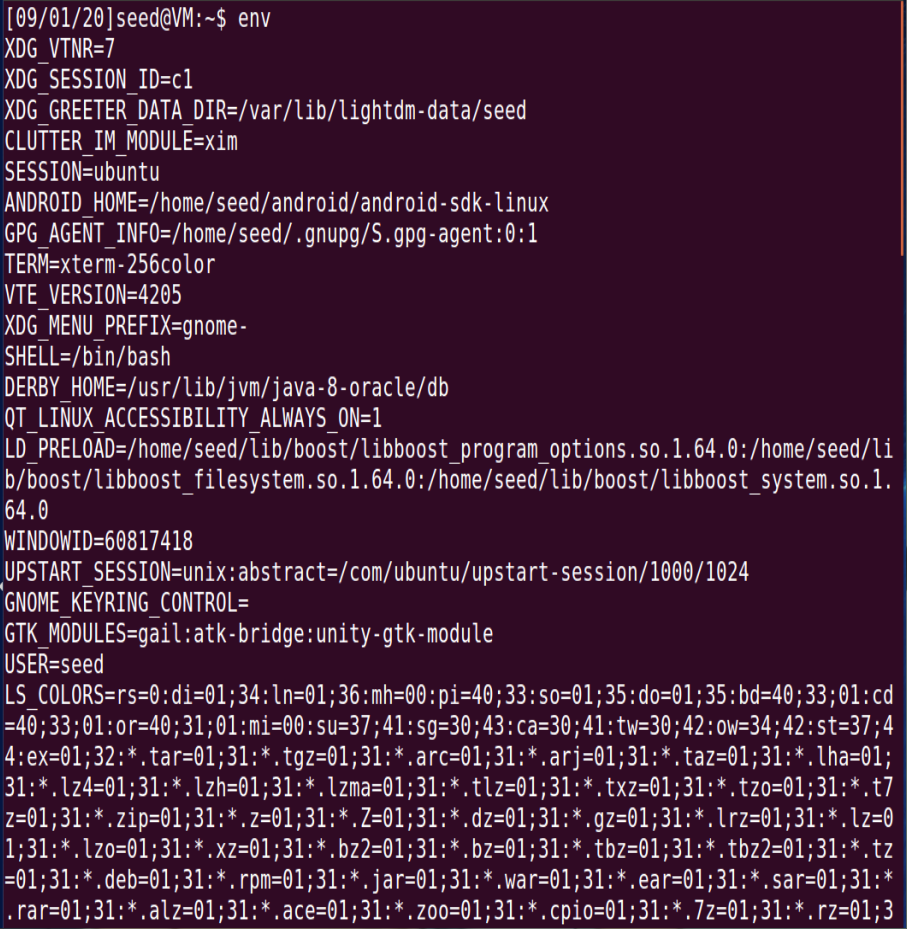
**网络空间安全实训Lab[1]**

**实验报告**

**57118117 谌雨阳**

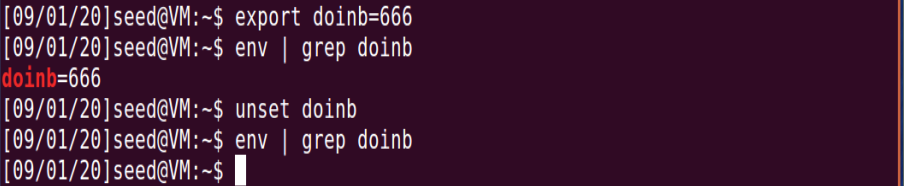
**Task1: Manipulating Environment Variables**

1、使用env命令显示环境变量：

2、使用printenv或env | grep查看特定环境变量

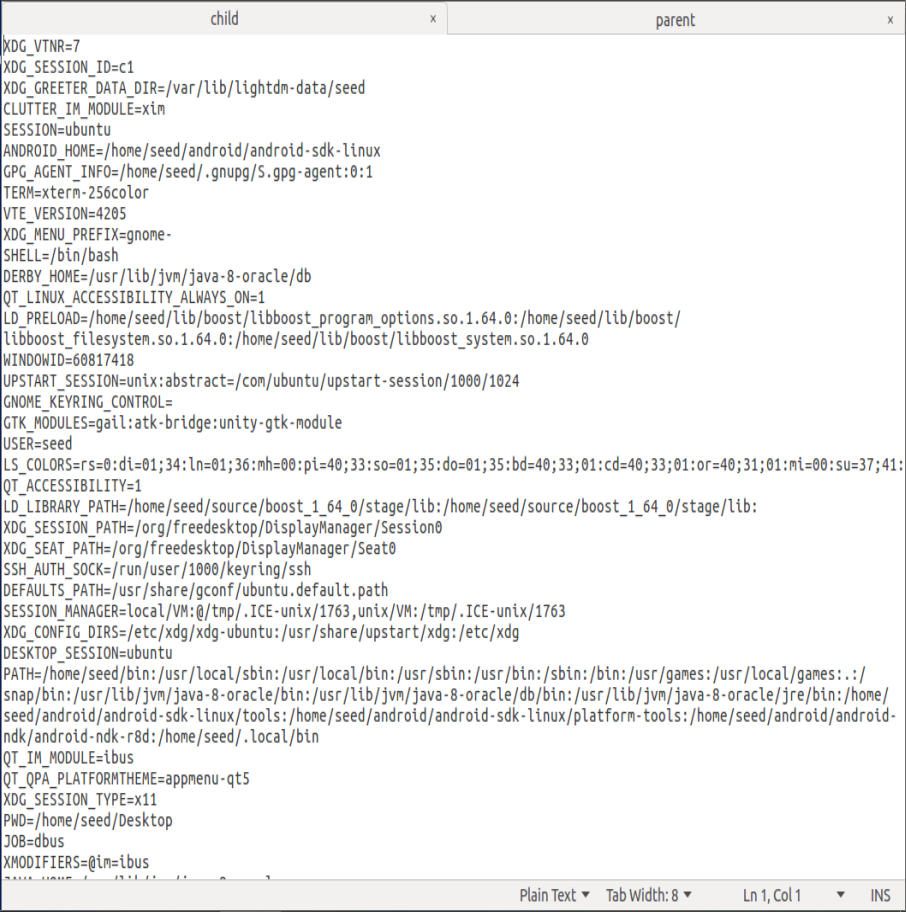


3、使用export和unset命令设置和清除环境变量

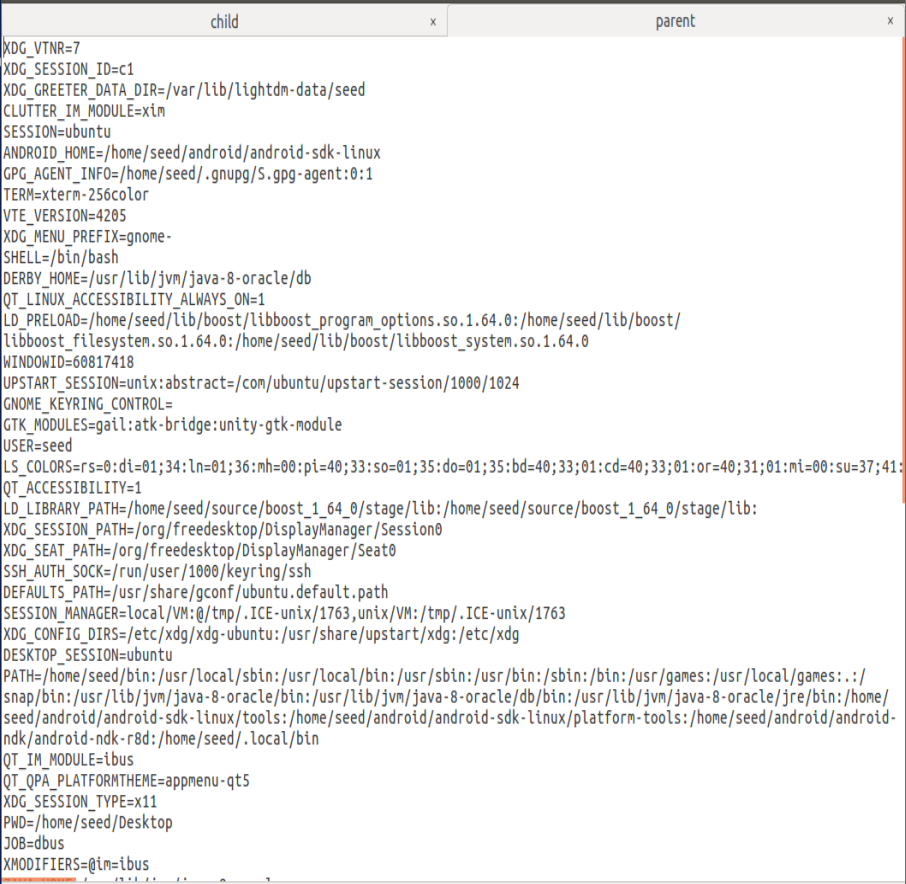


**Task2: Passing Environment Variables from Parent Process to Child Process**

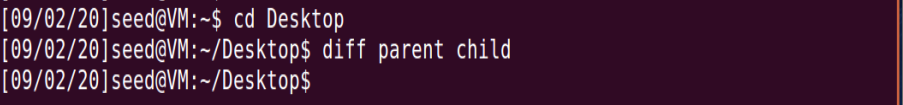
1、Step1（child）运行结果：



2、Step2（parent）运行结果：



3、使用diff命令比较两者区别：



结论：两个文件输出结果一致。

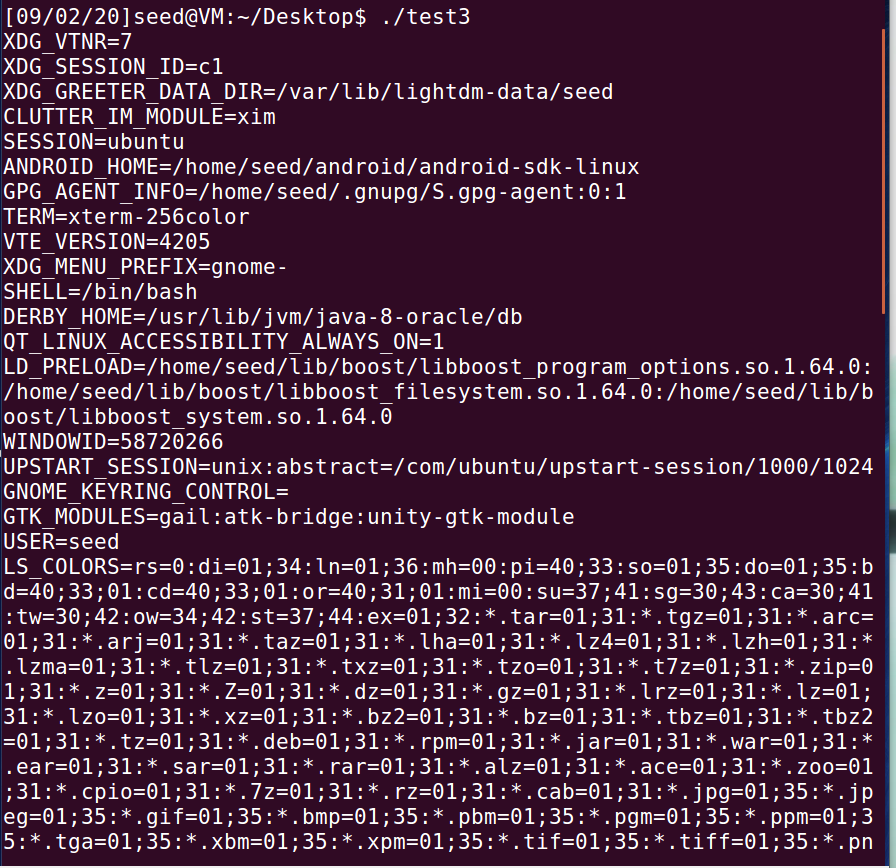
总结分析：两个文件输出的环境变量完全相同。说明子进程完全继承了原环境变量。通过manfork了解fork函数的内容如下：fork函数创建的是一个与原来进程几乎完全相同的进程，子进程自父进程继承了进程的资格，环境，堆栈与内存根目录等；但是子进程没有继承父进程的某些特性，比如父进程号，文件描述符，在tms结构中的系统时间，资源使用等。

**Task3: Environment Variables and execve()**

1. Step1（NULL）运行结果：



2、Step2（environ）运行结果：

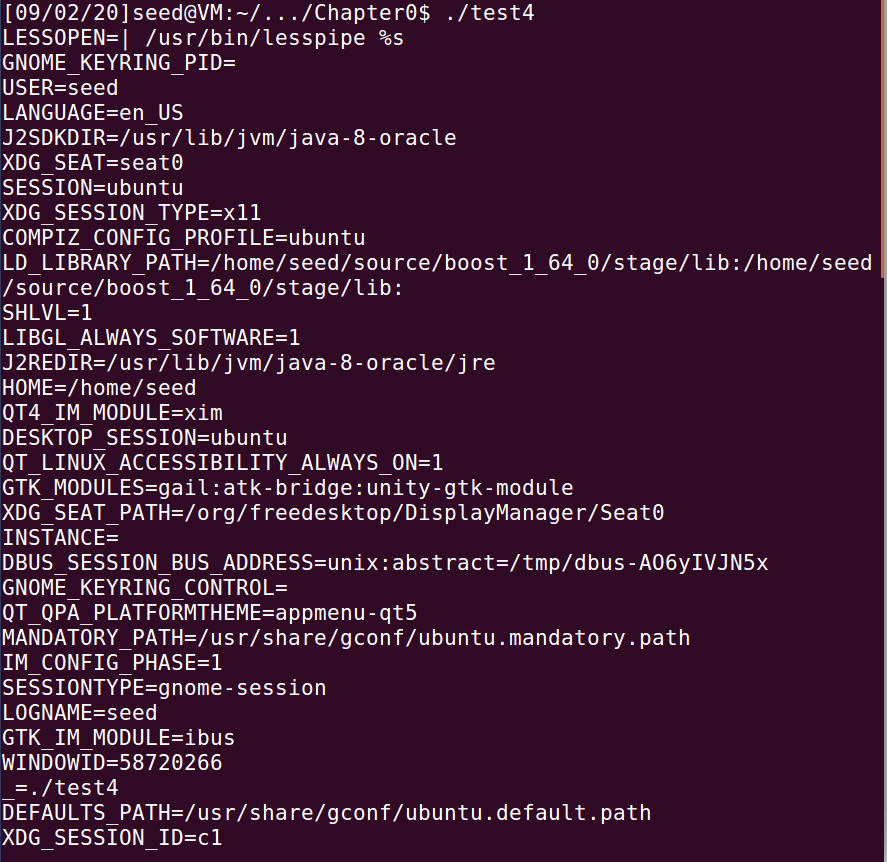


结论：两个文件输出结果不同新程序获取到了环境变量而原来的程序没有。

总结分析：通过查找资料了解到了execve函数的用法：int execve(const char \* filename，char \* const argv[]，char \* const envp[])。更改的参数为第三个envp，envp为传递给执行文件的新环境变量数组,同样也为参数序列，是实现功能所必要的一个参数。

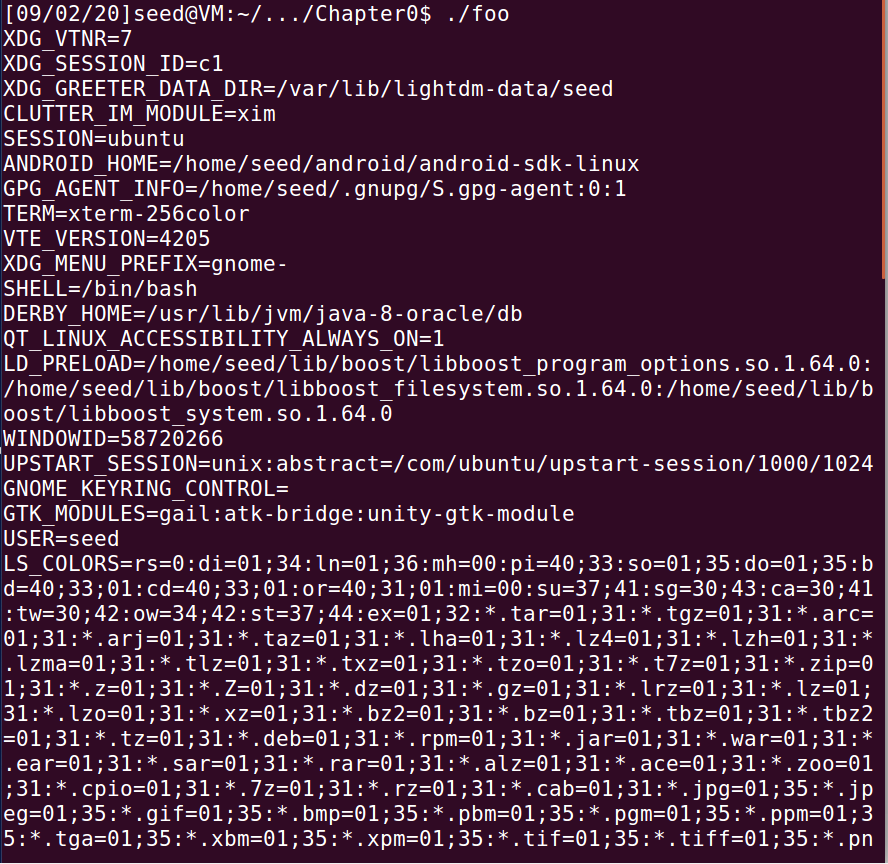
**Task4: Environment Variables and system()**

运行结果：



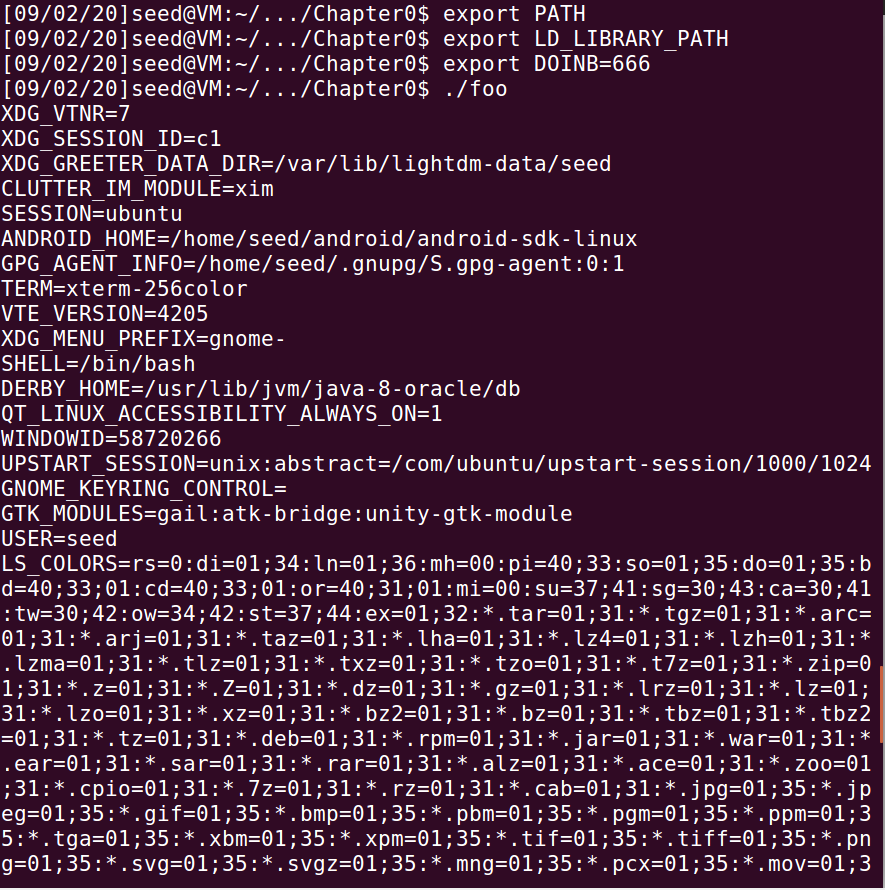
**Task5: Environment Variableand Set-UID Programs**

1、Step1运行结果：

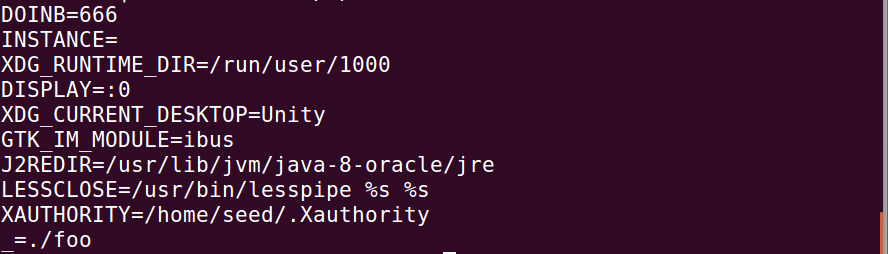
2、将程序所有者移交给root并设置为setUID程序：



3、设置环境变量重新运行程序：



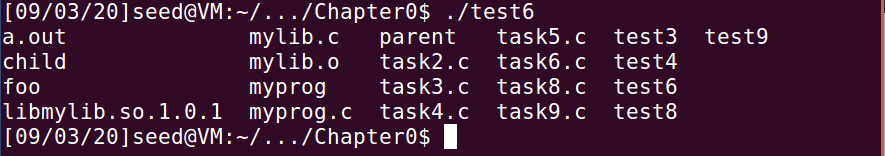
4、寻找自行设置的环境变量：



观察描述：自行设置的几个环境变量出现在了程序运行结果中。

**Task6: The PATH Environment Variable and Set-UID Programs**

将程序设置成为SET-UID root程序后运行结果如下：



从上图可以看出该程序执行的是ls命令。

执行手册中Note中的命令后即可获得root权限。

**Task7: The LD\_PRELOAD Environment Variable and Set-UID Programs**

四种情况运行结果如下：

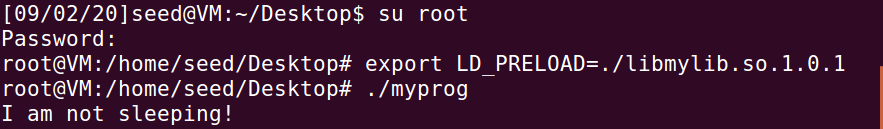
1. Regular program & normal user:



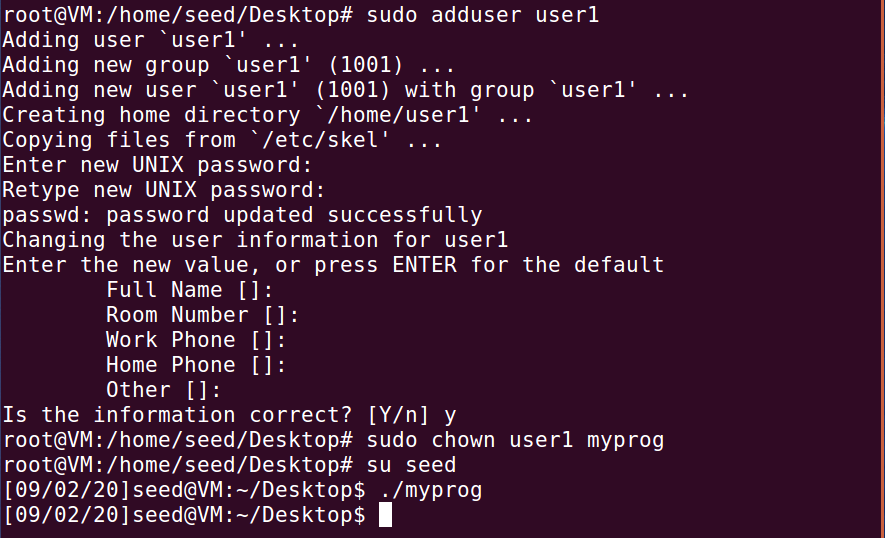
2、SET-UID root program & normal user:



3、SET-UID root program & root user



4、SET-UID user1 program & different user



解释：

从上面的运行结果中可以看出。第一次和第三次运行结果为输出“I am not sleeping”，第二次和第四次是暂停一秒。造成该区别的原因是：LD\_PRELOAD环境变量可以影响程序的运行时的链接，它允许定义在程序运行前优先加载的动态链接库。在该实验中，mylib.c通过sleep函数，生成了一个libmylib.so.1.0.1链接库。然后将该链接库添加到LD\_PRELOAD环境变量上。第一次和第三次实验myprog程序均具有seed用户权限，而在seed用户的LD\_PRELOAD环境变量中也添加了该链接库。因此，这两次都运行的是用户定义的sleep函数。

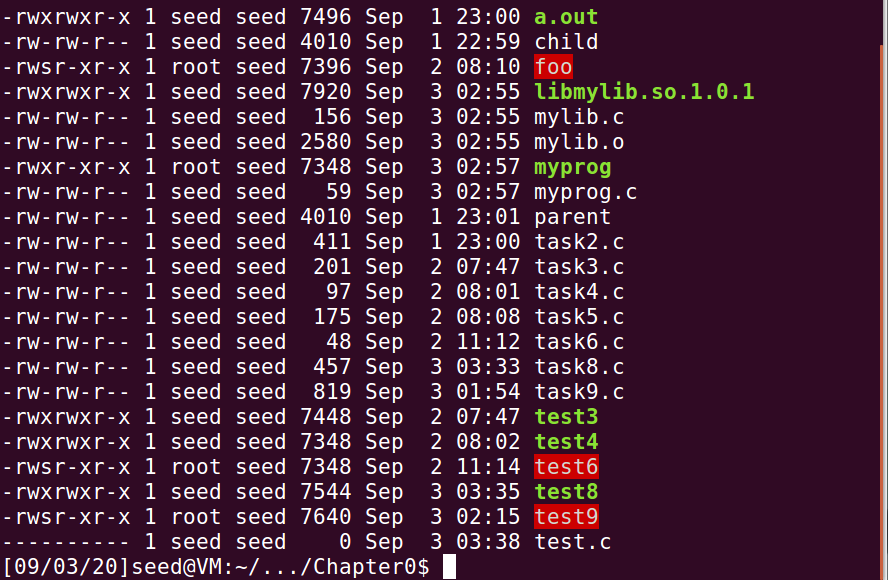
**Task8: Invoking External Programs Using system() versus execve()**

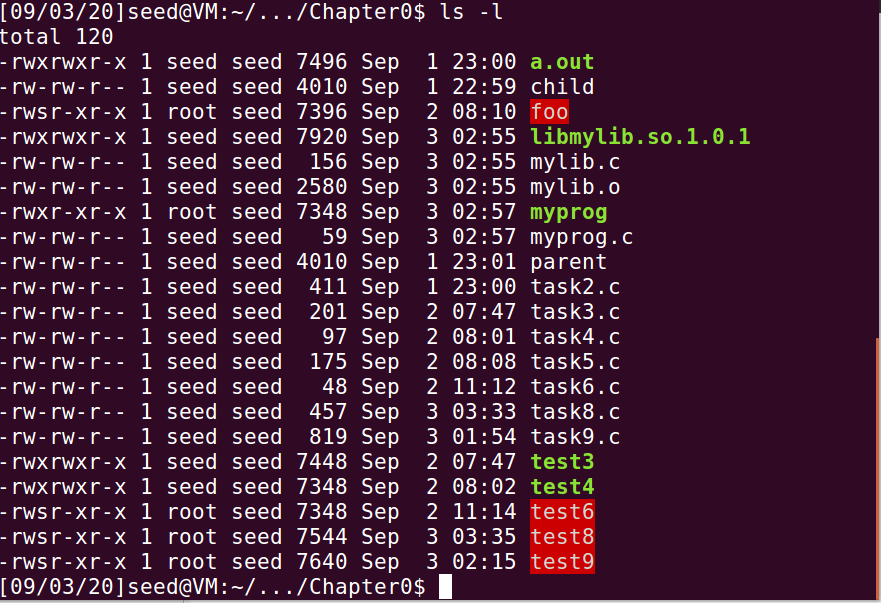
编译题目中所给的代码并将其设置为SET-UID后，使用如下方式尝试实现文件删除：

1、新建一个test.c文件并将其权限设置为000：

2、将原代码中的v[0] = "/bin/cat"改成v[0] = "rm"即删除指令，使其在后面的system指令中被调用。

3、查看现在文件夹中有的文件：



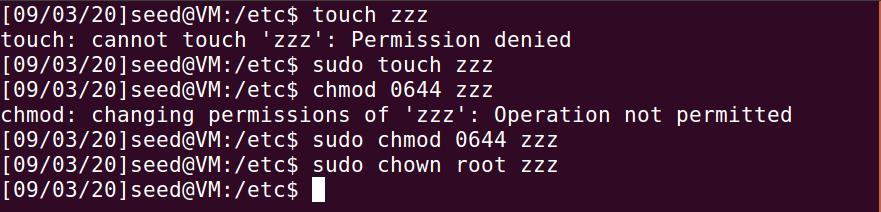
4、运行更改过后的test8程序：发现test.c文件被删除了。  


结论：本来test.c对seed用户是不可写的，但因为task8是SET-UID程序，暂时拥有root权限，因此可以使用system调用rm命令删除test.c文件。由此可见，set-UID程序具有一定的危险性。

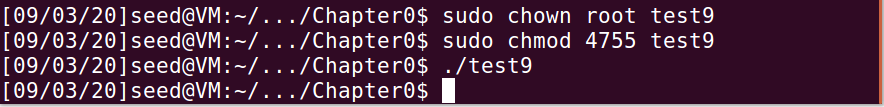
Step2：将system（commend）注释掉并取消注释execve后实现不了删除该文件的功能，因此该攻击无效。

**Task9: Capability Leaking**

1、/etc/zzz文件创建及权限设置：



2、程序owner、权限设定及运行结果：

3、程序运行后的/etc/zzz文件：



观察解释：原本的空文件被写入了程序中制定的一行文字。说明在普通用户下也可以对该文件进行改动，即验证了权能泄露的问题，存在安全风险。