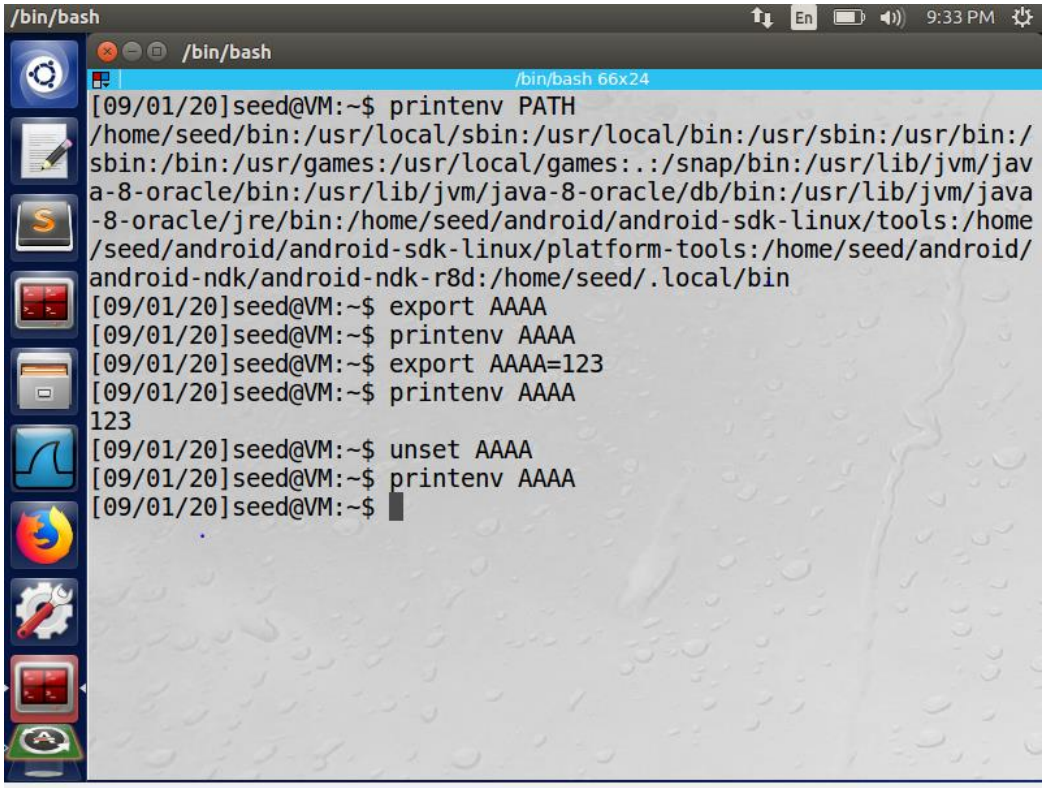
Lab 1

实验目的：实验为了了解环境变量是如何影响程序的和系统的行为。环境变量是一组动态命名值，可以影响这种方式正在运行的进程将在计算机上运行。大多数操作系统都使用它们。在这个实验中,我们将了解环境变量是如何工作的，以及它们如何从父进程传播到父进程儿童，以及他们如何影响系统/程序行为。我们特别感兴趣的是环境如何变量影响 Set-UID 程序的行为，而 Set-UID 程序通常是特权程序。

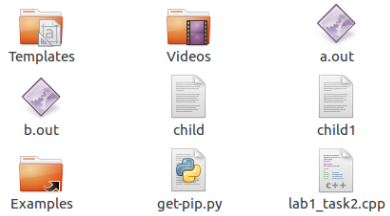
Task 1: Manipulating Environment Variables



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

```
[09/02/20]seed@VM:~$ g++ lab1_task2.cpp -o a.out
[09/02/20]seed@VM:~$ a.out > child
[09/02/20]seed@VM:~$ g++ lab1_task2.cpp -o b.out
[09/02/20]seed@VM:~$ b.out > child1
[09/02/20]seed@VM:~$ diff -u child child1
--- child       2020-09-02 03:07:54.651867445 -0400
+++ child1      2020-09-02 03:09:30.595815446 -0400
@@ -72,4 +72,4 @@
 LESSCLOSE=/usr/bin/lesspipe %s %s
 XAUTHORITY=/home/seed/.Xauthority
 COLORTERM=gnome-terminal
- _=./a.out
+ _=./b.out
```

我们首先根据示例的代码第一步执行后，将可执行文件 a.out 中的字符放入 child 文件中；第二步将子进程的 printenv()注释掉后，将可执行文件 b.out 中的字符放入 child1 文件中；用 diff 命令找出两个文件的不同之处。



结论：我们可以从该任务得出结论，子进程使用 fork()函数，会完全继承父进程的环境变量。

Task 3: Environment Variables and execve()

第一步：源码是 execve()的函数中传递了 NULL，子进程传递什么都没有，所以在输出 c.out 是无任何输出

```
[09/02/20]seed@VM:~$ gcc lab1_task3.c -o c.out
lab1_task3.c: In function ‘main’:
lab1_task3.c:13:2: warning: implicit declaration of function ‘exec
ve’ [-Wimplicit-function-declaration]
  execve("/usr/bin/env", argv, NULL);
  ^
[09/02/20]seed@VM:~$ c.out
[09/02/20]seed@VM:~$
```

第二步：源码中 execve()的函数中传递了自己的 environ，所以在 d.out 的输出中，如下面截图所示。

```
[09/02/20]seed@VM:~$ gcc lab1_task3.c -o d.out
lab1_task3.c: In function 'main':
lab1_task3.c:13:2: warning: implicit declaration of function 'exec
ve' [-Wimplicit-function-declaration]
  execve("/usr/bin/env", argv, environ);
  ^
[09/02/20]seed@VM:~$ d.out
XDG_VTNR=7
ORBIT_SOCKETDIR=/tmp/orbit-seed
XDG_SESSION_ID=c1
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
IBUS_DISABLE_SNOOPER=1
TERMINATOR_UUID=urn:uuid:973533ef-60d1-4435-80b3-5ccd48216ee8
CLUTTER_IM_MODULE=xim
SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE_PID=26596
ANDROID_HOME=/home/seed/android/android-sdk-linux
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
TERM=xterm
SHELL=/bin/bash
DERBY_HOME=/usr/lib/jvm/java-8-oracle/db
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
LD_PRELOAD=/home/seed/lib/boost/libboost_program_options.so.1.64.0
:/home/seed/lib/boost/libboost_filesystem.so.1.64.0:/home/seed/lib
```

结论：execve()函数在对传递不同的环境变量时的情况不同。

Task 4: Environment Variables and system()

```
[09/02/20]seed@VM:~$ gcc lab1_task4.c -o e.out
[09/02/20]seed@VM:~$ e.out
LESSOPEN=| /usr/bin/lesspipe %s
GNOME_KEYRING_PID=
USER=seed
LANGUAGE=en_US
UPSTART_INSTANCE=
J2SDKDIR=/usr/lib/jvm/java-8-oracle
XDG_SEAT=seat0
SESSION=ubuntu
XDG_SESSION_TYPE=x11
COMPIZ_CONFIG_PROFILE=ubuntu-lowgfx
ORBIT_SOCKETDIR=/tmp/orbit-seed
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/see
d/source/boost_1_64_0/stage/lib:
SHLVL=1
LIBGL_ALWAYS_SOFTWARE=1
J2REDIR=/usr/lib/jvm/java-8-oracle/jre
HOME=/home/seed
QT4_IM_MODULE=xim
DESKTOP_SESSION=ubuntu
GIO_LAUNCHED_DESKTOP_FILE=/usr/share/applications/terminator.deskt
op
QT_LINUX_ACCESSIBILITY_ALWAYS_ON=1
```

```
mf=01;35:*.ogv=01;35:*.ogx=01;35:*.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:*.opus=00
;36:*.spx=00;36:*.xspf=00;36:
XMODIFIERS=@im=ibus
XDG_SESSION_DESKTOP=ubuntu
XAUTHORITY=/home/seed/.Xauthority
XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/seed
SSH_AUTH_SOCK=/run/user/1000/keyring/ssh
TERMINATOR_UUID=urn:uuid:72e960c1-8626-48ee-92da-5916cbcf714c
SHELL=/bin/bash
QT_ACCESSIBILITY=1
GDMSESSION=ubuntu
LESSCLOSE=/usr/bin/lesspipe %s %s
UPSTART_EVENTS=xsession started
GPG_AGENT_INFO=/home/seed/.gnupg/S.gpg-agent:0:1
UPSTART_SESSION=unix:abstract=/com/ubuntu/upstart-session/1000/126
3
XDG_VTNR=7
QT_IM_MODULE=ibus
PWD=/home/seed
JAVA_HOME=/usr/lib/jvm/java-8-oracle
CLUTTER_IM_MODULE=xim
ANDROID_HOME=/home/seed/android/android-sdk-linux
```

System 通过 execl 调用 shell，会自动把环境变量传递过去。

Task 5: Environment Variable and Set-UID Programs

实验源码：

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

```
[09/02/20]seed@VM:~$ sudo chown root f.out
[09/02/20]seed@VM:~$ sudo chmod 4755 f.out
[09/02/20]seed@VM:~$ ls -l f.out
-rwsr-xr-x 1 root seed 7404 Sep  2 06:46 f.out
```

```
[09/02/20]seed@VM:~$ env |grep LD_LIBRARY_PATH
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/see
d/source/boost_1_64_0/stage/lib:
[09/02/20]seed@VM:~$ ./f.out |grep LD_LIBRARY_PATH
[09/02/20]seed@VM:~$ ls -l f.out
-rwsr-xr-x 1 root seed 7404 Sep  2 06:46 f.out
[09/02/20]seed@VM:~$
```

通过上图的实验结果，可得并没有继承父进程的 LD_LIBRARY_PATH 环境变量

```
[09/02/20]seed@VM:~$ export TEST_PATH=/home/seed
[09/02/20]seed@VM:~$ env | grep TEST_PATH
TEST_PATH=/home/seed
[09/02/20]seed@VM:~$ ./f.out |grep TEST_PATH
TEST_PATH=/home/seed
[09/02/20]seed@VM:~$
```

通过上图的实验结果，可得会继承自定义的 TEST_PATH 环境变量

```
[09/02/20]seed@VM:~$ env |grep PATH
LD_LIBRARY_PATH=/home/seed/source/boost_1_64_0/stage/lib:/home/see
d/source/boost_1_64_0/stage/lib:
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:.:/snap/bin:/usr/lib/jv
m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm
/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-linux/platform-tools:/home/seed/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
TEST_PATH=/home/seed
```

```
[09/02/20]seed@VM:~$ ./f.out |grep PATH
XDG_SESSION_PATH=/org/freedesktop/DisplayManager/Session0
XDG_SEAT_PATH=/org/freedesktop/DisplayManager/Seat0
DEFAULTS_PATH=/usr/share/gconf/ubuntu.default.path
PATH=/home/seed/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin:/usr/games:/usr/local/games:.:/snap/bin:/usr/lib/jv
m/java-8-oracle/bin:/usr/lib/jvm/java-8-oracle/db/bin:/usr/lib/jvm
/java-8-oracle/jre/bin:/home/seed/android/android-sdk-linux/tools:
/home/seed/android/android-sdk-linux/platform-tools:/home/seed/and
roid/android-ndk/android-ndk-r8d:/home/seed/.local/bin
MANDATORY_PATH=/usr/share/gconf/ubuntu.mandatory.path
COMPIZ_BIN_PATH=/usr/bin/
TEST_PATH=/home/seed
[09/02/20]seed@VM:~$
```

通过上图的实验结果,可得会继承 PATH 环境变量

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

编译程序后将，g.out 设置为 set-uid 程序

```
[09/02/20]seed@VM:~$ sudo chown root g.out
[09/02/20]seed@VM:~$ sudo chmod 4755 g.out
[09/02/20]seed@VM:~$ ls
android         Desktop          get-pip.py      Music
a.out           Documents        g.out           mysl
bin             d.out            lab1_task2.cpp  Pictures
b.out           Downloads        lab1_task3.c    Public
child           e.out            lab1_task4.c    source
child1          example-content  lab1_task5.c    Templates
c.out           examples.desktop lab1_task6.c    Videos
Customization   f.out            lib
```

第一次运行

```
[09/02/20]seed@VM:~$ ls
android         Desktop          get-pip.py      ls
a.out           Documents        g.out           ls.c
bin             d.out            lab1_task2.cpp  Music
b.out           Downloads        lab1_task3.c    Pictures
child           e.out            lab1_task4.c    Public
child1          example-content  lab1_task5.c    source
c.out           examples.desktop lab1_task6.c    Templates
Customization   f.out            lib             Videos
[09/02/20]seed@VM:~$ ./g.out
android         Desktop          get-pip.py      ls
a.out           Documents        g.out           ls.c
bin             d.out            lab1_task2.cpp  Music
b.out           Downloads        lab1_task3.c    Pictures
child           e.out            lab1_task4.c    Public
child1          example-content  lab1_task5.c    source
c.out           examples.desktop lab1_task6.c    Templates
Customization   f.out            lib             Videos
```

下图为自己编译的 myls 程序：

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;

int main()
{
        cout<<"myls"<<endl;
        return 0;
}
```

运行，发现运行的是自己的 myls 的程序

```
[09/02/20]seed@VM:~$ cp ls ~/bin/
[09/02/20]seed@VM:~$ cd ~/bin/
[09/02/20]seed@VM:~/bin$ ls
ls  md5collgen  myls.out
[09/02/20]seed@VM:~/bin$ cd ~
[09/02/20]seed@VM:~$ ls
android        Desktop            get-pip.py      ls
a.out          Documents          g.out           ls.c
bin            d.out              lab1_task2.cpp  Music
b.out          Downloads          lab1_task3.c    Pictures
child          e.out              lab1_task4.c    Public
child1         example-content    lab1_task5.c    source
c.out          examples.desktop   lab1_task6.c    Templates
Customization  f.out              lib             Videos
[09/02/20]seed@VM:~$ ./g.out
myls
```

Task 7: The LD PRELOAD Environment Variable and Set-UID Programs

根据题意：先编译动态链接库，之后运行，发现链接到自己写的 sleep 函数中

```
[09/02/20]seed@VM:~$ gcc -fPIC -g -c mylib.c
[09/02/20]seed@VM:~$ gcc -shared -o libmylib.so.1.0.1 mylib.o -lc
[09/02/20]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
```

```
[09/02/20]seed@VM:~$ ./myprog
I am not sleeping!
[09/02/20]seed@VM:~$
```

1、Make myprog a regular program, and run it as a normal user

```
[09/02/20]seed@VM:~$ ./myprog
[09/02/20]seed@VM:~$
```

2、Make myprog a Set-UID root program, and run it as a normal user.
将 myprog 的设置为 set-uid 程序

```
[09/02/20]seed@VM:~$ sudo chown root myprog
[09/02/20]seed@VM:~$ sudo chmod 4755 myprog
```

```
[09/02/20]seed@VM:~$ ./myprog
[09/02/20]seed@VM:~$
```

发现执行的是系统自带的 sleep。
原因是，动态链接器的防御措施，特权程序类型的子进程不继承父进程的动态链接库 LD_* 环境变量。

3、Make myprog a Set-UID root program, export the LD PRELOAD environment variable again in the root account and run it.

```
[09/02/20]seed@VM:~$ sudo su
root@VM:/home/seed# export LD_PRELOAD=./libmylib.so.1.0.1
root@VM:/home/seed# ./myprog
root@VM:/home/seed#
```

4、Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account),export the LD PRELOAD environment variable again in a different user's account (not-root user) and run it.

```
root@VM:/home/seed# sudo useradd haha -m
root@VM:/home/seed# sudo chown haha myprog
root@VM:/home/seed# sudo chmod 4755 myprog
root@VM:/home/seed# su seed
[09/02/20]seed@VM:~$ export LD_PRELOAD=./libmylib.so.1.0.1
[09/02/20]seed@VM:~$ ./myprog
Trash seed@VM:~$
```

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

1、正常情况下普通用户不可以删除 root 目录下的文件。但可以强制删除 seed 用户下，root 创建的文件。

```
[09/03/20]seed@VM:~$ ls
android        Desktop          lab1_task9     Public     Videos
bin            Documents        lab1_task9.c   source
bob            Downloads        lib            task9_1
BOB.c          examples.desktop Music          task9_1.c
Customization  get-pip.py       Pictures       Templates
[09/03/20]seed@VM:~$ ./bob "task9_1; /bin/rm /home/seed/task9_1"
/bin/cat: task9_1: Permission denied
/bin/rm: remove write-protected regular file '/home/seed/task9_1'?
 y
[09/03/20]seed@VM:~$ ls
android  Customization  examples.desktop  lib        source
bin      Desktop        get-pip.py        Music      task9_1.c
bob      Documents      lab1_task9        Pictures   Templates
BOB.c    Downloads      lab1_task9.c      Public     Videos
[09/03/20]seed@VM:~$
```

2、如果更改为 execve 没有被删除

```
[09/03/20]seed@VM:~$ sudo chown root bob
[09/03/20]seed@VM:~$ sudo chmod 4755 bob
[09/03/20]seed@VM:~$ ./bob ./task9_1
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
        printf("prinuhg\n");
        return 0;
}

[09/03/20]seed@VM:~$ ls -l
total 1720
drwxrwxr-x 4 seed seed    4096 May  1  2018 android
Firefox Web Browser ed seed    4096 Jan 14  2018 bin
-rwsr-xr-x 1 root seed    7544 Sep  3 11:19 bob
-rw-rw-r-- 1 seed seed     374 Sep  3 11:19 BOB.c
drwxrwxr-x 2 seed seed    4096 Jan 14  2018 Customization
drwxr-xr-x 2 seed seed    4096 Jul 25  2017 Desktop
drwxr-xr-x 2 seed seed    4096 Jul 25  2017 Documents
drwxr-xr-x 2 seed seed    4096 May  9  2018 Downloads
-rw-r--r-- 1 seed seed    8980 Jul 25  2017 examples.desktop
-rw-rw-r-- 1 seed seed 1661676 Jan  2  2019 get-pip.py
```

2.9 Task 9: Capability Leaking

创建/etc/zzz 文件

将自己的写的 task9_1 文件复制到/etc/zzz 文件中

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
        printf("prinuhg\n");
        return 0;
}
```

```
[09/03/20]seed@VM:~$ sudo su
root@VM:/home/seed# cp task9_1 /etc/zzz
cp: cannot stat 'task9_1': No such file or directory
root@VM:/home/seed# su seed
[09/03/20]seed@VM:~$ sudo su
root@VM:/home/seed# cp task9_1.c /etc/zzz
root@VM:/home/seed# ls -l /etc/zzz
-rw-r--r-- 1 root root 109 Sep  3 09:43 /etc/zzz
root@VM:/home/seed# chmod 0644 /etc/zzz
root@VM:/home/seed# ls -l /etc/zzz
-rw-r--r-- 1 root root 109 Sep  3 09:43 /etc/zzz
root@VM:/home/seed#
```

设置特权程序，并运行

```
[09/03/20]seed@VM:~$ sudo chown root lab1_task9
[09/03/20]seed@VM:~$ sudo chmod 4755 lab1_task9
[09/03/20]seed@VM:~$ ./lab1_task9
[09/03/20]seed@VM:~$ sudo cat /etc/zzz
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
int main()
{
        printf("prinuhg\n");
        return 0;
}

Malicious Data
Malicious Data
[09/03/20]seed@VM:~$
```

由上图可知，文件已被修改

```
[09/03/20]seed@VM:~$ rm /etc/zzz
rm: remove write-protected regular file '/etc/zzz'? y
rm: cannot remove '/etc/zzz': Permission denied
[09/03/20]seed@VM:~$
```

但在正常情况下用户没有权限删除文件。

实验总结：通过这次实验，我们对环境变量与 set-uid 程序的了解更深一步，通过实验，更为形象化的对环境变量的作用，功能，以及一些潜在传递规则都有了更深一步的了解