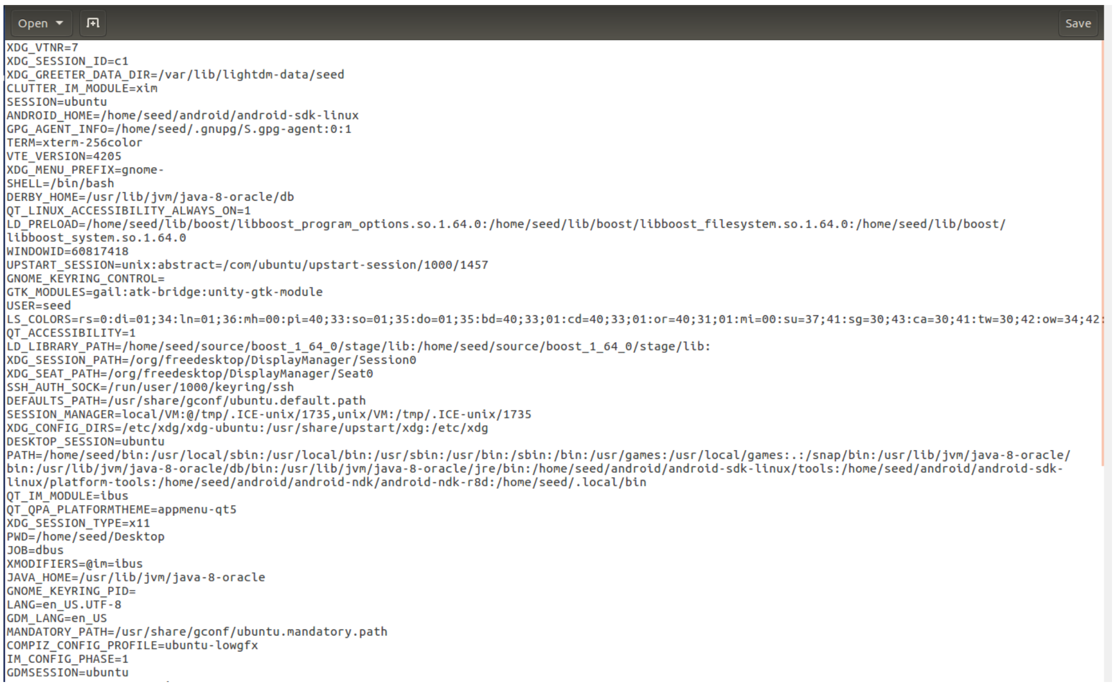
```
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/
libboost_system.so.1.64.0
WINDOWID=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1457
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42
QT_ACCESSIBILITY=1
LD_LIBRARY_PATH=/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
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1735,unix/VM:/tmp/.ICE-unix/1735
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
DESKTOP_SESSION=ubuntu
PATH=/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
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
PWD=/home/seed/Desktop
JOB=dbus
XMODIFIERS=@im=ibus
JAVA_HOME=/usr/lib/jvm/java-8-oracle
GNOME_KEYRING_PID=
LANG=en_US.UTF-8
GDM_LANG=en_US
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
IM_CONFIG_PHASE=1
GDMSESSION=ubuntu
```

2> 父进程的输出情况



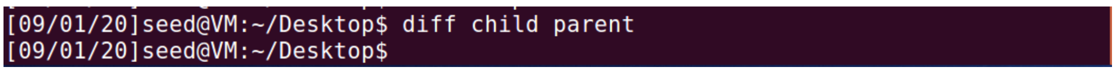
```
[09/01/20]seed@VM:~/Desktop$ gcc 1.c
[09/01/20]seed@VM:~/Desktop$ a.out > parent
[09/01/20]seed@VM:~/Desktop$ 
```

Parent 文件输出如下：



```
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/
libboost_system.so.1.64.0
WINDOWID=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1457
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:
QT_ACCESSIBILITY=1
LD_LIBRARY_PATH=/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
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1735,unix/VM:/tmp/.ICE-unix/1735
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/usr/share/upstart/xdg:/etc/xdg
DESKTOP_SESSION=ubuntu
PATH=/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
QT_IM_MODULE=ibus
QT_QPA_PLATFORMTHEME=appmenu-qt5
XDG_SESSION_TYPE=x11
PWD=/home/seed/Desktop
JOB=dbus
XMODIFIERS=@im=ibus
JAVA_HOME=/usr/lib/jvm/java-8-oracle
GNOME_KEYRING_PID=
LANG=en_US.UTF-8
GDM_LANG=en_US
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
IM_CONFIG_PHASE=1
GDMSESSION=ubuntu
```

3> 结果分析：

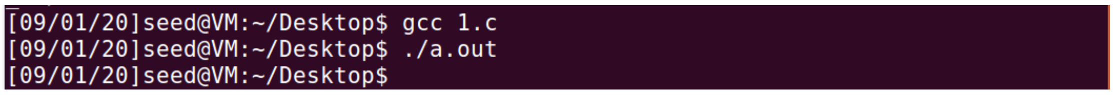


```
[09/01/20]seed@VM:~/Desktop$ diff child parent
[09/01/20]seed@VM:~/Desktop$
```

结论：子进程继承之后与父进程保留相同的环境变量。

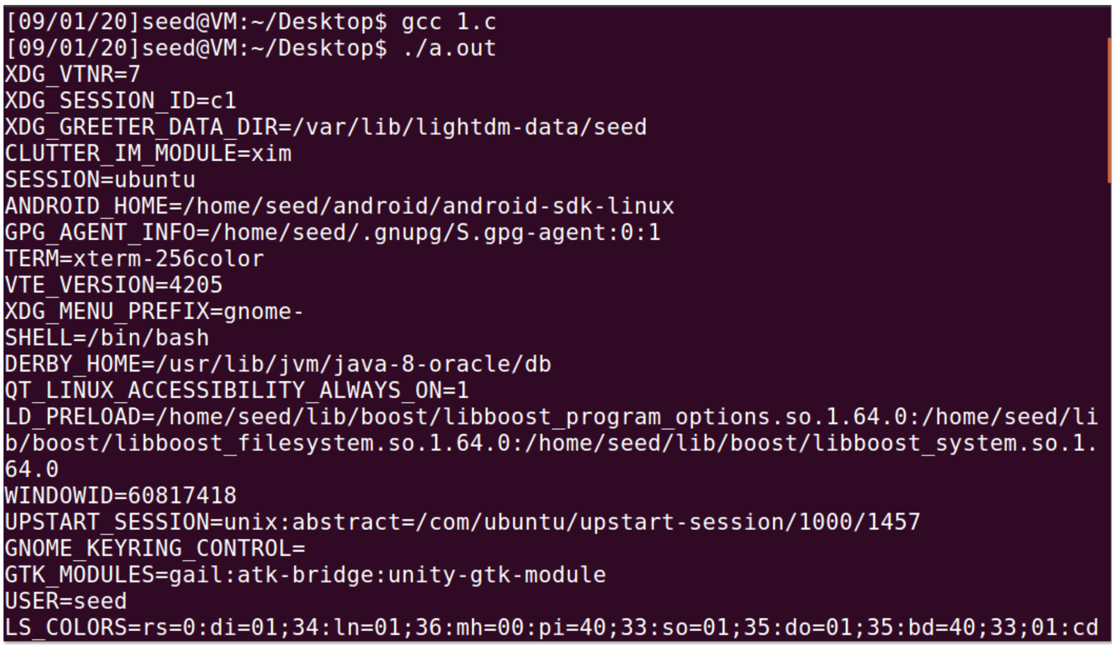
实验 3：环境变量与 execve()

1> 编译程序(参数为 NULL)



```
[09/01/20]seed@VM:~/Desktop$ gcc 1.c
[09/01/20]seed@VM:~/Desktop$ ./a.out
[09/01/20]seed@VM:~/Desktop$
```

2> 传入环境变量



```
[09/01/20]seed@VM:~/Desktop$ gcc 1.c
[09/01/20]seed@VM:~/Desktop$ ./a.out
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/li
b/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.
64.0
WINDOWID=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1457
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
```

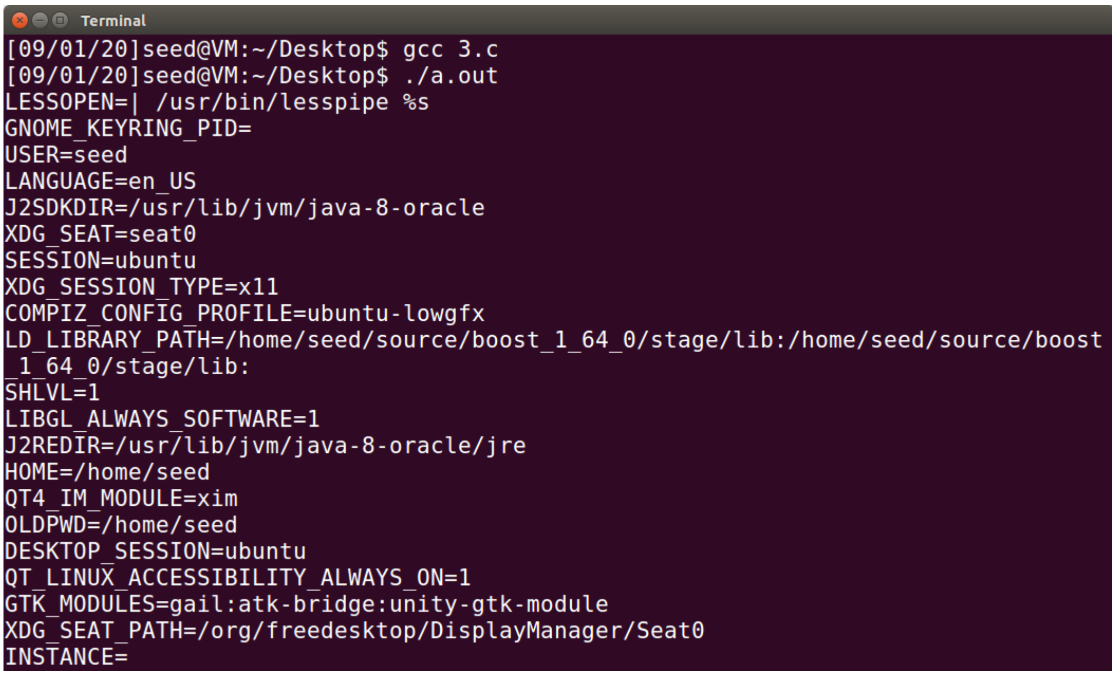
(省略下面环境变量的输出)

3> 总结结论

当不传入环境变量时.execve()是不会继承原进程的环境变量的,只有当将环境变量作为参数导入函数才会改变环境变量.

实验 4: 环境变量和 system()

1> 实验部分



```
[09/01/20]seed@VM:~/Desktop$ gcc 3.c
[09/01/20]seed@VM:~/Desktop$ ./a.out
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
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
OLDPWD=/home/seed
DESKTOP_SESSION=ubuntu
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
GTK_MODULES=gail:atk-bridge:unity-gtk-module
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
INSTANCE=
```

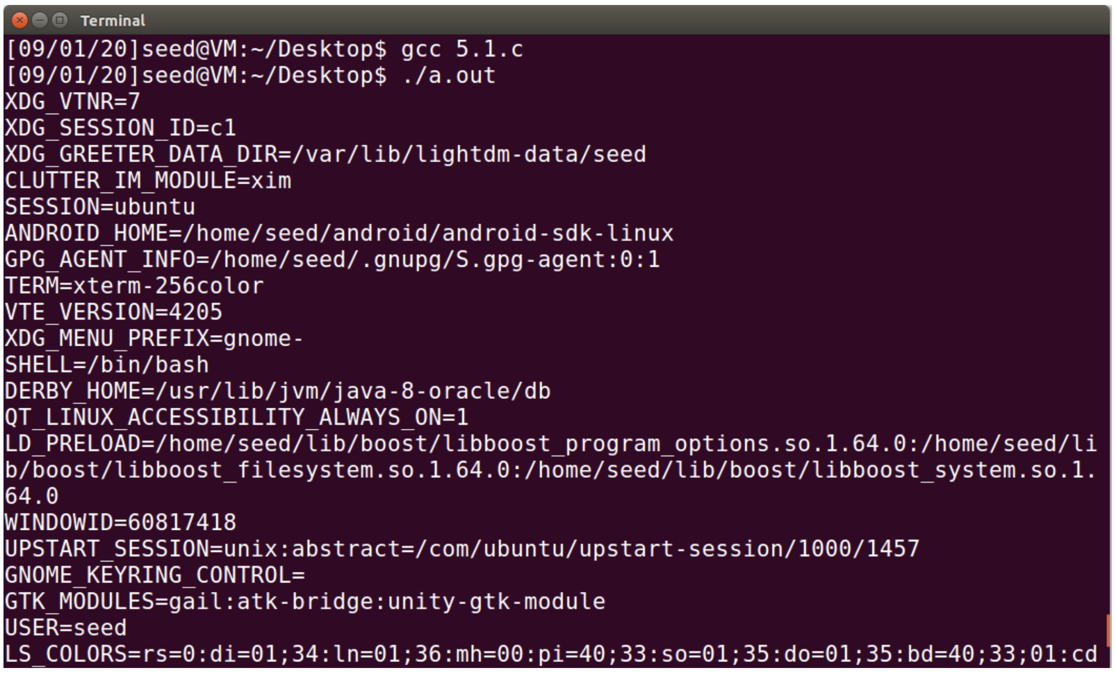
(省略后续环境变量的输出)

2> 结论总结

System()函数会将新的进程继承于原进程的环境变量,与 execve()不同,有较大的安全隐患.

实验 5: set-uid 程序

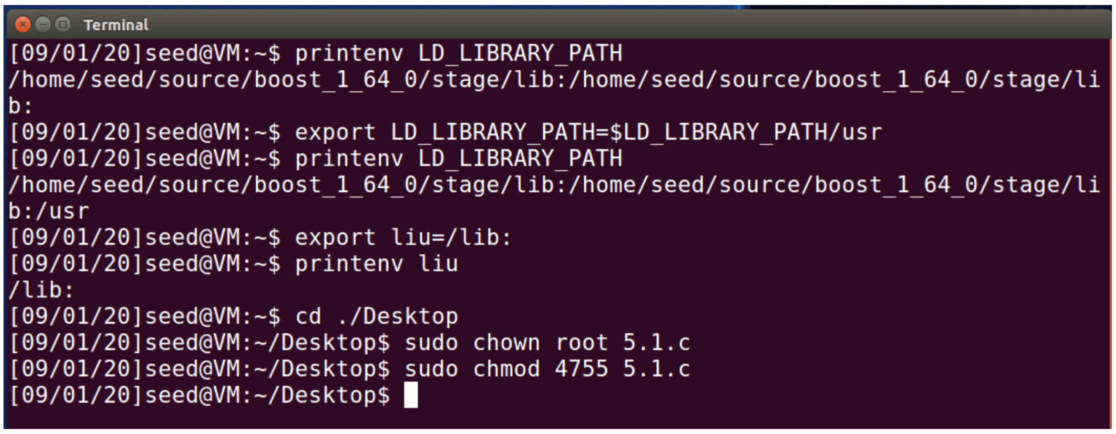
1> 编写环境变量输出程序并运行



```
[09/01/20]seed@VM:~/Desktop$ gcc 5.1.c
[09/01/20]seed@VM:~/Desktop$ ./a.out
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/li
b/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.
64.0
WINDOWID=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1457
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
```

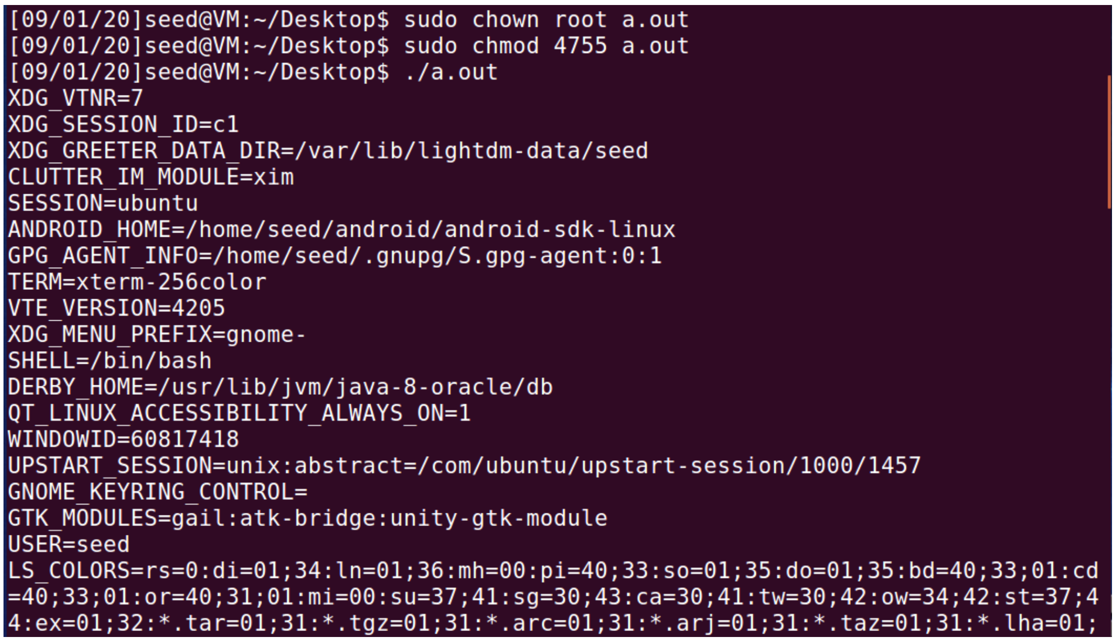
(省略后面输出的环境变量)

2> 更改属性与添加两条路径



```
[09/01/20]seed@VM:~$ printenv LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/li
b:
[09/01/20]seed@VM:~$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH/usr
[09/01/20]seed@VM:~$ printenv LD_LIBRARY_PATH
/home/seed/source/boost_1_64_0/stage/lib:/home/seed/source/boost_1_64_0/stage/li
b:/usr
[09/01/20]seed@VM:~$ export liu=/lib:
[09/01/20]seed@VM:~$ printenv liu
/lib:
[09/01/20]seed@VM:~$ cd ./Desktop
[09/01/20]seed@VM:~/Desktop$ sudo chown root 5.1.c
[09/01/20]seed@VM:~/Desktop$ sudo chmod 4755 5.1.c
[09/01/20]seed@VM:~/Desktop$ 
```

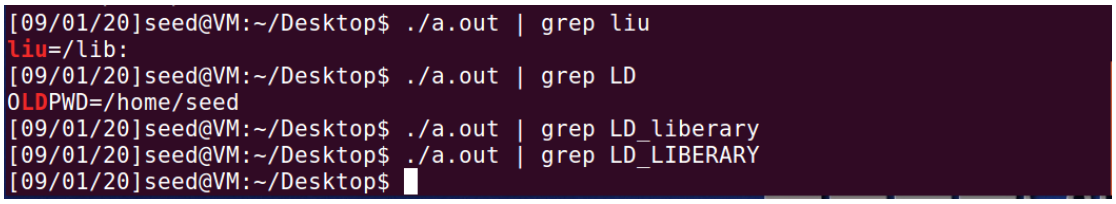
3> 运行新的程序



```
[09/01/20]seed@VM:~/Desktop$ sudo chown root a.out
[09/01/20]seed@VM:~/Desktop$ sudo chmod 4755 a.out
[09/01/20]seed@VM:~/Desktop$ ./a.out
XDG_VTNR=7
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm-256color
VTE_VERSION=4205
XDG_MENU_PREFIX=gnome-
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
WINDOWID=60817418
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/1457
GNOME_KEYRING_CONTROL=
GTK_MODULES=gail:atk-bridge:unity-gtk-module
USER=seed
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33;01:cd
=40;33;01:or=40;31;01:mi=00:su=37;41:sg=30;43:ca=30;41:tw=30;42:ow=34;42:st=37;4
4:ex=01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;
```

(省略环境变量输出)

4> 之前设置变量的输出情况



```
[09/01/20]seed@VM:~/Desktop$ ./a.out | grep liu
liu=/lib:
[09/01/20]seed@VM:~/Desktop$ ./a.out | grep LD
OLDPWD=/home/seed
[09/01/20]seed@VM:~/Desktop$ ./a.out | grep LD_liberary
[09/01/20]seed@VM:~/Desktop$ ./a.out | grep LD_LIBERARY
[09/01/20]seed@VM:~/Desktop$ 
```

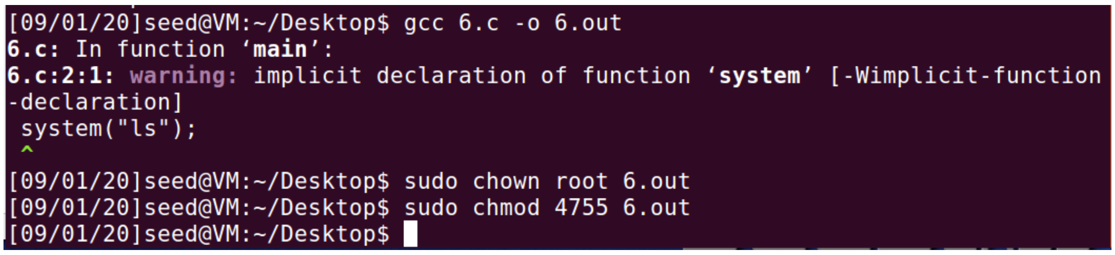
5> 结论

之前设置的 liu 环境变量被正常输出,而 LD_LIBRARY_PATH 比较奇怪,不仅没有输出改动之后的值,就连之前的值都没有输出,似乎是系统屏蔽删除了这一环境变量.这一原因

可能就是检测到了 set-uid 对其的修改从而自动删除了这一环境变量以保护系统安全.

实验 6: PATH 环境变量和 set-uid 程序

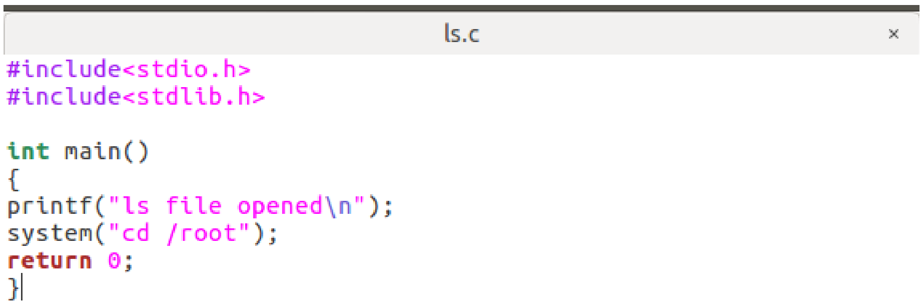
1> 设置 set-uid 程序



```
[09/01/20]seed@VM:~/Desktop$ gcc 6.c -o 6.out
6.c: In function ‘main’:
6.c:2:1: warning: implicit declaration of function ‘system’ [-Wimplicit-function
-declaration]
 system("ls");
 ^
[09/01/20]seed@VM:~/Desktop$ sudo chown root 6.out
[09/01/20]seed@VM:~/Desktop$ sudo chmod 4755 6.out
[09/01/20]seed@VM:~/Desktop$ 
```

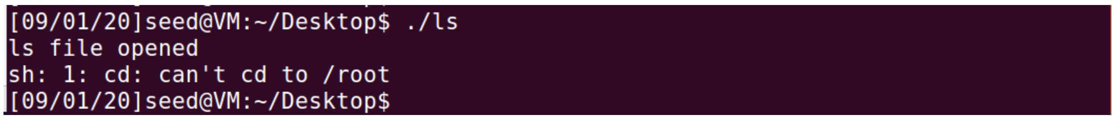
2> 编写 ls 文件并编译后放入/home/seed 下

Ls.c 文件如下:



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

int main()
{
printf("ls file opened\n");
system("cd /root");
return 0;
}
```

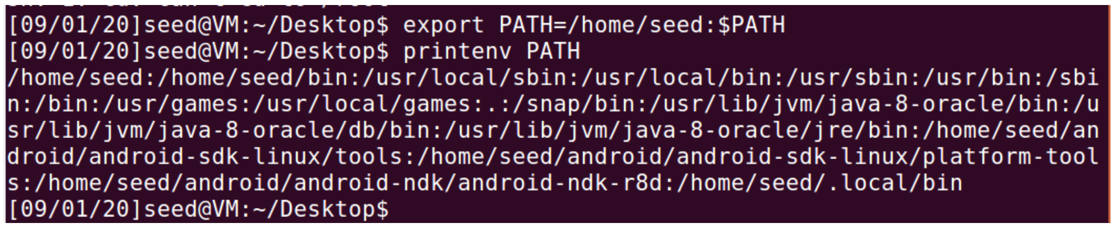
实现的功能:首先说明文件被调用正常打开,其次用 cd /root 命令来证明是否是 root 权限.在普通模式下编译运行的结果为:



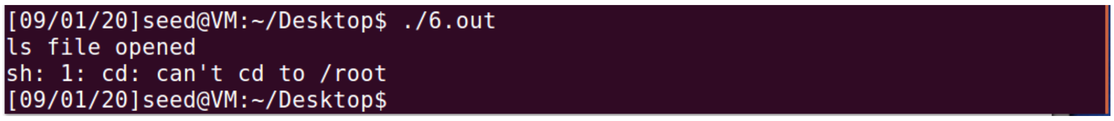
```
[09/01/20]seed@VM:~/Desktop$ ./ls
ls file opened
sh: 1: cd: can't cd to /root
[09/01/20]seed@VM:~/Desktop$
```

编译之后放入 seed 目录下.

3> 修改 PATH 路径,并运行 6.out,实现自己编写的 ls 被运行

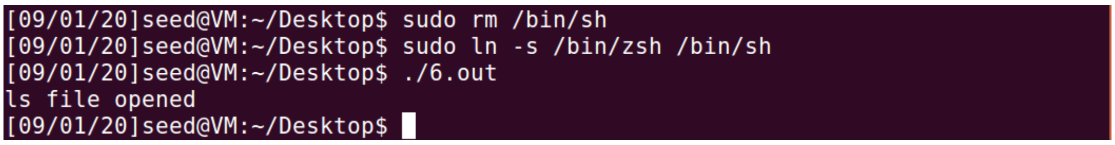


```
[09/01/20]seed@VM:~/Desktop$ export PATH=/home/seed:$PATH
[09/01/20]seed@VM:~/Desktop$ printenv PATH
/home/seed:/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbi
n:/bin:/usr/games:/usr/local/games:.:/snap/bin:/usr/lib/jvm/java-8-oracle/bin:/u
sr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm/java-8-oracle/jre/bin:/home/seed/an
droid/android-sdk-linux/tools:/home/seed/android/android-sdk-linux/platform-tool
s:/home/seed/android/android-ndk/android-ndk-r8d:/home/seed/.local/bin
[09/01/20]seed@VM:~/Desktop$
```



```
[09/01/20]seed@VM:~/Desktop$ ./6.out
ls file opened
sh: 1: cd: can't cd to /root
[09/01/20]seed@VM:~/Desktop$
```

成功运行了自己编写的 ls 文件,但是却没有获得 root 权限,原因是报告中给出的 dash 问题,经过修改后,实验结果为:

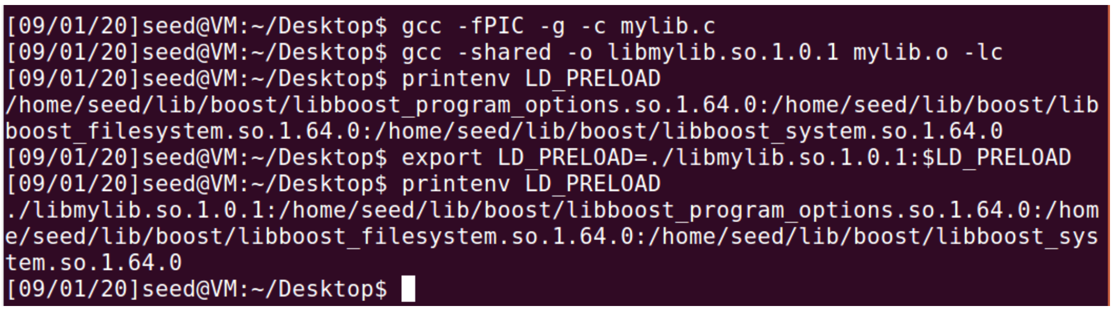


```
[09/01/20]seed@VM:~/Desktop$ sudo rm /bin/sh
[09/01/20]seed@VM:~/Desktop$ sudo ln -s /bin/zsh /bin/sh
[09/01/20]seed@VM:~/Desktop$ ./6.out
ls file opened
[09/01/20]seed@VM:~/Desktop$ 
```

成功获得了实验结果.

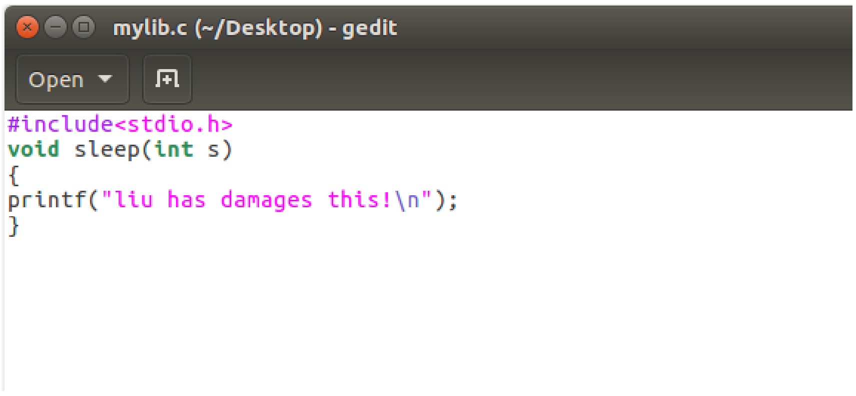
实验 7: LD_PRELOAD 与环境变量

1> 完成 libmylib.so.1.0.1 的动态链接库的生成



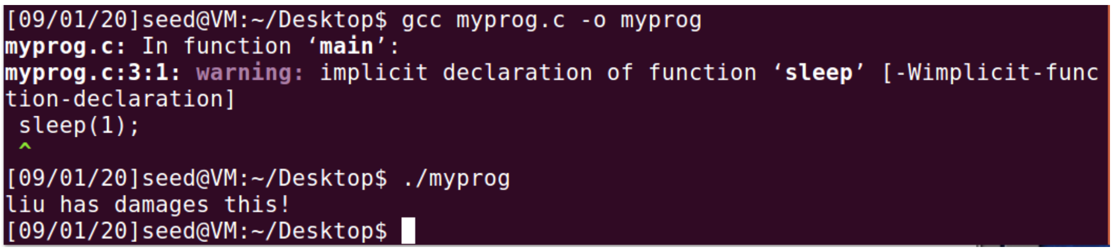
```
[09/01/20]seed@VM:~/Desktop$ gcc -fPIC -g -c mylib.c
[09/01/20]seed@VM:~/Desktop$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/01/20]seed@VM:~/Desktop$ printenv LD_PRELOAD
/home/seed/lib/boost/libboost_program_options.so.1.64.0:/home/seed/lib/boost/lib
boost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_system.so.1.64.0
[09/01/20]seed@VM:~/Desktop$ export LD_PRELOAD=./libmylib.so.1.0.1:$LD_PRELOAD
[09/01/20]seed@VM:~/Desktop$ printenv LD_PRELOAD
./libmylib.so.1.0.1:/home/seed/lib/boost/libboost_program_options.so.1.64.0:/hom
e/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib/boost/libboost_sys
tem.so.1.64.0
[09/01/20]seed@VM:~/Desktop$
```

其中,mylib.c 文件如下:



```
#include<stdio.h>
void sleep(int s)
{
printf("liu has damages this!\n");
}
```

2> 完成 myprog 的编译与链接动态链接

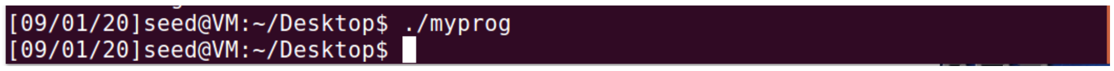


```
[09/01/20]seed@VM:~/Desktop$ gcc myprog.c -o myprog
myprog.c: In function 'main':
myprog.c:3:1: warning: implicit declaration of function 'sleep' [-Wimplicit-func
tion-declaration]
 sleep(1);
 ^
[09/01/20]seed@VM:~/Desktop$ ./myprog
liu has damages this!
[09/01/20]seed@VM:~/Desktop$
```

其中,myprog 同实验报告中所写.可以看到,实验取得了成功,自己编写的动态链接成功输出.但是可以注意到在编译的时候会有 warning 报出.

3> 在不同的环境中执行 myprog
正常程序普通用户结果如 2 所示;

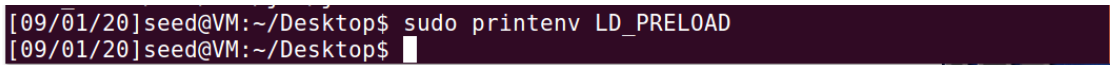
Set-uid 程序,普通用户结果如下:



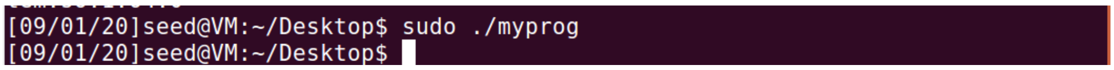
```
[09/01/20]seed@VM:~/Desktop$ ./myprog
[09/01/20]seed@VM:~/Desktop$
```

程序睡眠 1s

Root 账户中的 LD_PRELOAD 与运行结果如下:

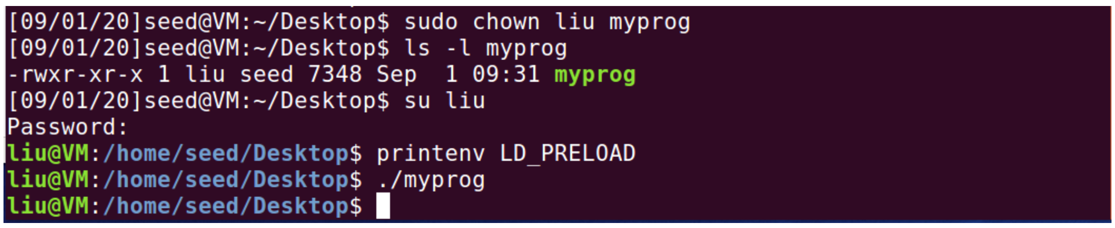


```
[09/01/20]seed@VM:~/Desktop$ sudo printenv LD_PRELOAD
[09/01/20]seed@VM:~/Desktop$ 
```



```
[09/01/20]seed@VM:~/Desktop$ sudo ./myprog
[09/01/20]seed@VM:~/Desktop$ 
```

其他账户的运行结果如下:



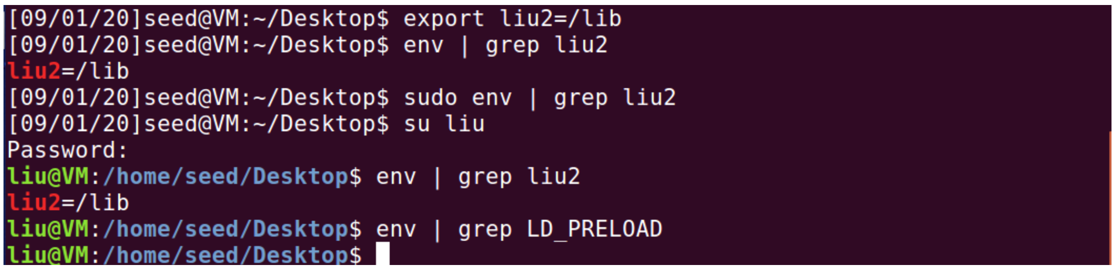
```
[09/01/20]seed@VM:~/Desktop$ sudo chown liu myprog
[09/01/20]seed@VM:~/Desktop$ ls -l myprog
-rwxr-xr-x 1 liu seed 7348 Sep  1 09:31 myprog
[09/01/20]seed@VM:~/Desktop$ su liu
Password:
liu@VM:/home/seed/Desktop$ printenv LD_PRELOAD
liu@VM:/home/seed/Desktop$ ./myprog
liu@VM:/home/seed/Desktop$ 
```

4> 结果分析

首先我认为 root 账户与普通环境变量不同，直接导致了 2 和 3 的结果，在 root 权限或是 set-uid 程序下的 root 权限都会复原环境变量。

在其他账户下，普通的环境变量会被继承，但是 LD_PRELOAD 并没有被继承，这就导致了出现 4 的结果。

验证结果的实验如下:

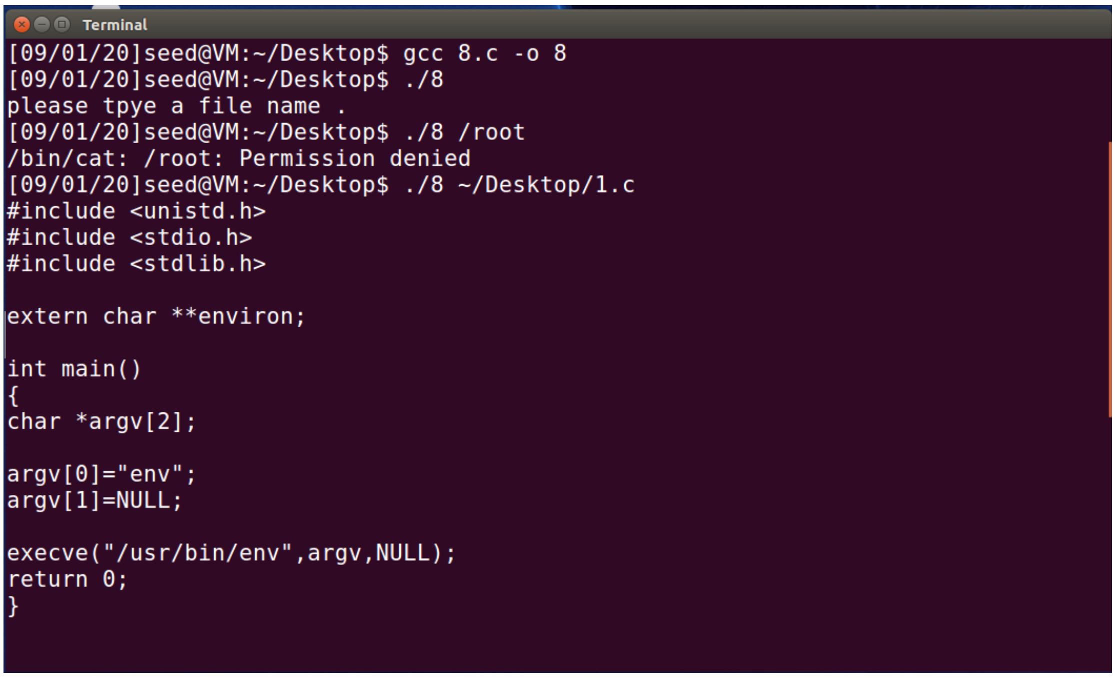


```
[09/01/20]seed@VM:~/Desktop$ export liu2=/lib
[09/01/20]seed@VM:~/Desktop$ env | grep liu2
liu2=/lib
[09/01/20]seed@VM:~/Desktop$ sudo env | grep liu2
[09/01/20]seed@VM:~/Desktop$ su liu
Password:
liu@VM:/home/seed/Desktop$ env | grep liu2
liu2=/lib
liu@VM:/home/seed/Desktop$ env | grep LD_PRELOAD
liu@VM:/home/seed/Desktop$ 
```

其中，先创建了 liu2 的环境变量，之后分别在 root 账户与 liu 账户下查找 liu2 环境变量，结果发现 root 中没有，而其他普通账户继承了 liu2.但是在其他账户下却并没有继承 LD_PRELOAD 这一环境变量。

实验 8： system()与 execve()调用外部程序

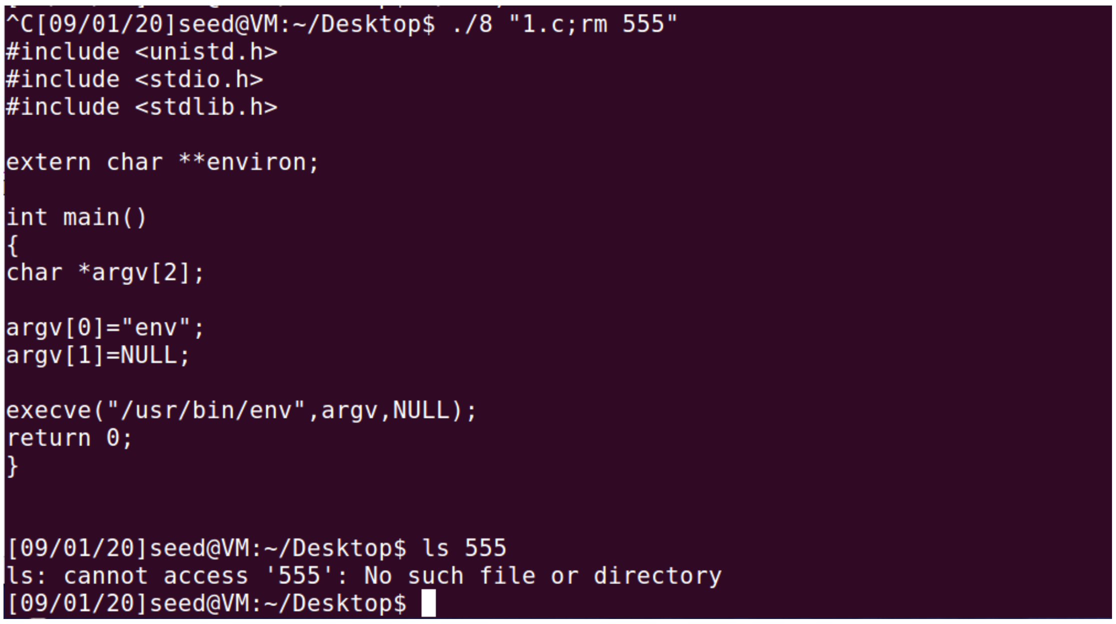
1> system()实验
在正常情况下的调用



```
[09/01/20]seed@VM:~/Desktop$ gcc 8.c -o 8
[09/01/20]seed@VM:~/Desktop$ ./8
please tpye a file name .
[09/01/20]seed@VM:~/Desktop$ ./8 /root
/bin/cat: /root: Permission denied
[09/01/20]seed@VM:~/Desktop$ ./8 ~/Desktop/1.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

argv[0]="env";
argv[1]=NULL;

execve("/usr/bin/env",argv,NULL);
return 0;
}
```

可以看到输入文件名可以实现正常的 cat 功能。

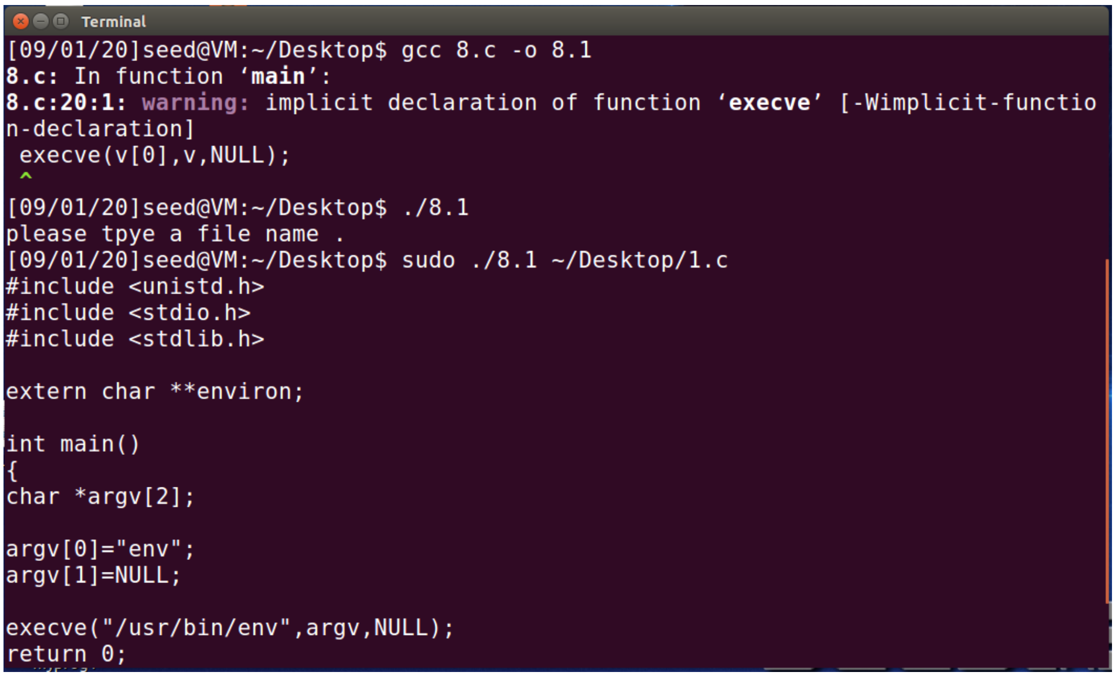
实现删除文件



```
^C[09/01/20]seed@VM:~/Desktop$ ./8 "1.c;rm 555"
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

argv[0]="env";
argv[1]=NULL;

execve("/usr/bin/env",argv,NULL);
return 0;
}


[09/01/20]seed@VM:~/Desktop$ ls 555
ls: cannot access '555': No such file or directory
[09/01/20]seed@VM:~/Desktop$ 
```

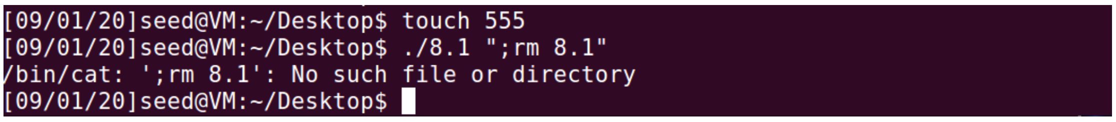
可以看到本来的 555 文件被删除，其中输入的参数为"1.c; 555"。

2> execve()实验
正常实现



```
[09/01/20]seed@VM:~/Desktop$ gcc 8.c -o 8.1
8.c: In function ‘main’:
8.c:20:1: warning: implicit declaration of function ‘execve’ [-Wimplicit-functio
n-declaration]
 execve(v[0],v,NULL);
 ^
[09/01/20]seed@VM:~/Desktop$ ./8.1
please tpye a file name .
[09/01/20]seed@VM:~/Desktop$ sudo ./8.1 ~/Desktop/1.c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>

extern char **environ;

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

argv[0]="env";
argv[1]=NULL;

execve("/usr/bin/env",argv,NULL);
return 0;
```

删除操作:



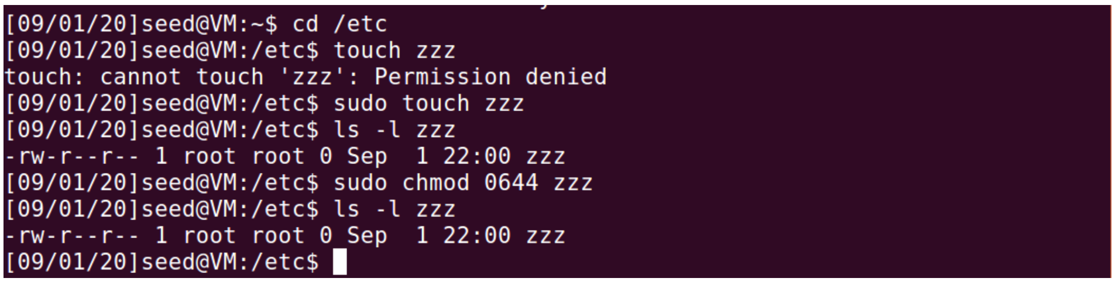
```
[09/01/20]seed@VM:~/Desktop$ touch 555
[09/01/20]seed@VM:~/Desktop$ ./8.1 ";rm 8.1"
/bin/cat: ';rm 8.1': No such file or directory
[09/01/20]seed@VM:~/Desktop$ 
```

3> 结论:

用 system()不是很安全, 会把数据与命令都当做数据处理。而 execve()会自动检测, 将数据与命令合并为非法数据, 从而不会造成漏洞问题。

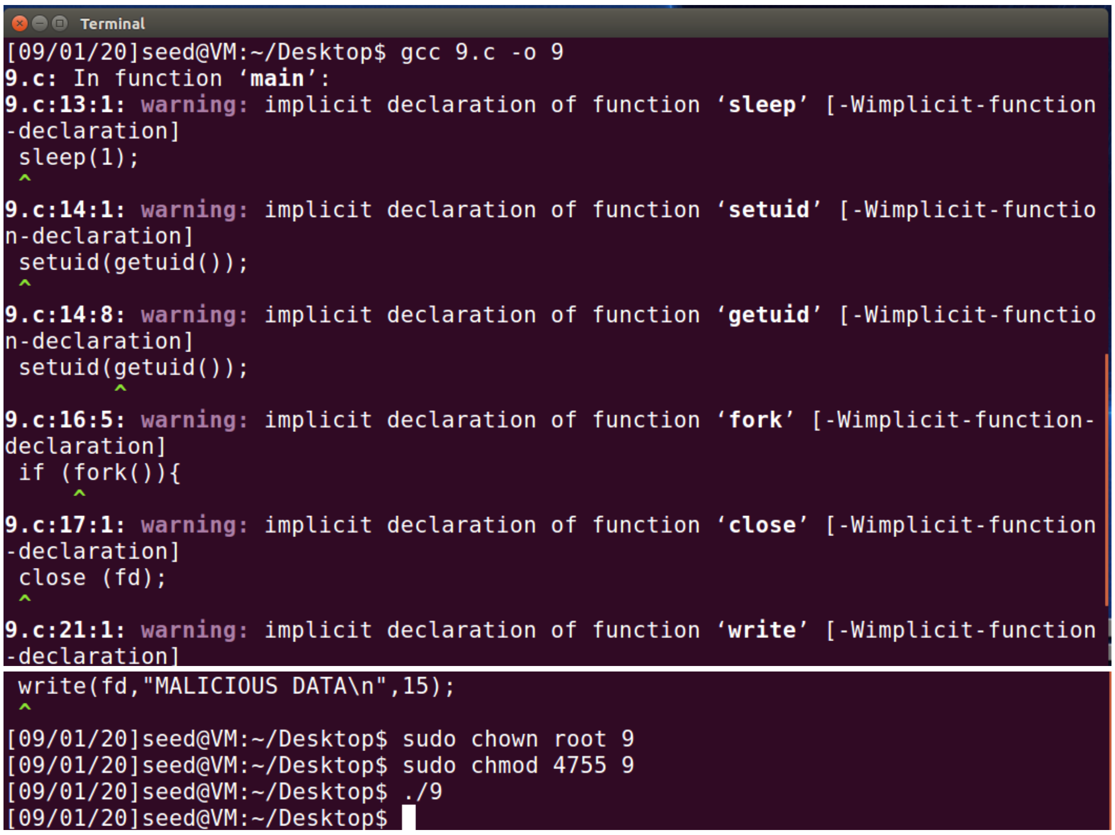
实验 9:　特权遗留

1> 创建 zzz 文件



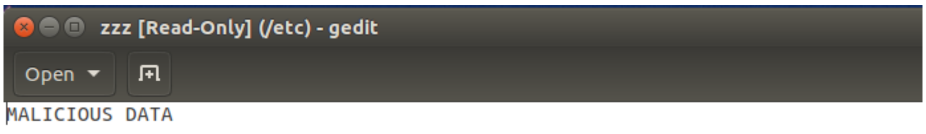
```
[09/01/20]seed@VM:~$ cd /etc
[09/01/20]seed@VM:/etc$ touch zzz
touch: cannot touch 'zzz': Permission denied
[09/01/20]seed@VM:/etc$ sudo touch zzz
[09/01/20]seed@VM:/etc$ ls -l zzz
-rw-r--r-- 1 root root 0 Sep  1 22:00 zzz
[09/01/20]seed@VM:/etc$ sudo chmod 0644 zzz
[09/01/20]seed@VM:/etc$ ls -l zzz
-rw-r--r-- 1 root root 0 Sep  1 22:00 zzz
[09/01/20]seed@VM:/etc$
```

2> 编译并运行程序



```
[09/01/20]seed@VM:~/Desktop$ gcc 9.c -o 9
9.c: In function ‘main’:
9.c:13:1: warning: implicit declaration of function ‘sleep’ [-Wimplicit-function
-declaration]
 sleep(1);
 ^
9.c:14:1: warning: implicit declaration of function ‘setuid’ [-Wimplicit-functio
n-declaration]
 setuid(getuid());
 ^
9.c:14:8: warning: implicit declaration of function ‘getuid’ [-Wimplicit-functio
n-declaration]
 setuid(getuid());
        ^
9.c:16:5: warning: implicit declaration of function ‘fork’ [-Wimplicit-function-
declaration]
 if (fork()){
     ^
9.c:17:1: warning: implicit declaration of function ‘close’ [-Wimplicit-function
-declaration]
 close (fd);
 ^
9.c:21:1: warning: implicit declaration of function ‘write’ [-Wimplicit-function
-declaration]
 write(fd,"MALICIOUS DATA\n",15);
 ^
[09/01/20]seed@VM:~/Desktop$ sudo chown root 9
[09/01/20]seed@VM:~/Desktop$ sudo chmod 4755 9
[09/01/20]seed@VM:~/Desktop$ ./9
[09/01/20]seed@VM:~/Desktop$
```

3> 查看 zzz 文件



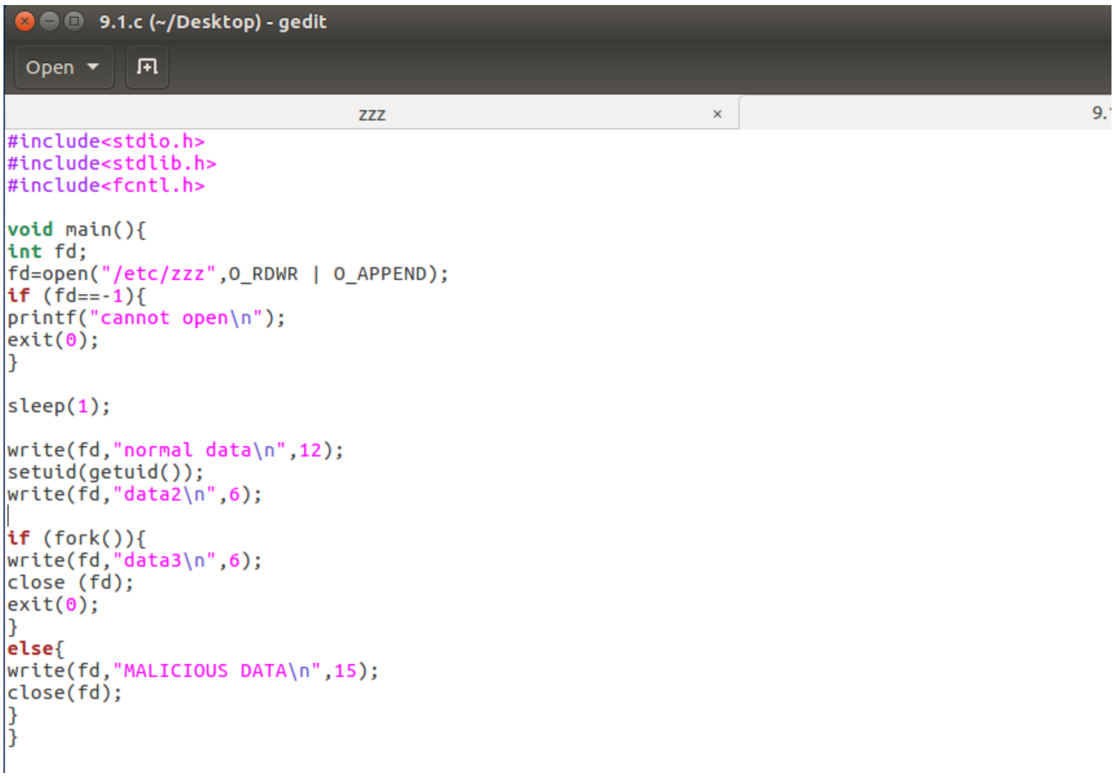
```
MALICIOUS DATA
```

4> 原因总结

原因主要是编写程序的疏漏，当收回权限时，并没有及时的关闭文件，导致文件

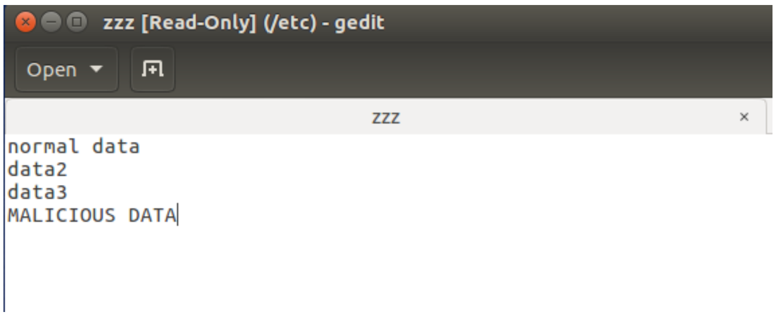
此时还是处于可写状态。实验证明如下：

程序为：



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

void main(){
int fd;
fd=open("/etc/zzz",O_RDWR | O_APPEND);
if (fd==-1){
printf("cannot open\n");
exit(0);
}

sleep(1);

write(fd,"normal data\n",12);
setuid(getuid());
write(fd,"data2\n",6);

if (fork()){
write(fd,"data3\n",6);
close (fd);
exit(0);
}
else{
write(fd,"MALICIOUS DATA\n",15);
close(fd);
}
}
```

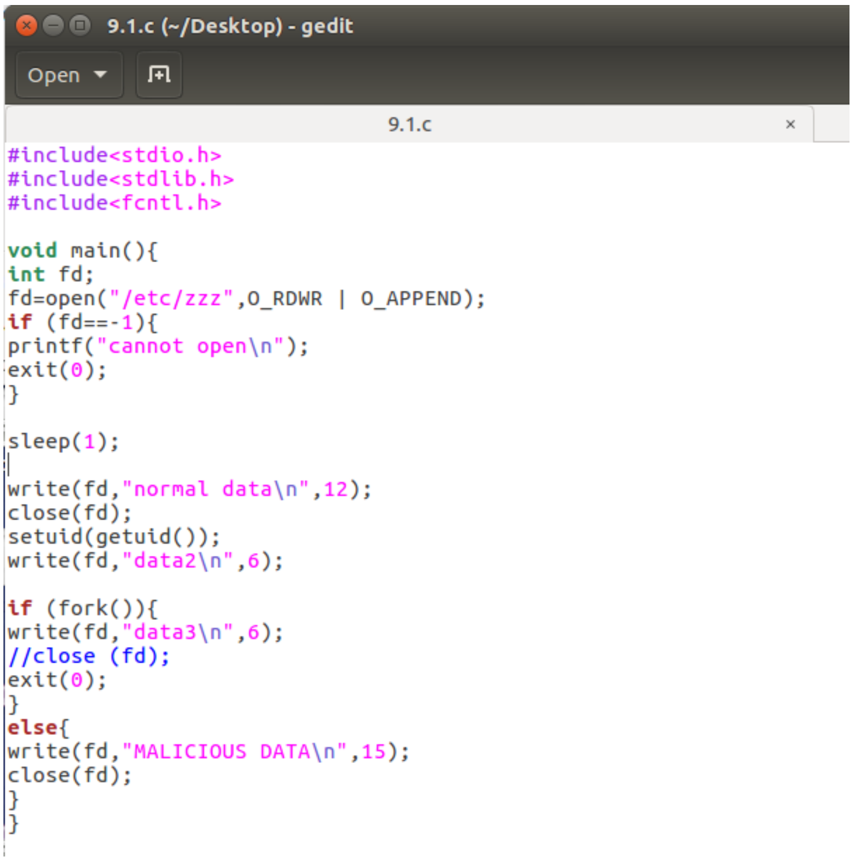
增加了在不同地方的输出，编译运行之后的 zzz 文件改写为：



```
normal data
data2
data3
MALICIOUS DATA
```

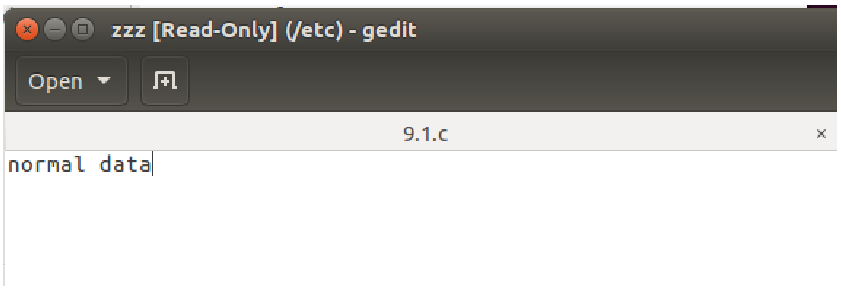
这证明了只要不关闭文件，即使是回收了权限，下面的操作还是可以对文件进行写操作。

同样为了证明及时关闭文件可以避免此问题，写了如下的程序：



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

void main(){
int fd;
fd=open("/etc/zzz",O_RDWR | O_APPEND);
if (fd==-1){
printf("cannot open\n");
exit(0);
}

sleep(1);

write(fd,"normal data\n",12);
close(fd);
setuid(getuid());
write(fd,"data2\n",6);

if (fork()){
write(fd,"data3\n",6);
//close (fd);
exit(0);
}
else{
write(fd,"MALICIOUS DATA\n",15);
close(fd);
}
}
```

此时及时关闭了 zzz 文件，编译之后的运行结果为：



```
normal data
```

发现成功避免了上述的问题，只有正常部分被正常输出。