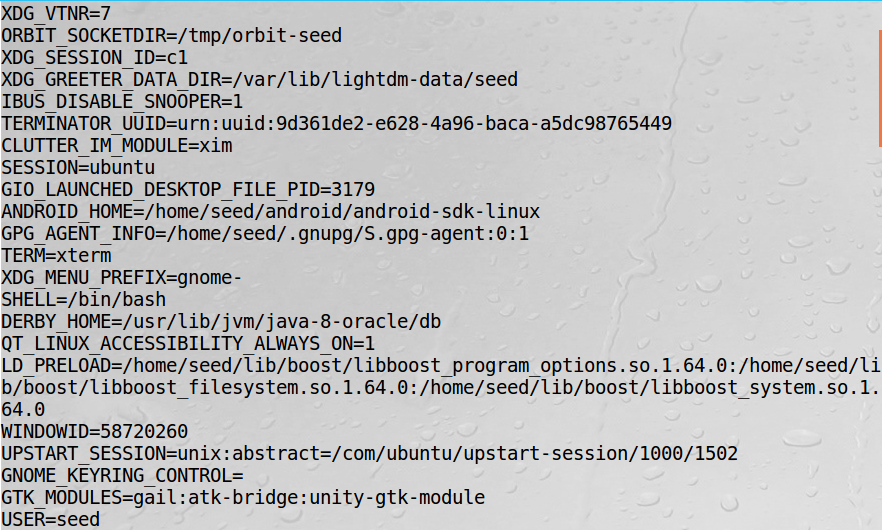
**Lab1**

张镇57117126

Task1

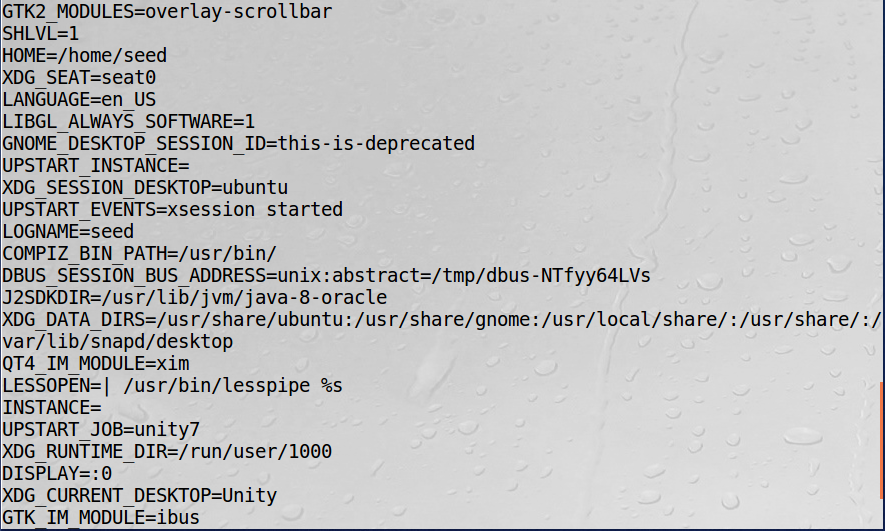
实验过程一：使用printenv或者env打印出环境变量

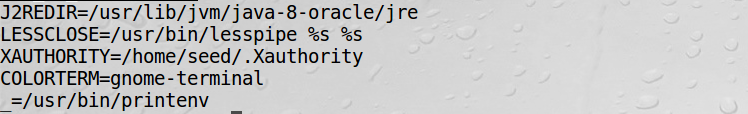
在终端输入命令，显示结果如下图所示：



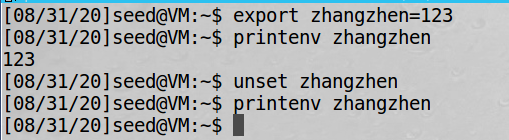






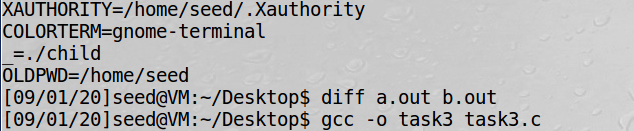


实验过程二



Task2

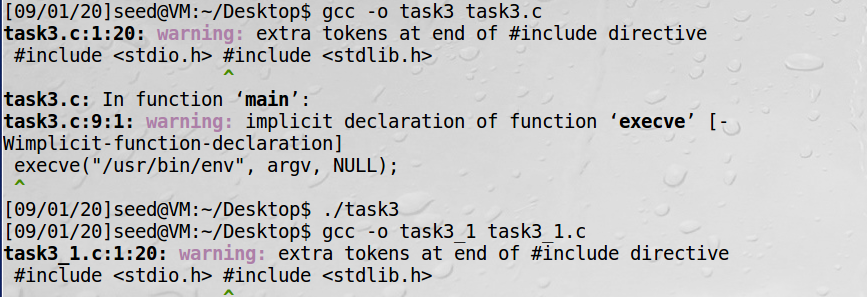
实验过程：将实验手册中二维码在虚拟机中创建一个child.c的文件，编译并运行，将输出结果保存到a.out中，再将子进程注释掉，编译运行将输出结果保存到b.out。最后用diff a.out b.out命令比较俩个输出的不同。



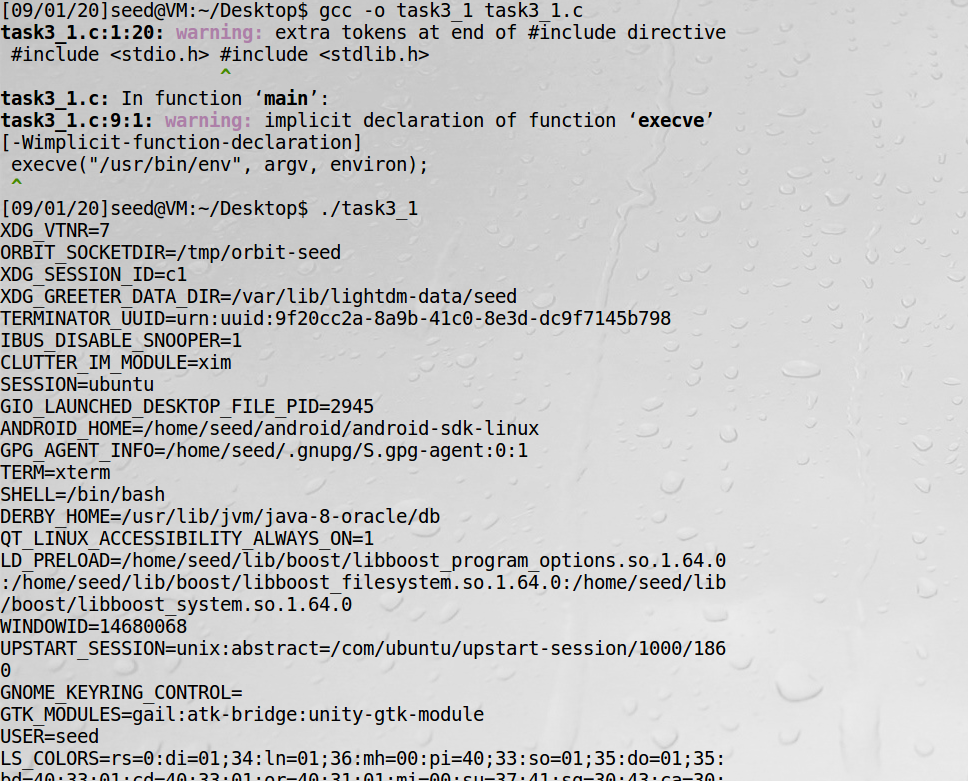
实验结论：通过比较这两个文件，可以发现，这两个文件输出的环境变量完全相同。说明原环境变量被子进程完全继承。通过man fork，对fork函数做了进一步了解。fork函数通过系统调用创建一个与原来进程几乎完全相同的进程，子进程自父进程继承了进程的资格，环境，堆栈与内存根目录等；但是子进程没有继承父进程的某些特性，比如父进程号，文件描述符，在tms结构中的系统时间，资源使用等。

Task3

实验过程一：编译并运行pdf中所给代码，如下图所示：



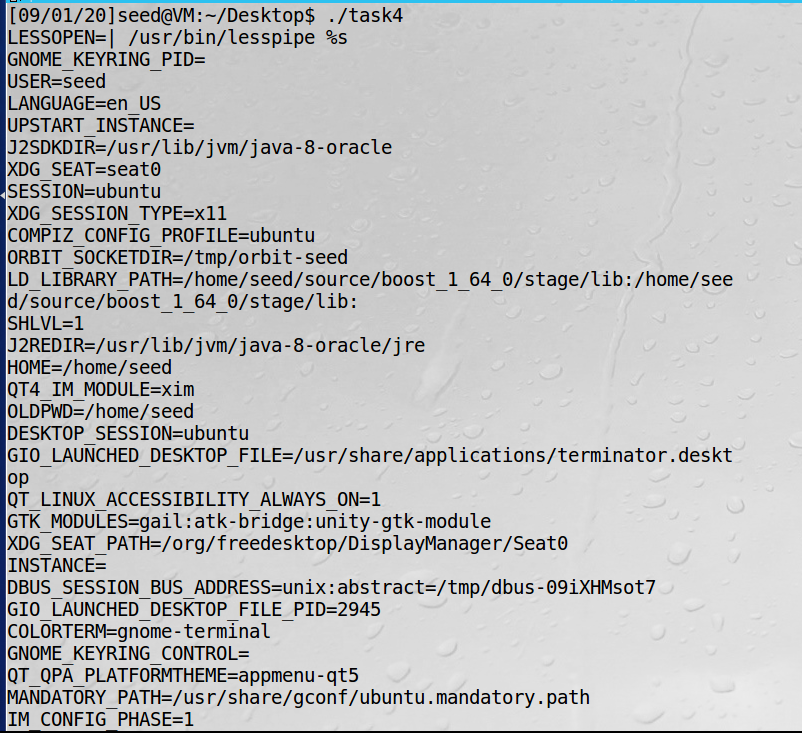
将execve中的参数NULL改为environ，编译并运行，如下图所示：



实验结论：execve()用来执行参数filename字符串所代表的文件路径，filename必须是一个二进制的可执行文件，或者是一个脚本以#!格式开头的解释器参数。如果是后者，这个解释器必须是一个可执行的有效的路径名。第二个参数系利用数组指针来传递给执行文件，argv是要调用的程序执行的参数序列，也就是我们要调用的程序需要传入的参数。envp则为传递给执行文件的新环境变量数组,同样也为参数序列。

Task4

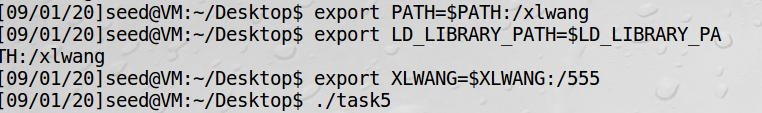
实验结果：编译并运行所给代码，如下图所示：



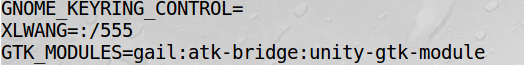
实验分析：该函数调用/bin/sh来执行参数指定的命令,/bin/sh一般是一个软连接，指向某个具体的shell，比如bash,-c 选项是告诉shell从字符串command中读取命令；在该command执行期间，SIGCHLD信号会被暂时搁置，SIGINT和SIGQUIT则会被忽略，意思是进程收到这两个信号后没有任何动作。

Task5

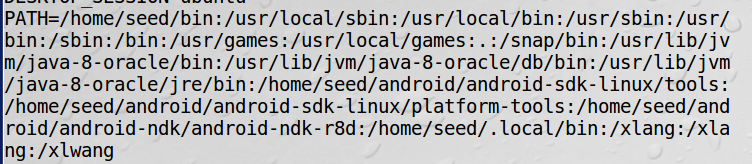
实验过程：首先将pdf上的代码编译一下，然后第二步通过sudo chown root task5.c和sudo chmod 4755 task5.c俩个命令将这个程序变为set-uid程序，最后设置三个环境变量PATH、LD LIBRARY PATH、XLWANG，过程如下图所示：



运行上述可执行文件，可以发现三个环境变量都包括在其中：



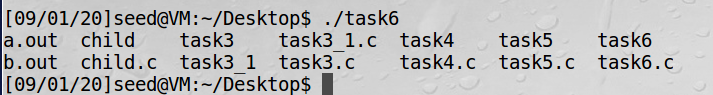
C:\Users\lenovo\AppData\Roaming\Tencent\Users\345312414\QQ\WinTemp\RichOle\_IP~CEP1Q1)Z$T(R{UOECYR.png



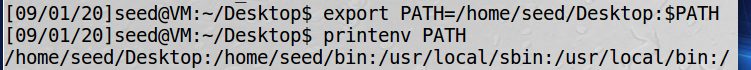
Task6

实验结果：

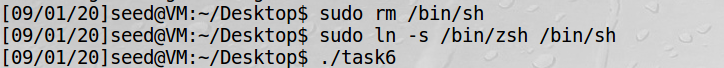
编译运行task6.c，结果如下：



修改环境变量PATH，在其最前端插入当前的工作目录：



做一定的处理：



再将/bin/sh 复制到当前目录，命名为 ls：

C:\Users\lenovo\AppData\Roaming\Tencent\Users\345312414\QQ\WinTemp\RichOle\OM4B8@[G7$V6PO3A$C8SLRS.png

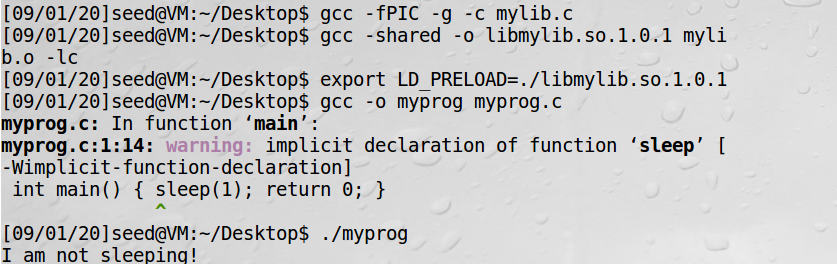


实验结论：

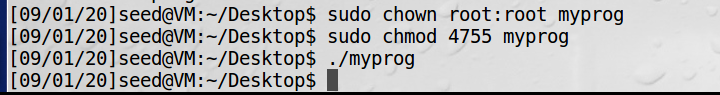
Set-UID 程序运行的子程序是具有 root 权限的。PATH 环境变量为可执行程 序的搜索路径，可通过修改 PATH 路径调整程序搜索目录的顺序。

Task7

根据第一步指导实验结果如下：

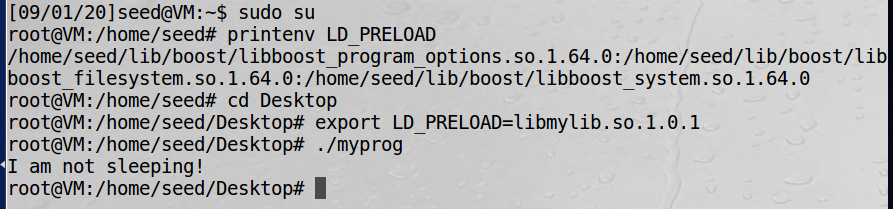


将 myprog 可执行程序变为 set-UID root 程序，再次运行：

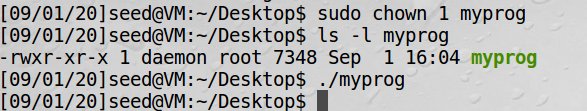


发现在停顿过后直接进入下一行，没任何输出，说明此时 myproc 程序

对应的 LD\_PRELOAD 环境变量并非 seed 用户下所设置的 LD\_PRELOAD 值。



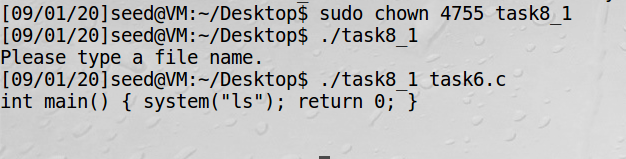
root权限下运行结果如上图，新建一个账号1，再次运行：



依然没有输出，得出结论，由以上实验进行对比，可以得出结论，只有 ruid 和 euid 相同时， LD\_PRELOAD变量才会有效。当二者有差异时，该变量不发挥作用。

Task8

按照pdf上的指导将task8\_1.c编译。并设置为set-uid程序，打开我这个工作目录下的task6.c：



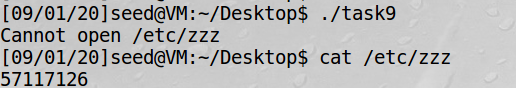
然后新建一个task8\_2.c编译，并设置为set-uid程序，同样打开这个工作目录下的task6.c，此时发现不能打开了。

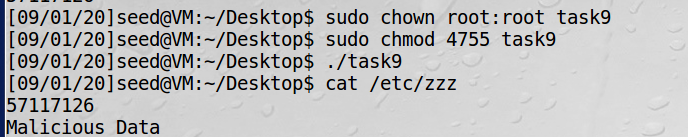
实验结论：

Execve 与 sysytem 方法的底层逻辑不同，system 方法本质上是开启新的shell 去执行一段 cmd，而 execve 则是传入可执行程序的参数的方式执行新的程序，execve 方法显然更加安全。System 方法在特权程序中被调用会显得十分不安全。

Task9

首先在/etc创建一个zzz，写入我的学号57117126，zzz的文件此时只有root权限才能对其进行写，然后对task9.c进行编译、权限修改，变为set-uid程序：





可以看到zzz文件已经被修改。

实验结论：

之所以 zzz 文件内容可以被修改，是因为在 task9 可执行程序运行时打开了zzz 文件，在执行完相关任务之后却没有在撤销特权前及时关闭 zzz 的文件描述符 fd，此时 fd 还具有 root 特权，因此可以执行对 zzz 文件的写操作。