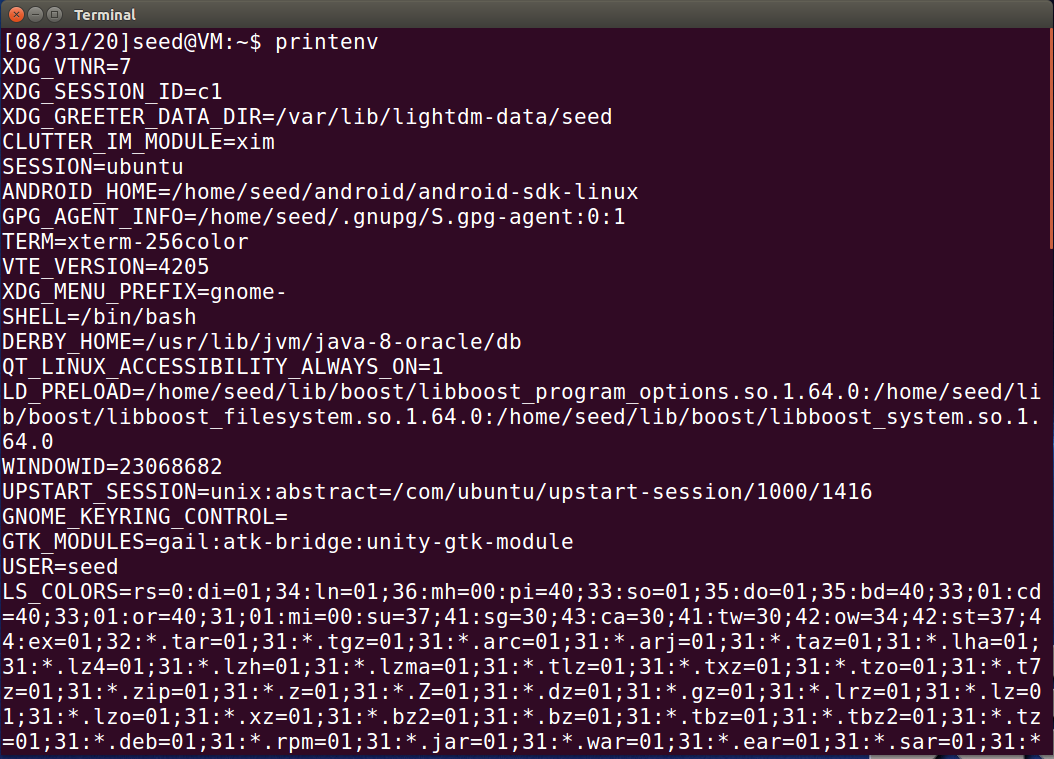
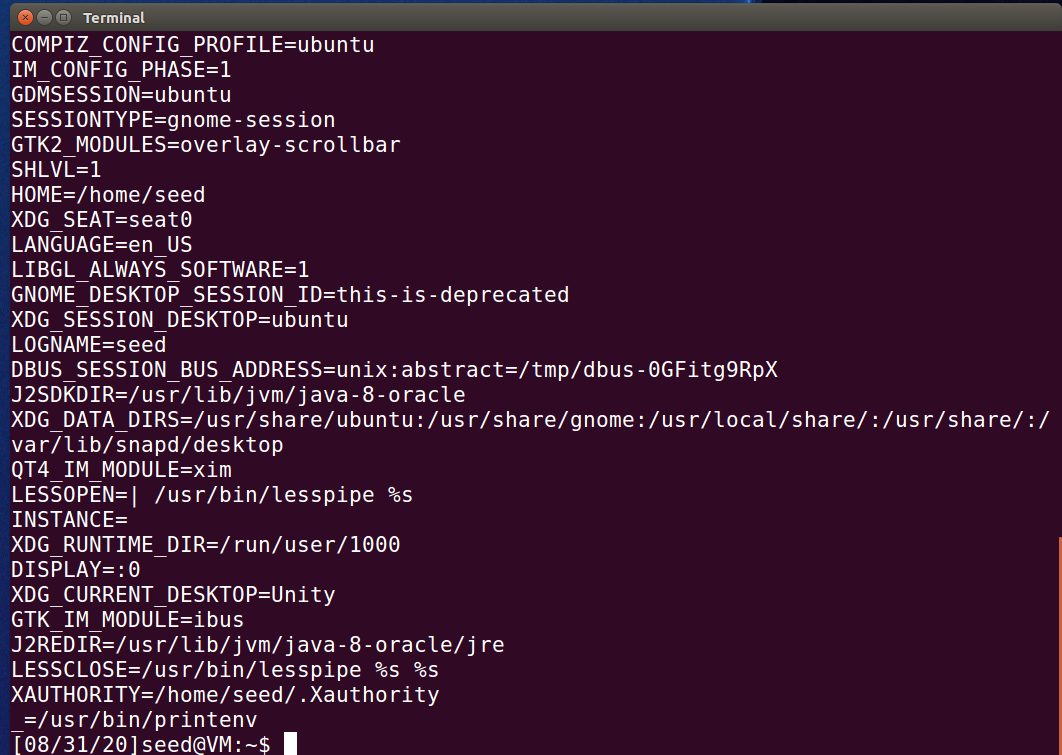
**lab1实验报告**

57117133 臧宇豪

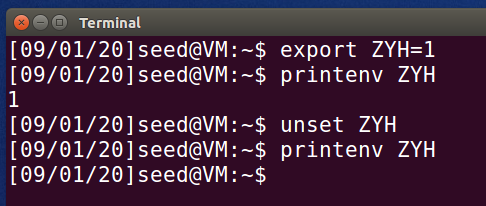
**Task 1: Manipulating Environment Variables**

使用命令打印环境变量:

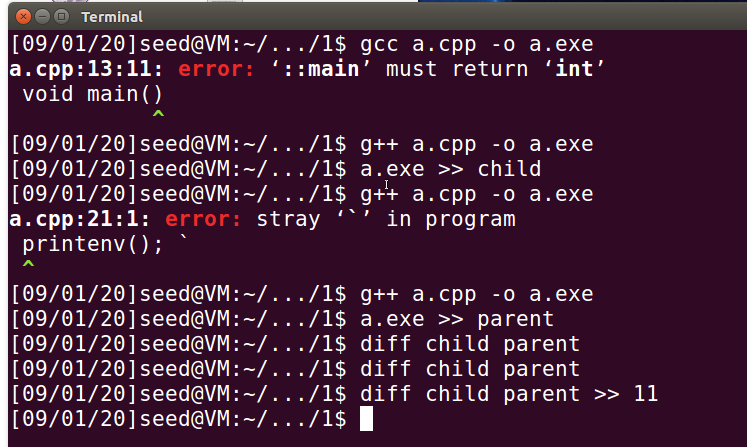




设置和取消环境变量：



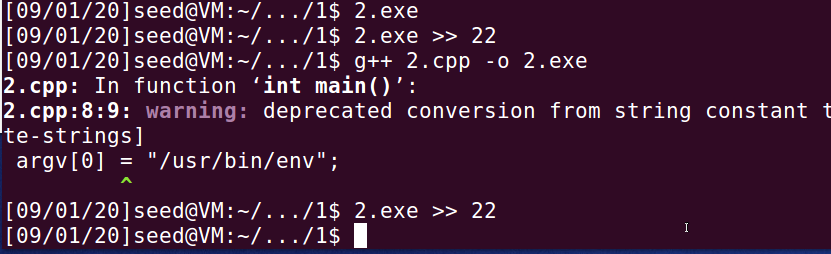
**Task 2: Passing Environment Variables from Parent Process to Child Process**



可以发现父进程和子进程的输出用diff比较之后发现没有输出结果，即没有区别，因此我们可以判断父进程的环境变量被子进程继承了。

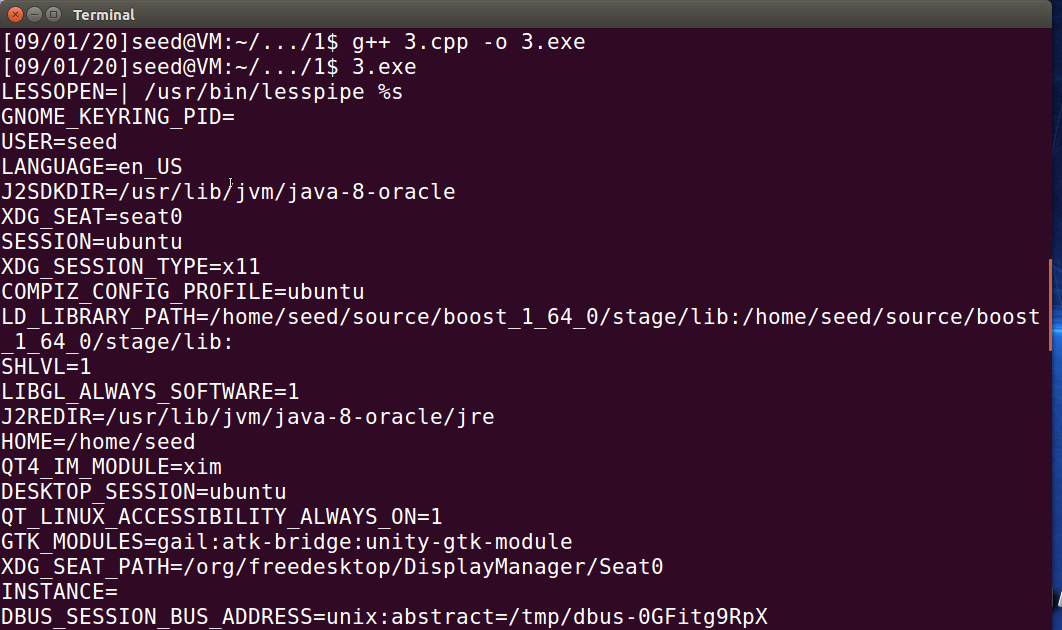
**Task 3: Environment Variables and execve()**

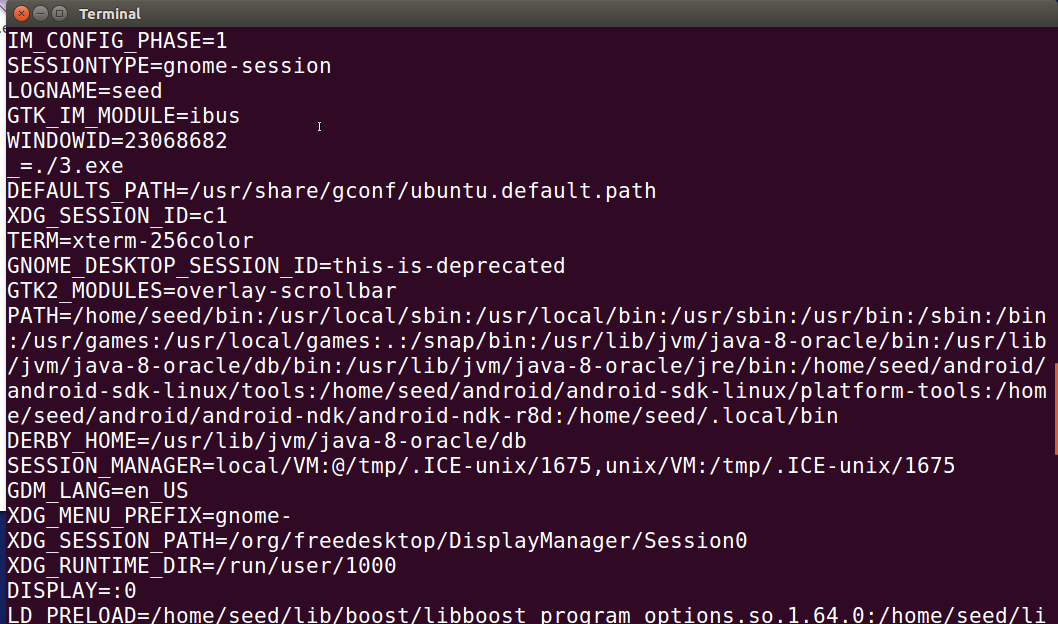
可以发现第一步的程序结果没有输出，第二步更改后输出了系统的环境变量。第二步的程序获取环境变量是通过调用操作系统已经定义的environ，这个变量关联着操作系统的环境变量。

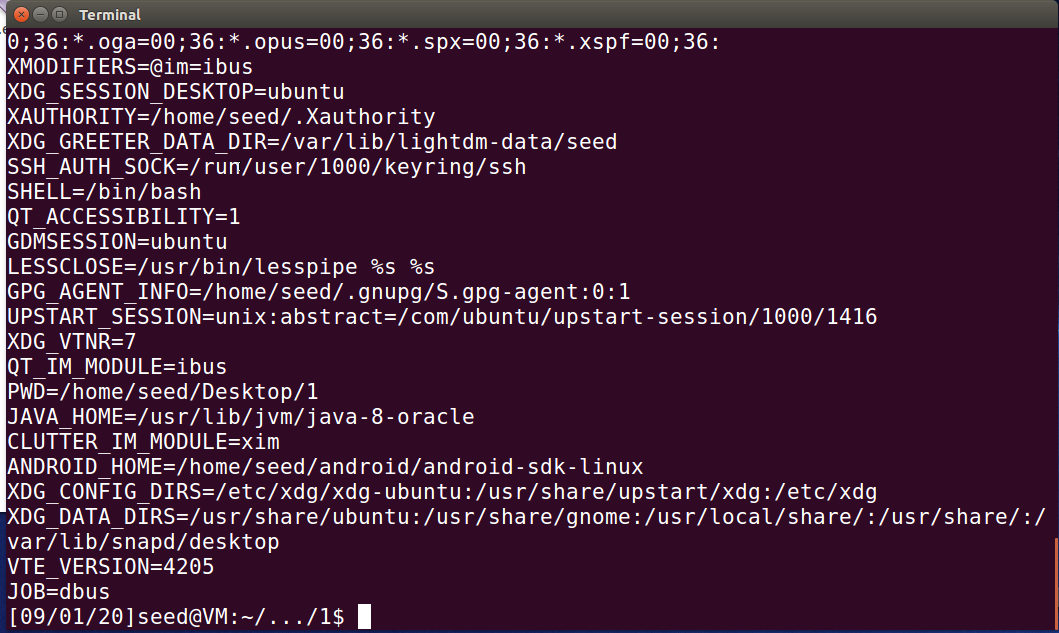


**Task 4: Environment Variables and system()**

程序运行结果如图所示：





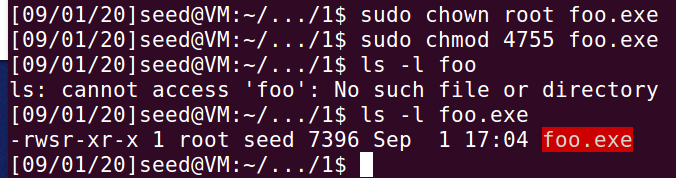


**Task 5: Environment Variable and Set-UID Programs**

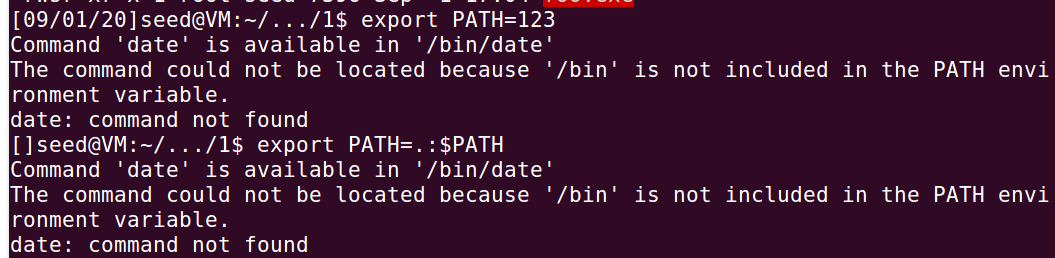
第一步：编译运行程序

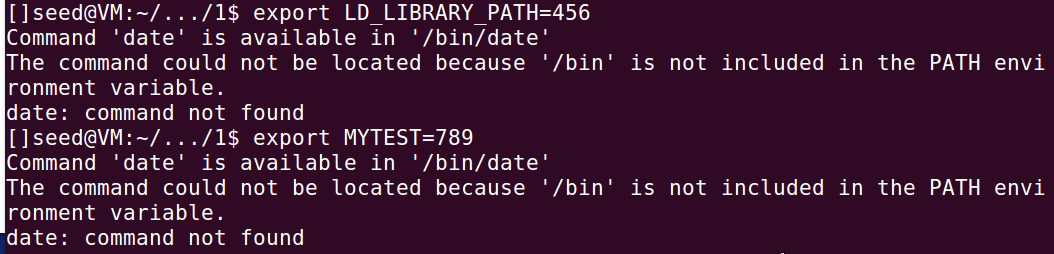


第二步：设置Set-UID

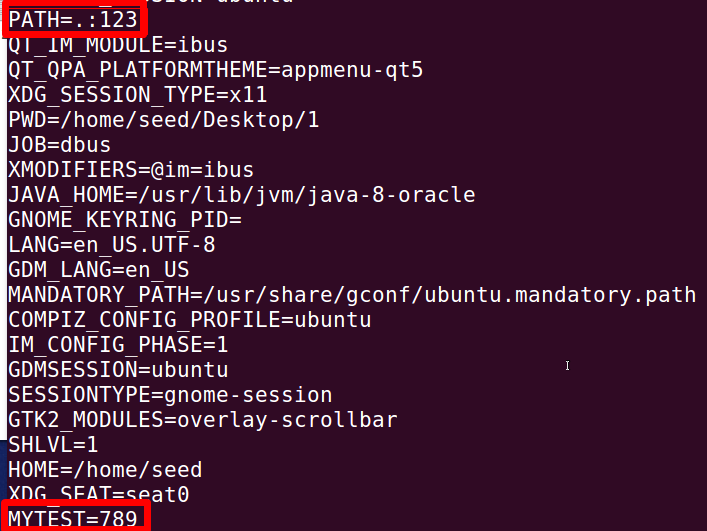


第三步：更改环境变量



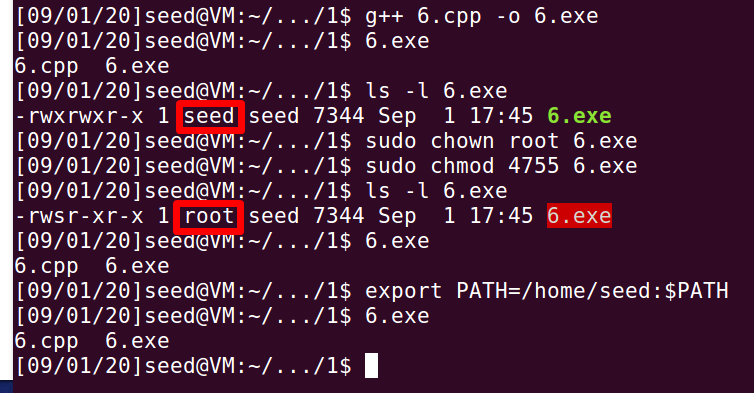


结果发现：再次执行foo.exe之后，发现结果中的目标环境变量已经变成了上一步设置的结果:



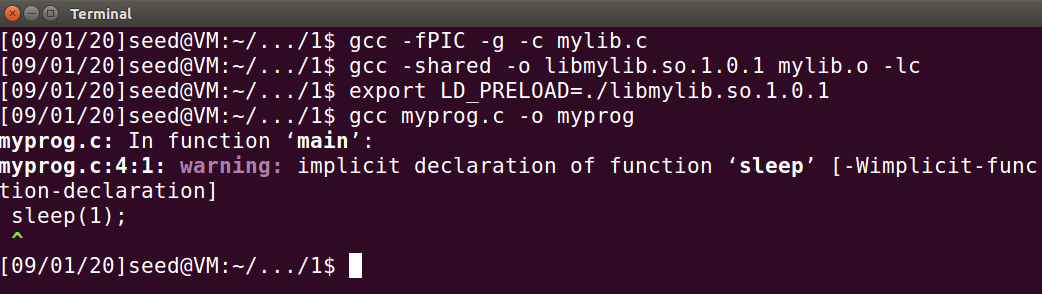
**Task 6: The PATH Environment Variable and Set-UID Programs**

如图所示：

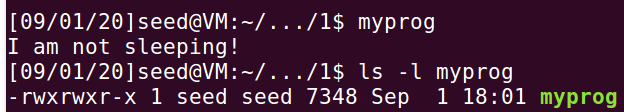


**Task 7: The LD PRELOAD Environment Variable and Set-UID Programs**

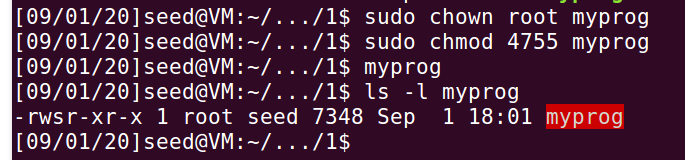
准备和配置：



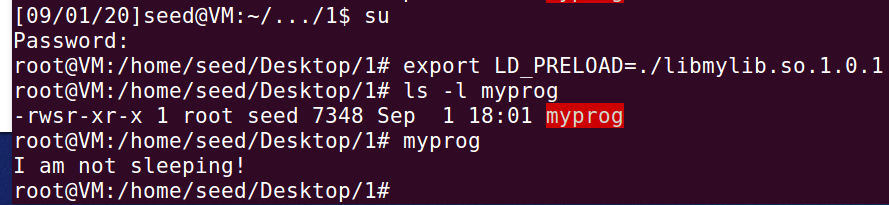
1. 作为常规程序运行:



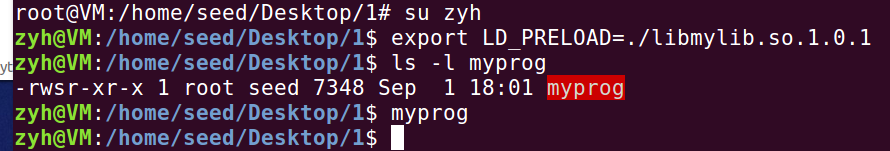
2. 以普通用户身份运行SetUID程序：



3. 以root用户运行SetUID程序



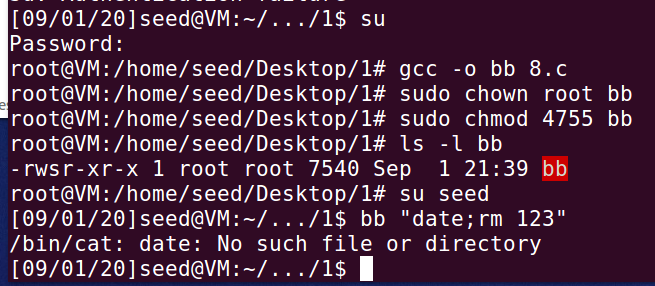
4. 用其他普通用户运行SetUID程序：



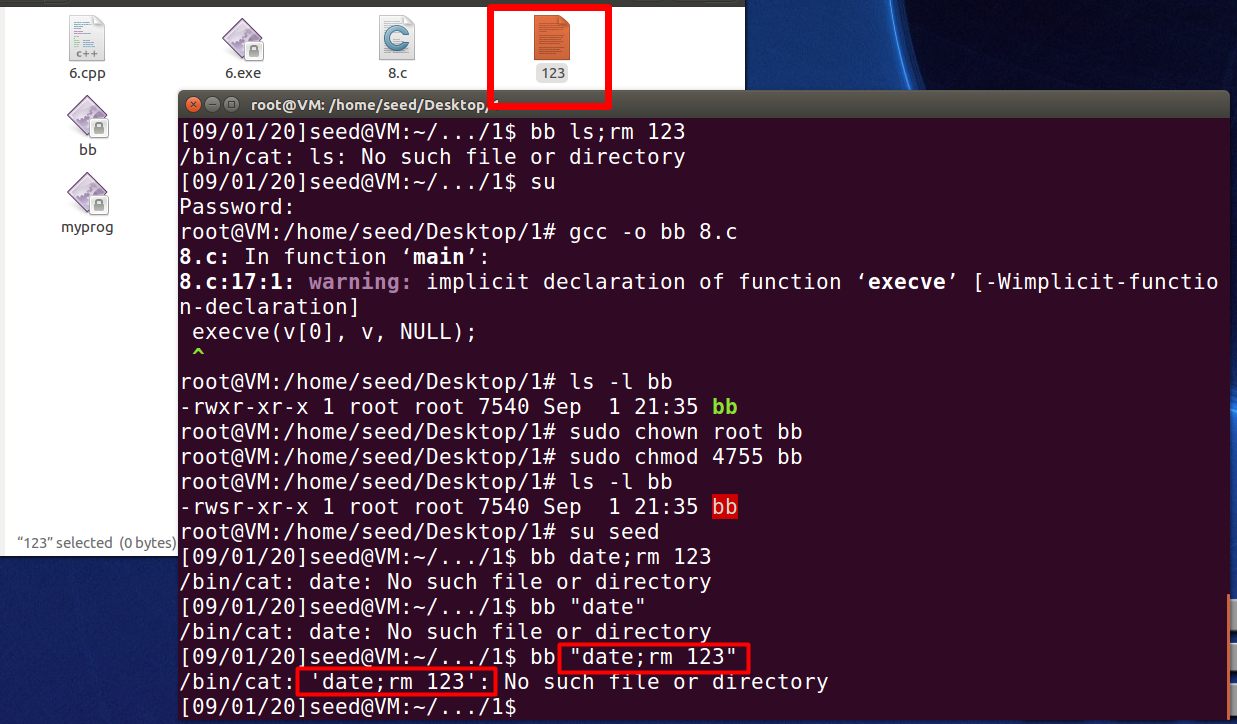
通过以上四个实验步骤我们可以发现，步骤2和步骤4由于子进程无法继承LD\_\*类型的环境变量，因此在执行SetUID程序时，因为EUID和RUID不同，因此程序忽略了LD\_PRELOAD 环境变量。而步骤一和步骤三可以看出这两个情况下对于环境变量的修改是有作用的。

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

使用system() 时，成功删除文件123：

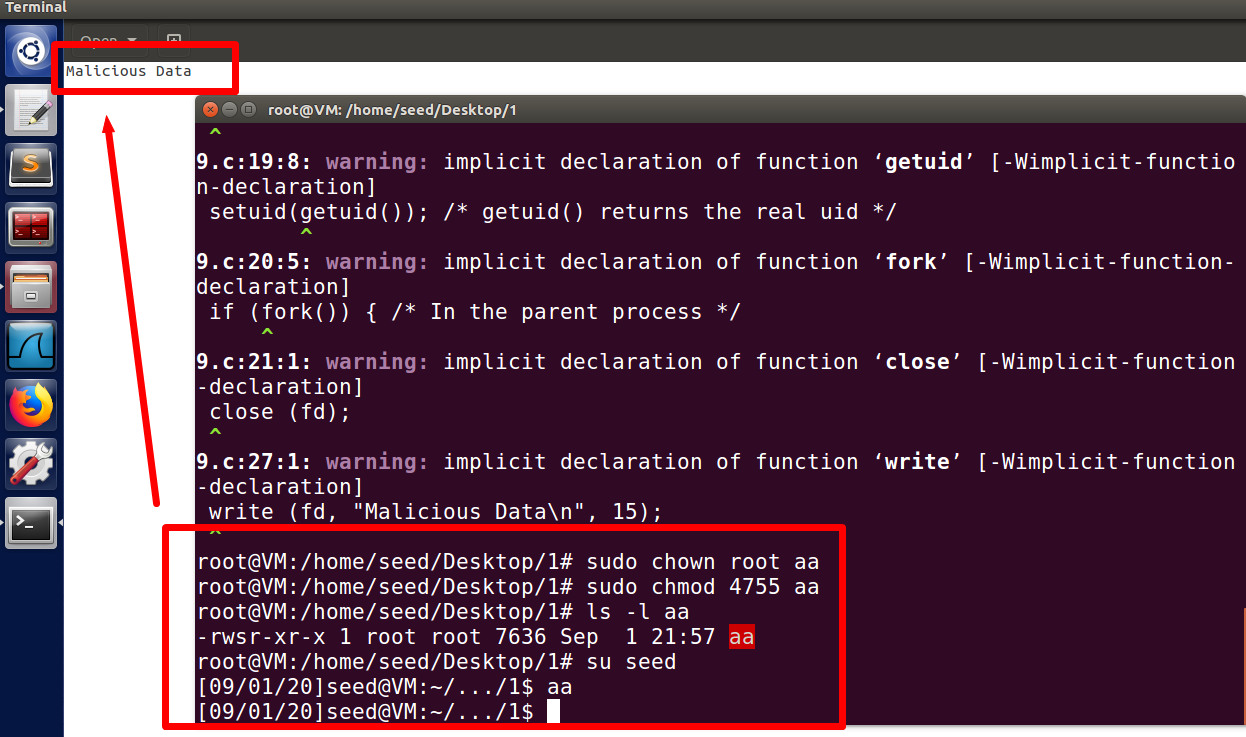


使用execve() 时，删除123失败，第一步的攻击失效：



**Task 9: Capability Leaking**

通过实验可以发现 /etc/zzz 文件被修改了：



这一现象发生的原因可能是父进程完成任务后放弃了root权限，然后fork()的子进程仍然持有root权限，造成了权限能力的泄露。