



哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

2019 年秋季学期
计算机学院大三
计算机系统安全课程

Lab 1 实验报告

姓名	冯帅
学号	1170301027
班号	1703201
电子邮件	1170301027@stu.hit.edu.cn
手机号码	15765513201

实验一

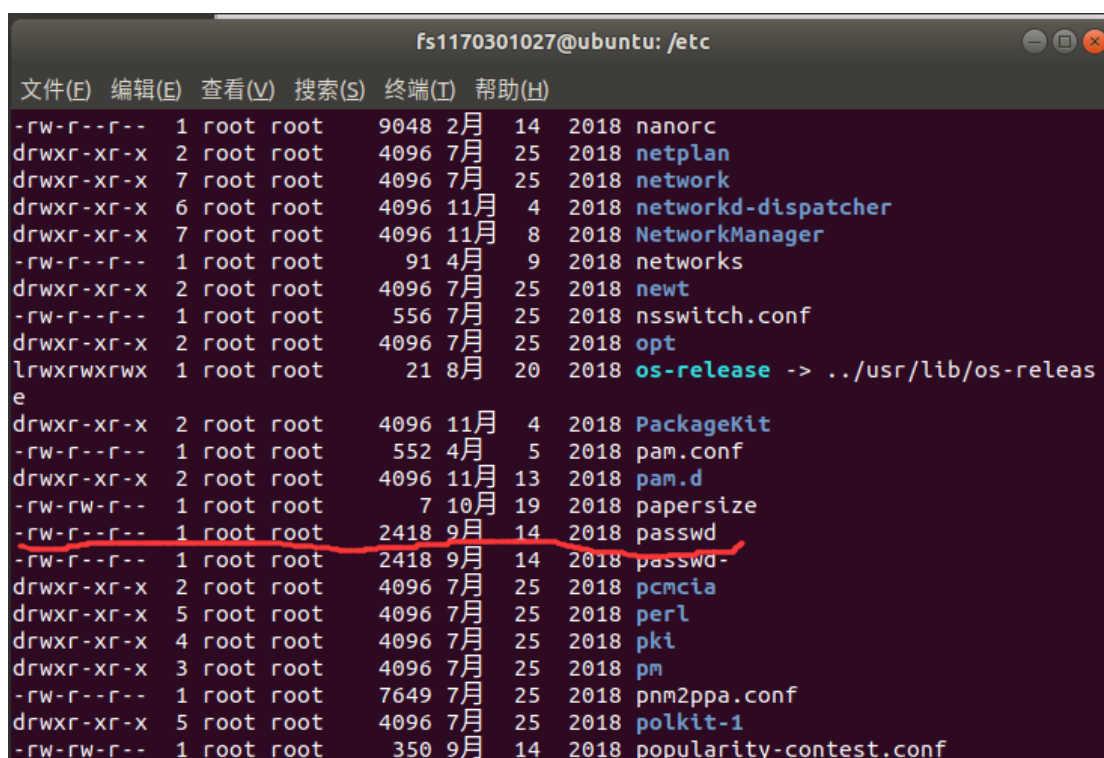
一、Linux 系统文件和目录权限设置与辨识 setuid 程序 uid 差异

(独立完成)

1、设计并实现不同用户对不同类文件的 r、w、x 权限:

(1) 查看系统文件的权限设置

a)查看/etc/passwd 文件和/etc/bin/passwd 文件的权限设置，并分析其权限为什么这么设置;



The screenshot shows a terminal window titled 'fs1170301027@ubuntu: /etc'. It displays the output of the 'ls -l' command for files in the /etc directory. The files and their permissions are as follows:

File	Permissions	Owner	Group	Size	Month	Day	Year	File Name
nanorc	-rw-r--r--	1 root	root	9048	2月	14	2018	nanorc
netplan	drwxr-xr-x	2 root	root	4096	7月	25	2018	netplan
network	drwxr-xr-x	7 root	root	4096	7月	25	2018	network
networkd-dispatcher	drwxr-xr-x	6 root	root	4096	11月	4	2018	networkd-dispatcher
NetworkManager	drwxr-xr-x	7 root	root	4096	11月	8	2018	NetworkManager
networks	-rw-r--r--	1 root	root	91	4月	9	2018	networks
newt	drwxr-xr-x	2 root	root	4096	7月	25	2018	newt
nsswitch.conf	-rw-r--r--	1 root	root	556	7月	25	2018	nsswitch.conf
opt	drwxr-xr-x	2 root	root	4096	7月	25	2018	opt
os-release	lrwxrwxrwx	1 root	root	21	8月	20	2018	os-release -> ../usr/lib/os-releas
PackageKit	drwxr-xr-x	2 root	root	4096	11月	4	2018	PackageKit
pam.conf	-rw-r--r--	1 root	root	552	4月	5	2018	pam.conf
pam.d	drwxr-xr-x	2 root	root	4096	11月	13	2018	pam.d
papersize	-rw-rw-r--	1 root	root	7	10月	19	2018	papersize
passwd	-rw-r--r--	1 root	root	2418	9月	14	2018	passwd
passwd-	-rw-r--r--	1 root	root	2418	9月	14	2018	passwd-
pcmcia	drwxr-xr-x	2 root	root	4096	7月	25	2018	pcmcia
perl	drwxr-xr-x	5 root	root	4096	7月	25	2018	perl
pki	drwxr-xr-x	4 root	root	4096	7月	25	2018	pki
pn	drwxr-xr-x	3 root	root	4096	7月	25	2018	pn
pnm2ppa.conf	-rw-r--r--	1 root	root	7649	7月	25	2018	pnm2ppa.conf
polkit-1	drwxr-xr-x	5 root	root	4096	7月	25	2018	polkit-1
popularity-contest.conf	-rw-rw-r--	1 root	root	350	9月	14	2018	popularity-contest.conf

- -rw-r--r--:十个字符确定不同用户能对文件干什么，第一个字符“-”代表文件，之后每三个一组，限定文件所有者，所有者同一组和不与所有者同一组的用户对该文件的操作，/etc/passwd 对于文件所有者的权限是可读可写，对其他人只有读操作。
- Unix 的口令文件 /etc/passwd 记录着所有用户和对应的登录密码等信息。
/etc/passwd 文件的所有者是 root 而且只有 root 用户有权限对该文件进行写操作。但是 Unix 系统其实是允许所有的用户修改自己的登录口令的

```

fs1170301027@ubuntu: /bin
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
-rwxr-xr-x 1 root root 88672 12月 1 2017 ntfssecaudit
-rwxr-xr-x 1 root root 38944 12月 1 2017 ntfstruncate
-rwxr-xr-x 1 root root 30744 12月 1 2017 ntfsusermap
-rwxr-xr-x 1 root root 47752 12月 1 2017 ntfswipe
lrwxrwxrwx 1 root root 6 9月 15 2018 open -> openvt
-rwxr-xr-x 1 root root 18872 1月 22 2018 openvt
lrwxrwxrwx 1 root root 14 9月 15 2018 pidof -> /sbin/killall5
-rwsr-xr-x 1 root root 64424 3月 10 2017 ping
lrwxrwxrwx 1 root root 4 9月 15 2018 ping4 -> ping
lrwxrwxrwx 1 root root 4 9月 15 2018 ping6 -> ping
-rwxr-xr-x 1 root root 38904 9月 12 2018 plymouth
-rwxr-xr-x 1 root root 133432 5月 14 2018 ps
-rwxr-xr-x 1 root root 35000 1月 18 2018 pwd
lrwxrwxrwx 1 root root 4 9月 15 2018 rbash -> bash
-rwxr-xr-x 1 root root 43192 1月 18 2018 readlink
-rwxr-xr-x 1 root root 89 4月 27 2016 red
-rwxr-xr-x 1 root root 63704 1月 18 2018 rm
-rwxr-xr-x 1 root root 43192 1月 18 2018 rmdir
lrwxrwxrwx 1 root root 4 9月 15 2018 rnano -> nano
-rwxr-xr-x 1 root root 18760 12月 31 2017 run-parts
-rwxr-xr-x 1 root root 109000 1月 30 2018 sed
-rwxr-xr-x 1 root root 35512 4月 22 2017 setfacl
-rwxr-xr-x 1 root root 43144 1月 22 2018 setfont
-rwxr-xr-x 1 root root 39288 8月 24 2018 setupcon

```

- -rwxr-xr-x:/bin/pwd 对文件所有者有读写执行的权限，对其他人只有读和执行的权限，

b)找到 2 个设置了 setuid 位的可执行程序，该程序的功能，该程序如果不设置 setuid 位是否能够达到相应的功能，

```

fs1170301027@ubuntu:/usr/bin$ ls -l sudo
-rwsr-xr-x 1 root root 149080 1月 18 2018 sudo

```

1.

sudo 是用户临时获取 root 权限的一个命令，很多程序都需要 root 权限才能执行，如果不设置，普通用户将很难执行那些程序等。

```

fs1170301027@ubuntu:/usr/bin$ ls -l gpasswd
-rwsr-xr-x 1 root root 75824 1月 25 2018 gpasswd

```

2.

passwd 为了用户可以修改自己的密码，etc/shadow 只用 root 才能修改，如果不设置 setuid 位，则用户无法修改自己的密码，gpasswd 同

chmod 参数说明：

u: 表示文件拥有者，即 user

g: 组拥有者，即 group

o: 其它用户拥有者,即 other

a: 所有用户，即相当于 ugo

: 省略不写代表 a，即所有用户

(2) 设置文件或目录权限

a)用户 A 具有文本文件“流星雨.txt”，该用户允许别人下载；

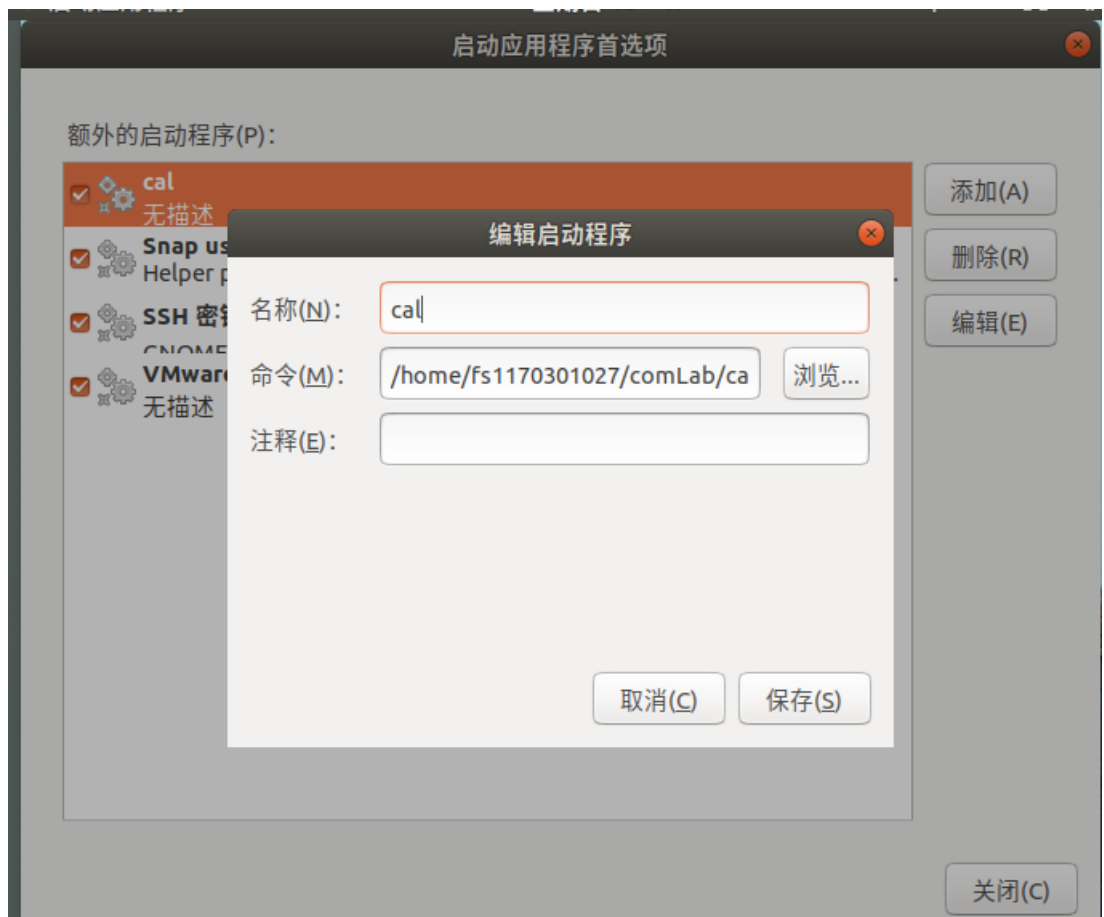
```
fs1170301027@ubuntu:~$ mkdir comLab
fs1170301027@ubuntu:~$ cd comLab
fs1170301027@ubuntu:~/comLab$ touch liuxingyu.txt
fs1170301027@ubuntu:~/comLab$ chmod 744 liuxingyu.txt
fs1170301027@ubuntu:~/comLab$ ls -l liuxingyu.txt
-rwxr--r-- 1 fs1170301027 fs1170301027 0 12月  1 18:05 liuxingyu.txt
fs1170301027@ubuntu:~/comLab$
```

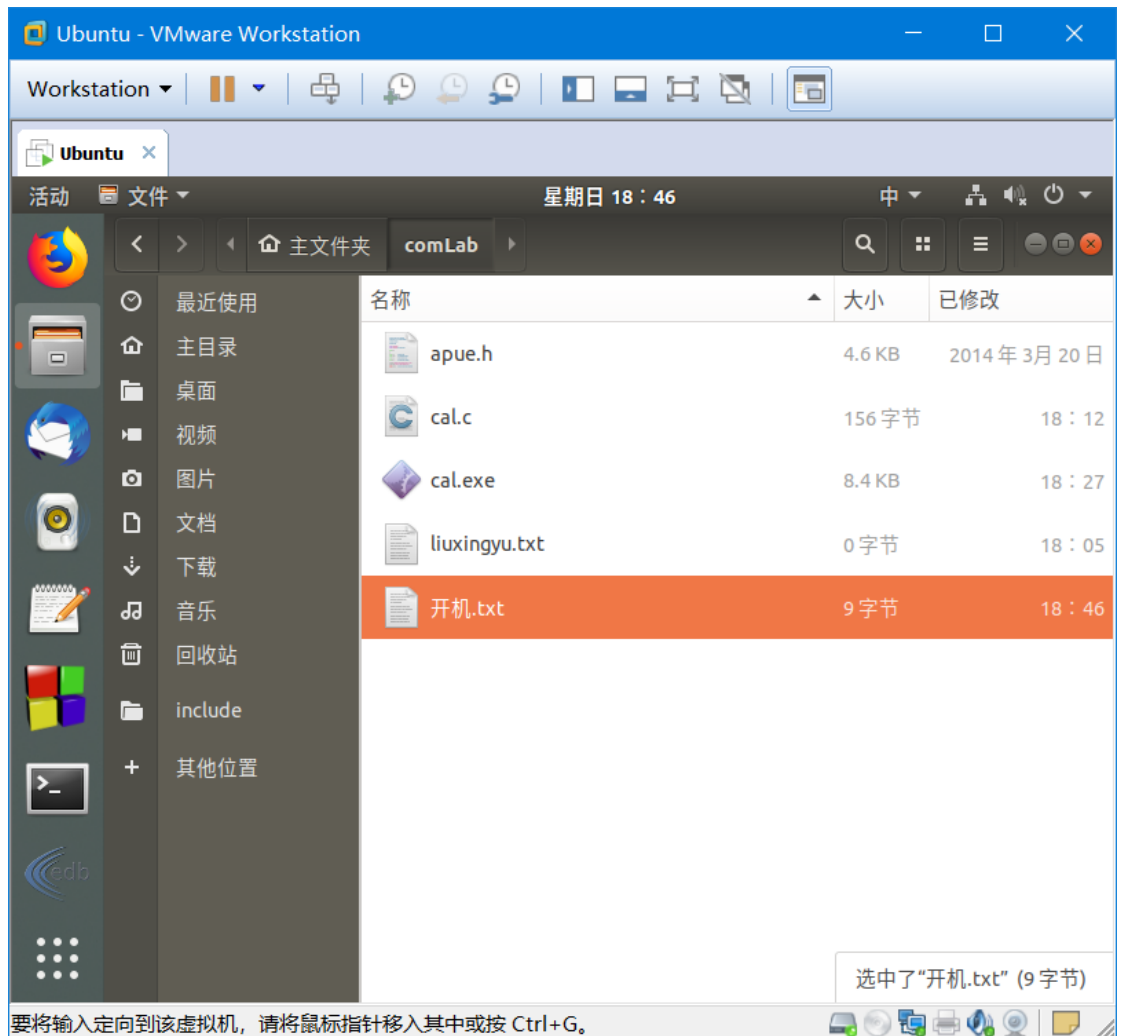
允许其他人有读权限

b)用户 A 编译了一个可执行文件“cal.exe”，该用户想在系统启动时运行；

```
fs1170301027@ubuntu:~/comLab$ ls -l cal.exe
-rwxr-xr-x 1 fs1170301027 fs1170301027 8384 12月  1 18:27 cal.exe
fs1170301027@ubuntu:~/comLab$ sudo cp cal.exe /etc/init.d
[sudo] fs1170301027 的密码:
```

修改权限后，添加到开机启动项





c)用户 A 有起草了文件“demo.txt”，想让同组的用户帮其修改文件；

```
fs1170301027@ubuntu:~/comLab$ touch demo.txt
fs1170301027@ubuntu:~/comLab$ chmod 664 demo.txt
fs1170301027@ubuntu:~/comLab$ ls -l demo.txt
-rw-rw-r-- 1 fs1170301027 fs1170301027 0 12月 1 18:57 demo.txt
fs1170301027@ubuntu:~/comLab$
```

同组用户有写权限

d)一个 root 用户拥有的网络服务程序“netmonitor.exe”，需要设置 setuid 位才能完成其功能。

```
fs1170301027@ubuntu:~/comLab$ chmod u+s netmonitor.exe
fs1170301027@ubuntu:~/comLab$ ls -l netmonitor.exe
-rwsr--r-- 1 fs1170301027 fs1170301027 0 12月 1 18:59 netmonitor.exe
fs1170301027@ubuntu:~/comLab$
```

设置 setuid 位

2、一些可执行程序运行时需要系统管理员权限，在 UNIX 中可以利用 setuid 位实现其功能，但 setuid 了的程序运行过程中拥有了 root 权限，因此在完成管

理操作后需要切换到普通用户的身份执行后续操作。

(1)设想一种场景，比如提供 http 网络服务，需要设置 setuid 位，并为该场景编制相应的代码；

```
void show_id()
{
    printf("real uid: %d\n",getuid());
    printf("effective uid: %d\n",geteuid());
}

int main(int argc, char *argv[])
{
    int uid;
    printf("Set以前: \n");
    show_id();
    if(setuid(getuid())<0)
    {
        perror("setuid error!");
    }
    printf("Set以后: \n");
    show_id();
    system("echo chmod u+s http.exe");
    system("chmod u+s http.exe");
    system("ls -l http.exe");

    return(0);
}
```

```
fs1170301027@ubuntu:~/comLab/setuid$ g++ http.c -o http.exe
fs1170301027@ubuntu:~/comLab/setuid$ ./http.exe
Set以前:
real uid: 1000
effective uid: 1000
Set以后:
real uid: 1000
effective uid: 1000
chmod u+s http.exe
-rwsr-xr-x 1 fs1170301027 fs1170301027 8600 12月  2 21:55 http.exe
fs1170301027@ubuntu:~/comLab/setuid$
```

Setuid ()

(2)如果用户 fork 进程后，父进程和子进程中 euid、ruid、suid 的差别；

```

void show_id(uid_t *ruid,uid_t *euid,uid_t *suid)
{
    getresuid(ruid,euid,suid);
    printf("ruid: %d, euid:%d, suid:%d\n", *ruid,*euid,*suid);
}

int main(int argc, char *argv[])
{
    uid_t ruid,euid,suid;
    if(fork()==0)
    {
        printf("\n");
        printf("Child process:\n");
        show_id(&ruid,&euid,&suid);
    }
    else
    {
        printf("\n");
        printf("Father process:\n");
        show_id(&ruid,&euid,&suid);
    }
    return(0);
}

```

```

fs1170301027@ubuntu:~/comLab/setuid$ sudo g++ fork.c -o fork
fs1170301027@ubuntu:~/comLab/setuid$ ./fork

```

```

Father process:
ruid: 1000, euid:1000, suid:1000

Child process:
ruid: 1000, euid:1000, suid:1000

```

Fork()函数两次返回观察父进程与子进程的 ruid, euid, suid 的变化, 发现无变化

(3)利用 execl 执行 setuid 程序后, euid、ruid、suid 是否有变化;

```

void show_id(uid_t *ruid,uid_t *euid,uid_t *suid)
{
    getresuid(ruid,euid,suid);
    printf("ruid: %d, euid:%d, suid:%d\n", *ruid,*euid,*suid);
}

int main(int argc, char *argv[])
{
    uid_t ruid,euid,suid,pid;
    if((pid = fork())==0)
    {
        execl("/home/fs1170301027/comLab/setuid/
newset.exe","newset.exe",NULL);

    }
    else if (pid>0)
    {
        printf("\n");
        printf("Father process:\n");
        show_id(&ruid,&euid,&suid);
    }
    return(0);
}

```

```
fs1170301027@ubuntu:~/comLab/setuid$ ./execl.exe
```

```

Father process:
ruid: 1000, euid:1000, suid:1000
A process after setuid:
ruid: 1000, euid:1000, suid:1000

```

```
fs1170301027@ubuntu:~/comLab/setuid$ sudo ./execl.exe
```

```

Father process:
ruid: 0, euid:0, suid:0
A process after setuid:
ruid: 0, euid:1000, suid:1000

```

(4)程序何时需要临时性放弃 root 权限，何时需要永久性放弃 root 权限，并在程序中分别实现两种放弃权限方法；


```

//程序临时性放弃root权限
void abandonRootTemporary(uid_t uid_tran)
{
    uid_t ruid, euid,suid;
    getresuid(&ruid, &euid, &suid);
    if (euid == 0)
    {
        // 临时性放弃root权限
        int is_seteuid = seteuid(uid_tran);
        getresuid(&ruid, &euid, &suid);
        if(euid > 0)
        {
            printf("临时性放弃root权限成功\n");
        }
        else
        {
            printf("临时性放弃root权限失败\n");
        }
        printf("ruid = %d, euid = %d, suid = %d\n", ruid, euid,
suid);
    }
    else
    {
        printf("无 root 权限, 无法放弃root权限\n");
    }
}

```

```

void abandonRootPermanent(uid_t uid_tran)
{
    uid_t ruid, euid,suid;
    getresuid(&ruid, &euid, &suid);
    if (euid != 0 && (ruid == 0 || suid == 0))
    {
        setuid(0);
        getresuid(&ruid, &euid, &suid);
    }
    if (euid == 0)
    {
        // 永久性放弃root权限
        setresuid(uid_tran, uid_tran, uid_tran);
        getresuid(&ruid, &euid, &suid);
        if(ruid > 0 && euid > 0 && suid > 0)
        {
            printf("永久性放弃root权限成功\n");
        }
        else
        {
            printf("永久性放弃root权限失败\n");
        }
        printf("ruid = %d, euid = %d, suid = %d\n", ruid, euid,
suid);
    }
    else
    {
        printf("无 root 权限, 无法放弃root权限\n");
    }
}

```

```

int main()
{
    abandonRootTemporary(1001); // 临时性放弃root权限
    abandonRootPermanent(1001); // 永久性放弃root权限
    return(0);
}

```

使用 `setresuid(0,1001,0)`临时放弃权限，使用 `setresuid(1001,1001,1001)`永久放弃权限。

```

fs1170301027@ubuntu:~/comLab/setuid$ g++ abandon.c -o abandon.exe
fs1170301027@ubuntu:~/comLab/setuid$ ./abandon.exe
无 root 权限，无法放弃root权限
无 root 权限，无法放弃root权限
fs1170301027@ubuntu:~/comLab/setuid$ sudo ./abandon.exe
临时性放弃root权限成功
ruid = 0, euid = 1001, suid = 0
永久性放弃root权限成功
ruid = 1001, euid = 1001, suid = 1001

```

(5)`execl` 函数族中有多个函数，比较有环境变量和无环境变量的函数使用的差异。

```

void show_id(uid_t *ruid,uid_t *euid,uid_t *suid)
{
    getresuid(ruid,euid,suid);
    printf("ruid: %d, euid:%d, suid:%d\n", *ruid,*euid,*suid);
}

int main()
{
    uid_t ruid,euid,suid,pid;

    if ((pid = fork()) == 0)
    {
        printf("有环境变量的函数使用\n");
        execlp("ls", "ls", "-l",NULL);
        show_id(&ruid,&euid,&suid);
    }
    else if (pid > 0)
    {
        printf("无环境变量的函数使用\n");
        execl("/bin/ls", "ls", "-l",NULL);
        show_id(&ruid,&euid,&suid);
    }

    return 0;
}

```

```
fs1170301027@ubuntu:~/comLab/setuid$ ./env.exe
无环境变量的函数使用
有环境变量的函数使用
总用量 124
总用量 124
-rwxr-xr-x 1 root root 8672 12月 2 22:49 abandon
-rw-r--r-- 1 fs1170301027 fs1170301027 1684 12月 2 22:49 abandon.c
-rwxr-xr-x 1 fs1170301027 fs1170301027 8680 12月 2 22:50 abandon.exe
-rwxr-xr-x 1 root root 8600 12月 3 01:56 env
-rw-r--r-- 1 fs1170301027 fs1170301027 604 12月 3 02:02 env.c
-rwxr-xr-x 1 fs1170301027 fs1170301027 8608 12月 3 02:02 env.exe
-rw-r--r-- 1 fs1170301027 fs1170301027 507 12月 2 22:28 execl.c
-rwxr-xr-x 1 fs1170301027 fs1170301027 8616 12月 2 22:29 execl.exe
-rwxr-xr-x 1 root root 8568 12月 2 22:04 fork
-rw-r--r-- 1 fs1170301027 fs1170301027 485 12月 2 22:04 fork.c
-rw-r--r-- 1 fs1170301027 fs1170301027 470 12月 2 21:54 http.c
-rwsr-xr-x 1 fs1170301027 fs1170301027 8600 12月 2 21:55 http.exe
-rw-r--r-- 1 fs1170301027 fs1170301027 367 12月 2 22:25 newset.c
-rwxr-xr-x 1 root root 8672 12月 2 22:49 abandon
-rwsr-xr-x 1 fs1170301027 fs1170301027 8488 12月 2 22:25 newset.exe
-rw-r--r-- 1 fs1170301027 fs1170301027 1102 12月 2 21:44 test.c
```

exec 家族一共有六个函数，分别是：

- (1) int execl(constchar*path,constchar*arg,.....);
- (2) int execlp(constchar*path,constchar*arg,.....,char*constenvp[]);
- (3) int execv(constchar*path,char*constargv[]);
- (4) int execvp(constchar*filename,char*constargv[],char*constenvp[]);
- (5) int execvp(constchar*file,char*constargv[]);
- (6) int execlp(constchar*file,constchar*arg,.....);

其中以 p 结尾的函数，可以向函数传递一个指向环境字符串指针数组的指针。即自个定义各个环境变量，而其它四个则使用进程中的环境变量。

例如，execlp,execvp，表示第一个参数 path 不用输入完整路径，只有给出命令名即可，它会在环境变量 PATH 当中查找命令。

程序设计：

- (1) 使用 execl 函数，不指定路径
 - (2) 使用 execlp 函数，不指定路径
 - (3) 利用 fork 的两次返回不同执行两个函数，观察结果
- 3、编制实验报告，对问题一说明原理，对问题 2 说明设计过程和实验步骤。并写出心得体会。

心得：：本次实验自己动手操作了有关于 linux 系统中文件和目录的权限管理部分的内容，更加深刻地理解了 ruid, euid, suid 之间的区别与联系；清楚了通过 fork 创建的子进程与父进程之间的关系。进一步学习了 execl 函数族的相关知识。

二、chroot 的配置和 root 的 capability 使用

1、利用 chroot 工具来虚拟化管理

```
fs1170301027@ubuntu:~/comLab$ ls -l /usr/sbin/chroot
-rwsr-xr-x 1 root root 39096 1月 18 2018 /usr/sbin/chroot
```

1) 实现 bash 或 ps 的配置使用:

```
root@ubuntu:~# chroot /var/chroot
bash-4.4# pure-ftpd -j -lpuredb:/etc/pure-ftpd/pureftpd.pdb
Unable to start a standalone server: Address already in use
bash-4.4# pure-ftpd -j -lpuredb:/etc/pure-ftpd/pureftpd.pdb
^Cbash-4.4# pure-ftpd -j -lpuredb:/etc/pure-ftpd/pureftpd.pdb
^Cbash-4.4# pure-ftpd -j -lpuredb:/etc/pure-ftpd/pureftpd.pdb
```

2) 利用 chroot 实现 SSH 服务或 FTP 服务的虚拟化隔离:

```
ftp> user
(username) hello
331 User hello OK. Password required
Password:
230 OK. Current directory is /
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> mkdir fengshuaishishab
257 "fengshuaishishab" : The directory was successfully created
ftp> ls
200 PORT command successful
150 Connecting to port 46313
drwxr-xr-x  2 1001      1001              4096 Dec  2 16:25 fengshuais
226-Options: -l
226 1 matches total
ftp> ls -h
```

2) chroot 后如何降低权限, 利用实验一中编制的程序检查权限的合理性:

```
void show_id(uid_t *ruid,uid_t *euid,uid_t *suid)
{
    getresuid(ruid,euid,suid);
    printf("ruid: %d, euid:%d, suid:%d\n", *ruid,*euid,*suid);
}

int main()
{
    uid_t ruid,euid,suid,pid;
    pid = getuid();
    printf("执行:\n");
    show_id(&ruid,&euid,&suid);
    chdir("/var/chroot");
    //chroot("/var/chroot");
    if(!chroot("/var/chroot"))
    {
        printf("chroot success!\n");
        show_id(&ruid,&euid,&suid);
        printf("降低权限:\n");
        setresuid(pid,pid,pid);
        show_id(&ruid,&euid,&suid);
        execlp("ls","ls","-l",NULL);
    }
}
```

```

fs1170301027@ubuntu:~/comLab$ ./afterchroot.exe
执行:
ruid: 1000, euid:0, suid:0
chroot success!
ruid: 1000, euid:0, suid:0
降低权限:
ruid: 1000, euid:1000, suid:1000
total 64
drwxr-xr-x 2 0 0 4096 Dec 2 15:20 bin
-rwxr-xr-x 1 0 0 4856 Dec 2 16:20 chroot_create.sh
-rwxr-xr-x 1 0 0 4970 Dec 2 15:18 chroot_create.sh~
drwxr-xr-x 3 0 0 4096 Dec 2 15:21 dev
drwxr-xr-x 3 0 0 4096 Dec 2 16:21 etc
drwxr-xr-x 3 0 0 4096 Dec 2 16:23 home
drwxr-xr-x 3 0 0 4096 Dec 2 15:20 lib
drwxr-xr-x 2 0 0 4096 Dec 2 16:21 lib64
drwxr-xr-x 2 0 0 4096 Dec 2 15:20 proc
drwxr-xr-x 2 0 0 4096 Dec 2 15:38 root

```

3) 在 chroot 之前没有采用 cd xx 目录, 会对系统有何影响, 编制程序分析其影响。
注释掉 chdir 然后前后执行情况对比。

```

fs1170301027@ubuntu:~/comLab$ sudo g++ afterchroot.c -o afterchroot.exe
fs1170301027@ubuntu:~/comLab$ sudo chmod 4571 afterchroot.exe
fs1170301027@ubuntu:~/comLab$ ./afterchroot.exe
执行:
ruid: 1000, euid:0, suid:0
chroot success!
ruid: 1000, euid:0, suid:0
降低权限:
ruid: 1000, euid:1000, suid:1000
total 68
-rwxr-xr-x 1 1000 1000 8760 Dec 6 04:35 afterchroot
-rw-r--r-- 1 1000 1000 624 Dec 6 06:00 afterchroot.c
-r-srwx--x 1 0 0 8672 Dec 6 06:00 afterchroot.exe
-rwxrwxrwx 1 1000 1000 4631 Mar 20 2014 apue.h
-rw-r--r-- 1 1000 1000 156 Dec 1 10:12 cal.c
-rwxr-xr-x 1 1000 1000 8384 Dec 1 10:27 cal.exe
-rw-rw-r-- 1 1000 1000 0 Dec 1 10:57 demo.txt
-rwxr--r-- 1 1000 1000 0 Dec 1 10:05 liuxingyu.txt
-rwSr--r-- 1 1000 1000 0 Dec 1 10:59 netmonitor.exe
-rw-r--r-- 1 1000 1000 4505 Dec 1 11:10 question2.cpp~
drwxr-xr-x 2 1000 1000 4096 Dec 2 18:02 setuid
-rw-rw-r-- 1 1000 1000 9 Dec 1 10:55 '$'\345\274\200\346\234\272'.txt'

```

前后目录不一样

2、编制实验报告, 对问题一说明原理, 对问题 2 说明设计过程和实验步骤。并写出心得体会。

心得: 搞了两天的 ftpd, 无限重装, 各种登录失败, 然后转战 pure-ftpd, 就成了。实验中初步尝试了脚本文件的撰写, 体会到了脚本文件的好处。

实验结果:

