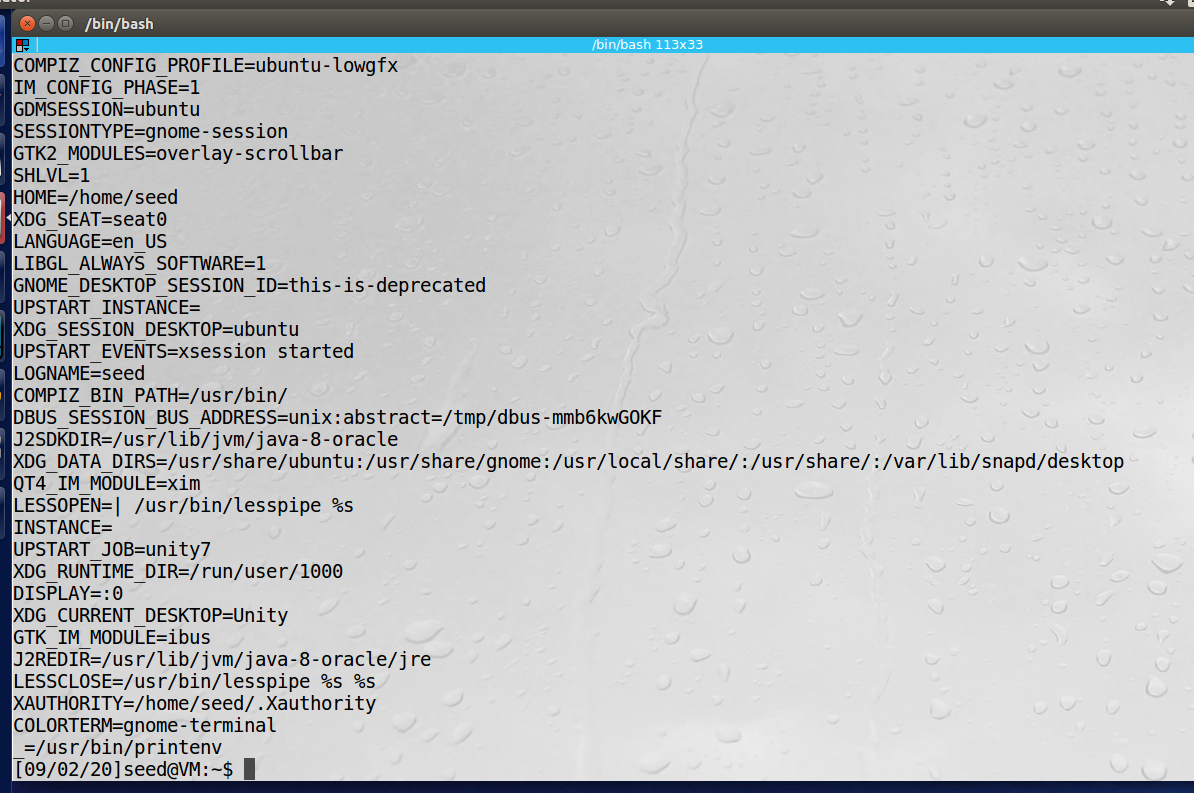
**Lab1**

**57117206张子燕**

**Task1**

输出所有环境变量：



输出特定的环境变量：

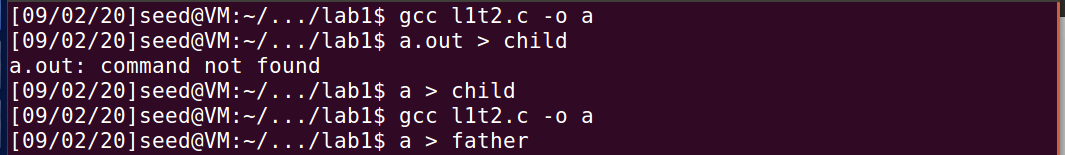


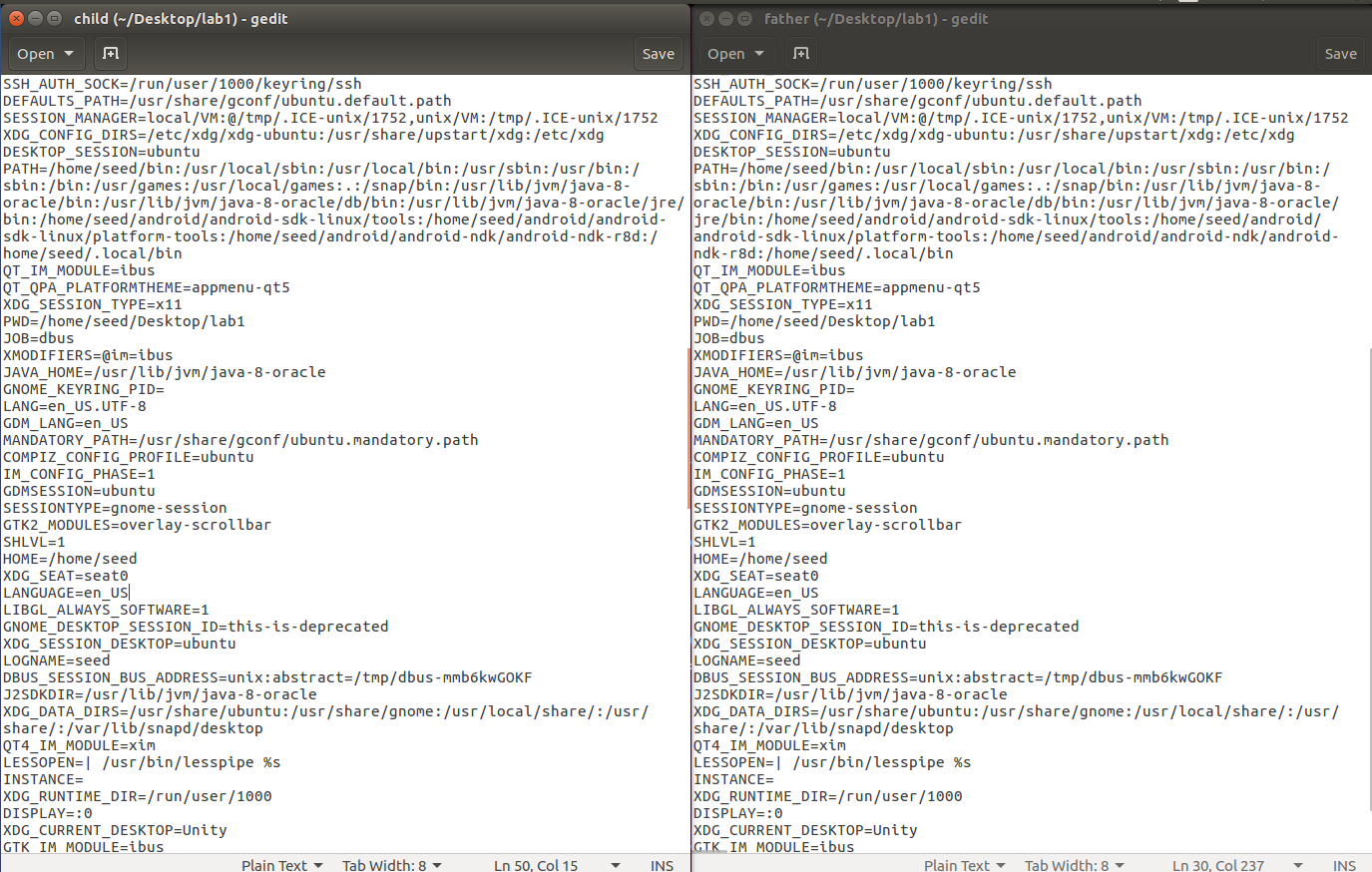
设置一个新的环境变量并删除：



**Task2**

运行并输出子进程、父进程的环境变量：





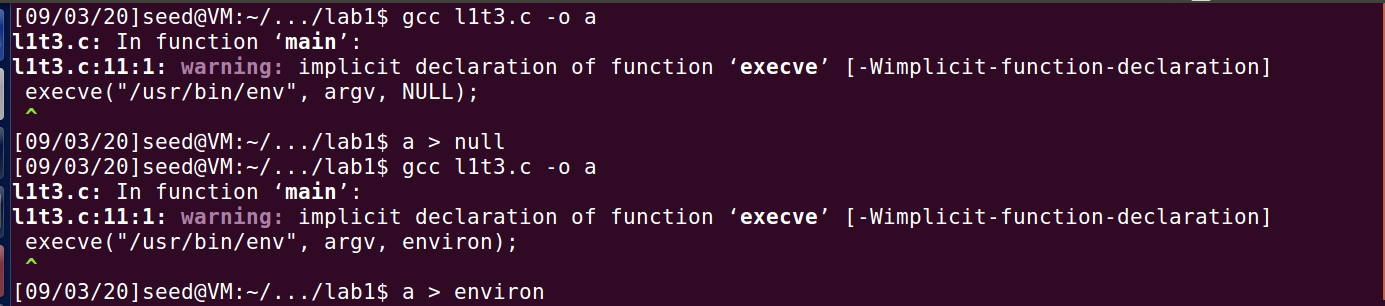
对比子进程和父进程的环境变量，发现两者一致：



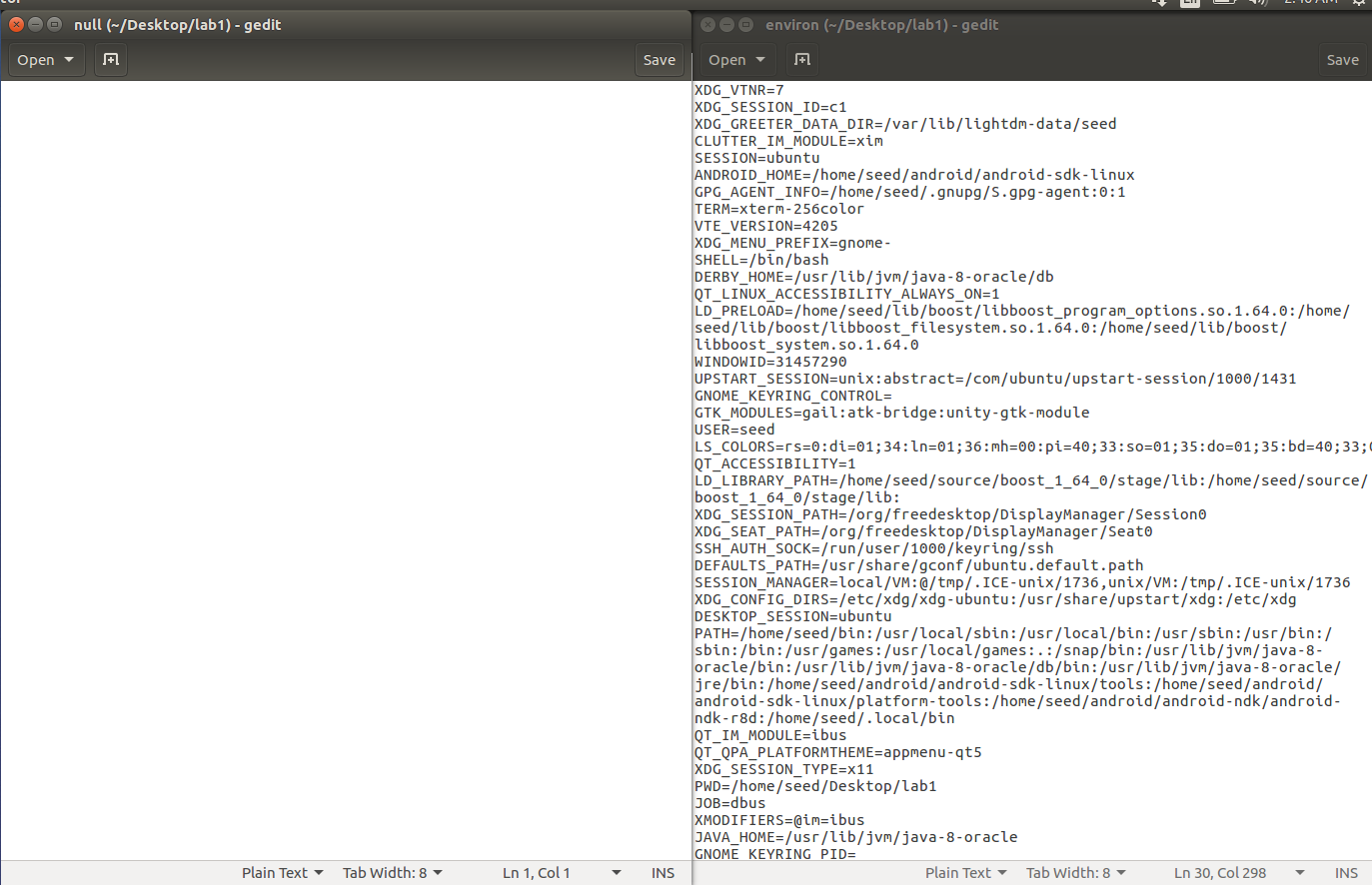
结论：子进程可以继承父进程的环境变量。

**Task3**

execve函数的执行为execve("/usr/bin/env", argv, NULL)以及执行为execve("/usr/bin/env", argv, environ)：



对比两者的输出内容：

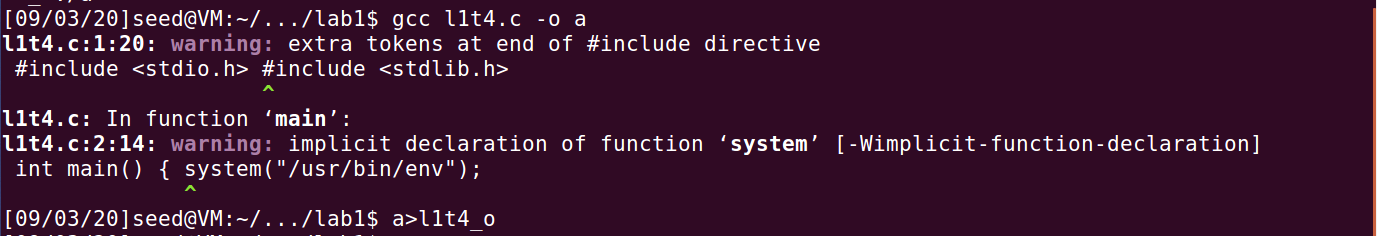


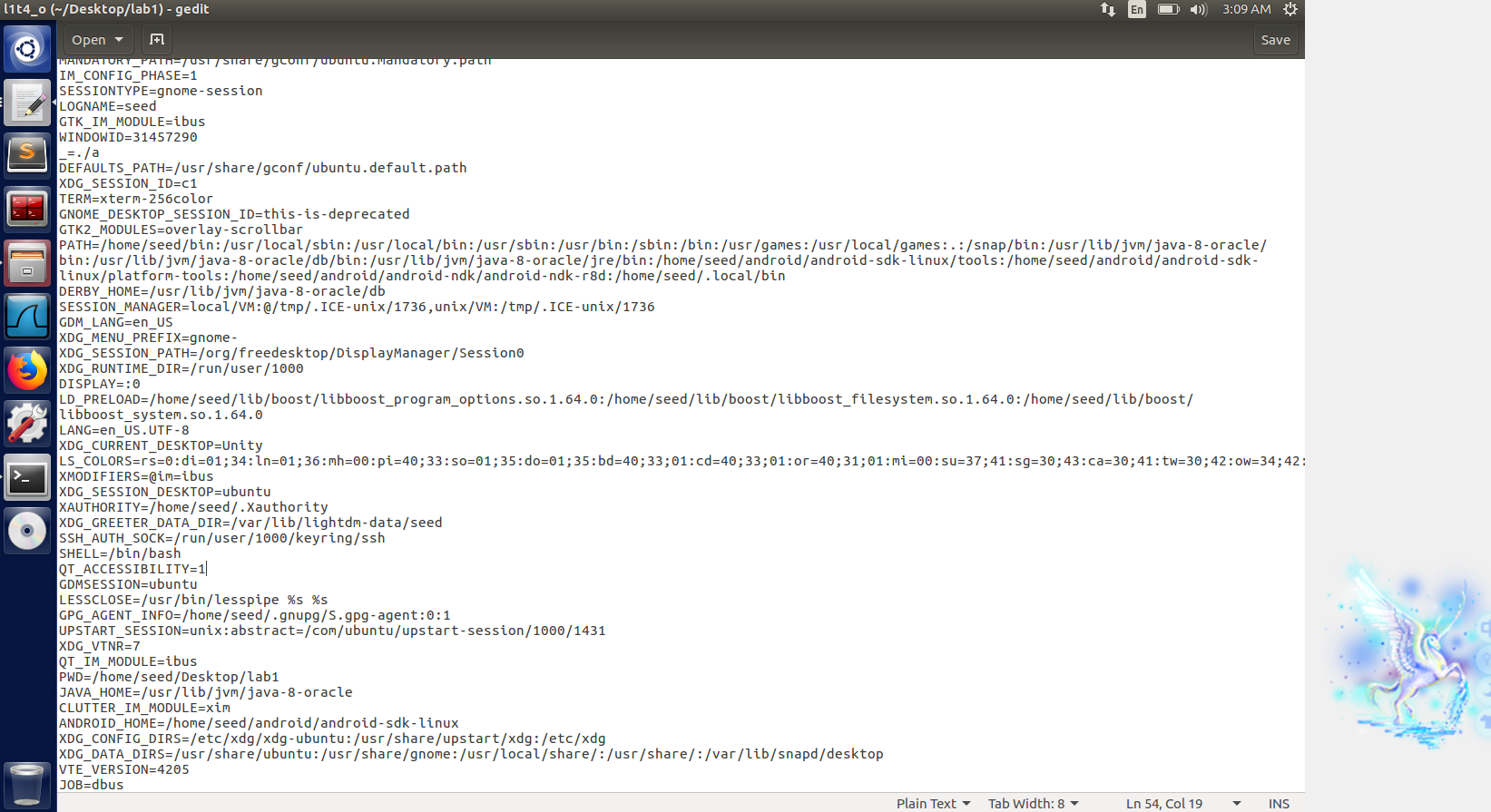
结论：execve()函数在调用过程中会执行新的程序，这时由于内存的覆盖，旧的环境变量将会丢失。可以通过设置execve()函数的第三个变量为environ，将旧的环境变量传递给新的程序。

**Task4**

system()函数是调用外部程度的一个典型例子：直接调用execl()函数，并通过execl()函数间接调用execve()函数，execve()函数运行/bin/sh。

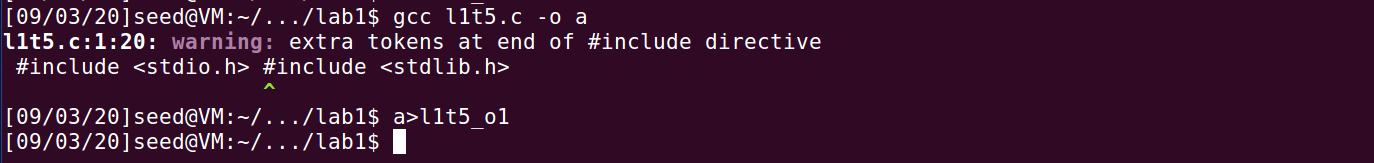
通过以下结果，可以说明使用system()函数可以传递环境变量。

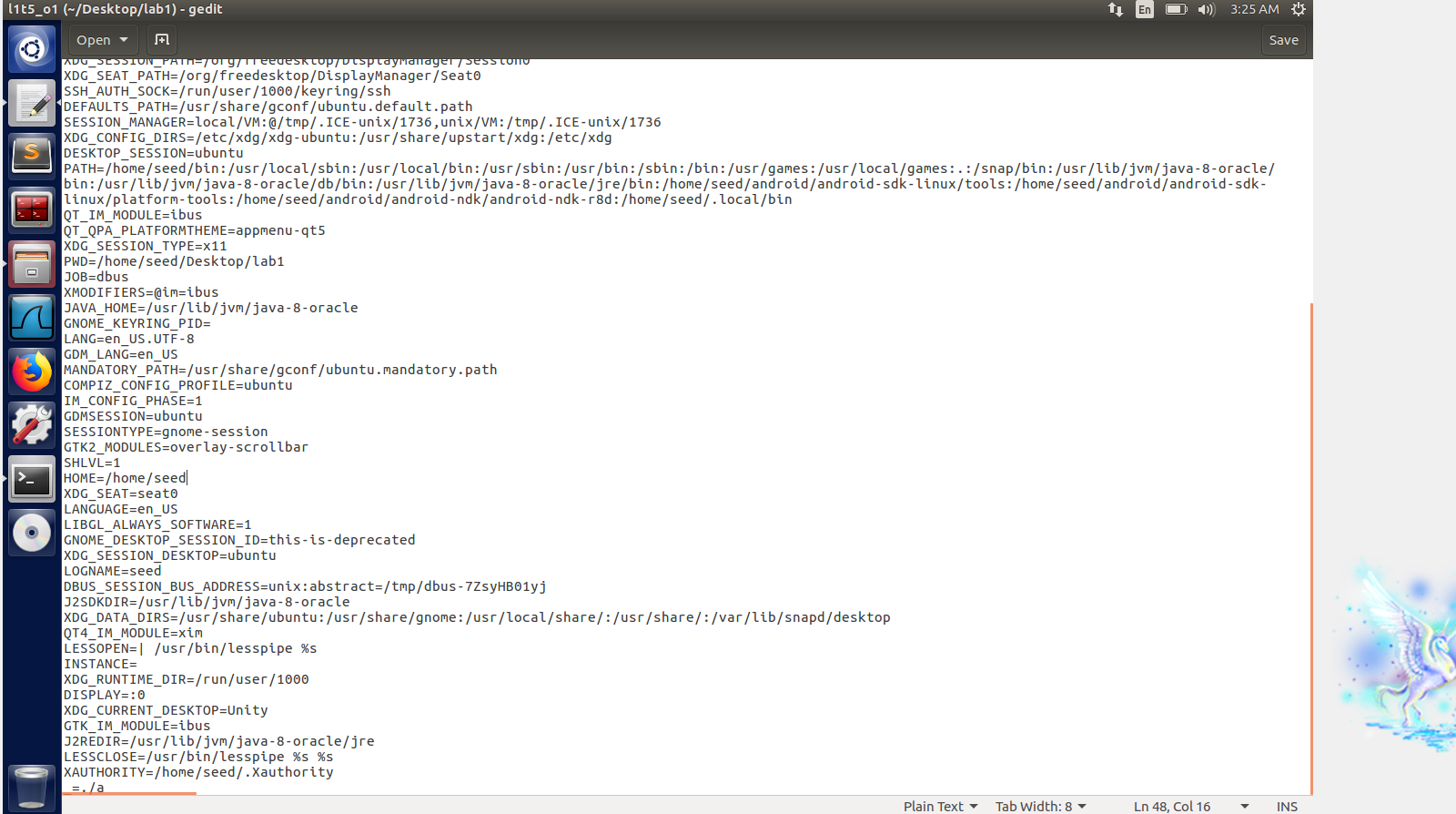




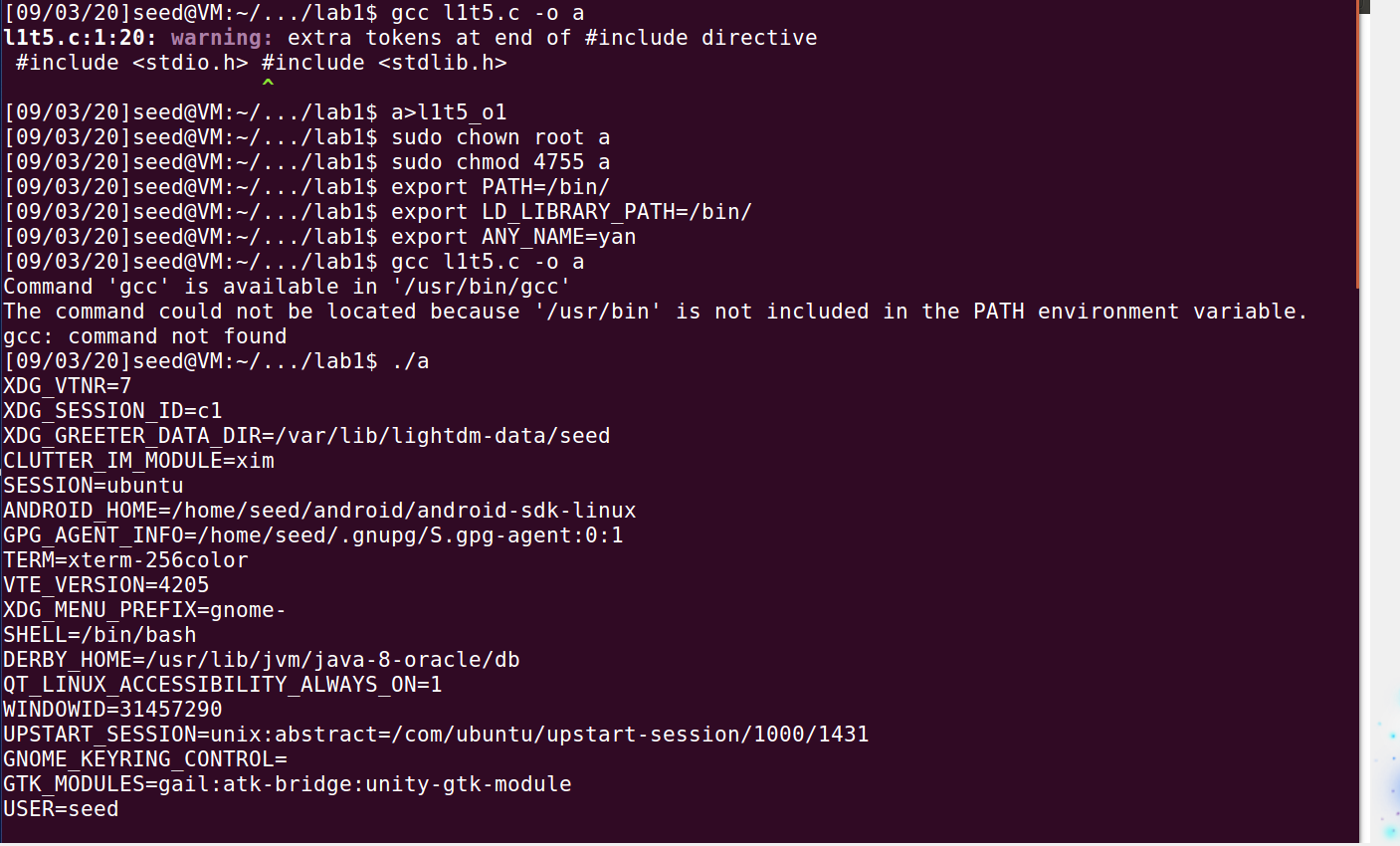
**Task5**

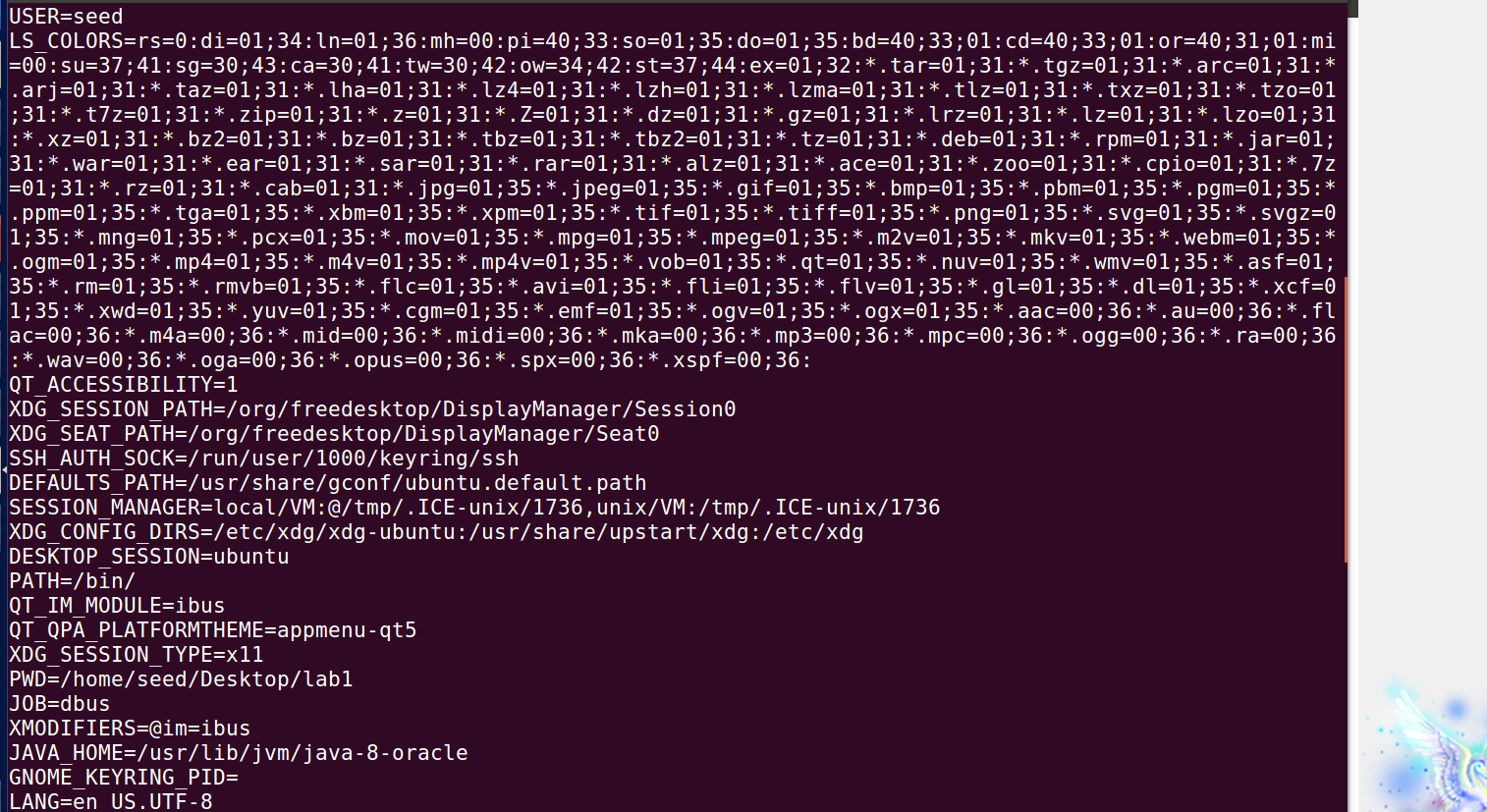
输出当前环境的所有环境变量：

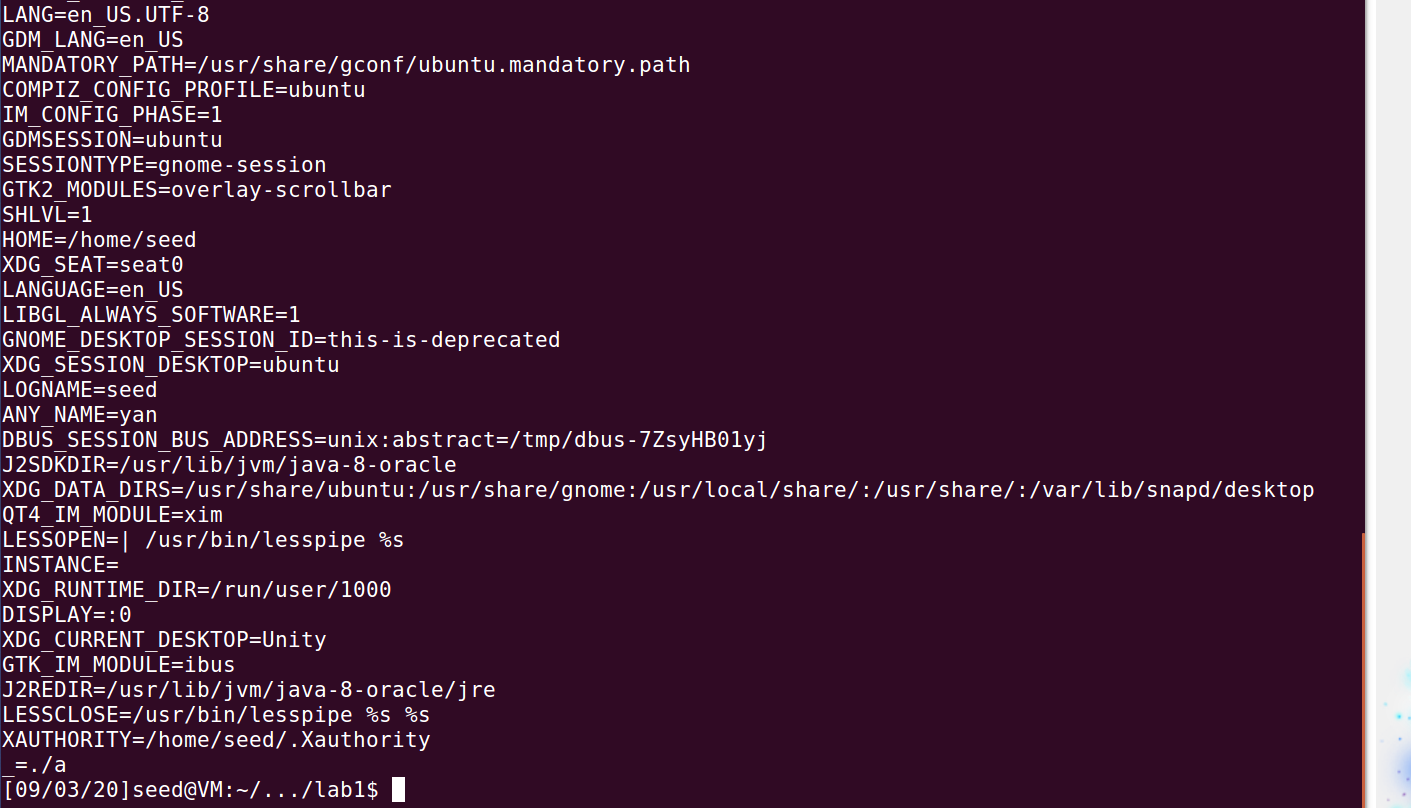




更改为root权限，使上述运行程序成为一个Set-UID程序，设置环境变量PATH、LD\_LIBRARY\_PATH 和ANY\_NAME ：





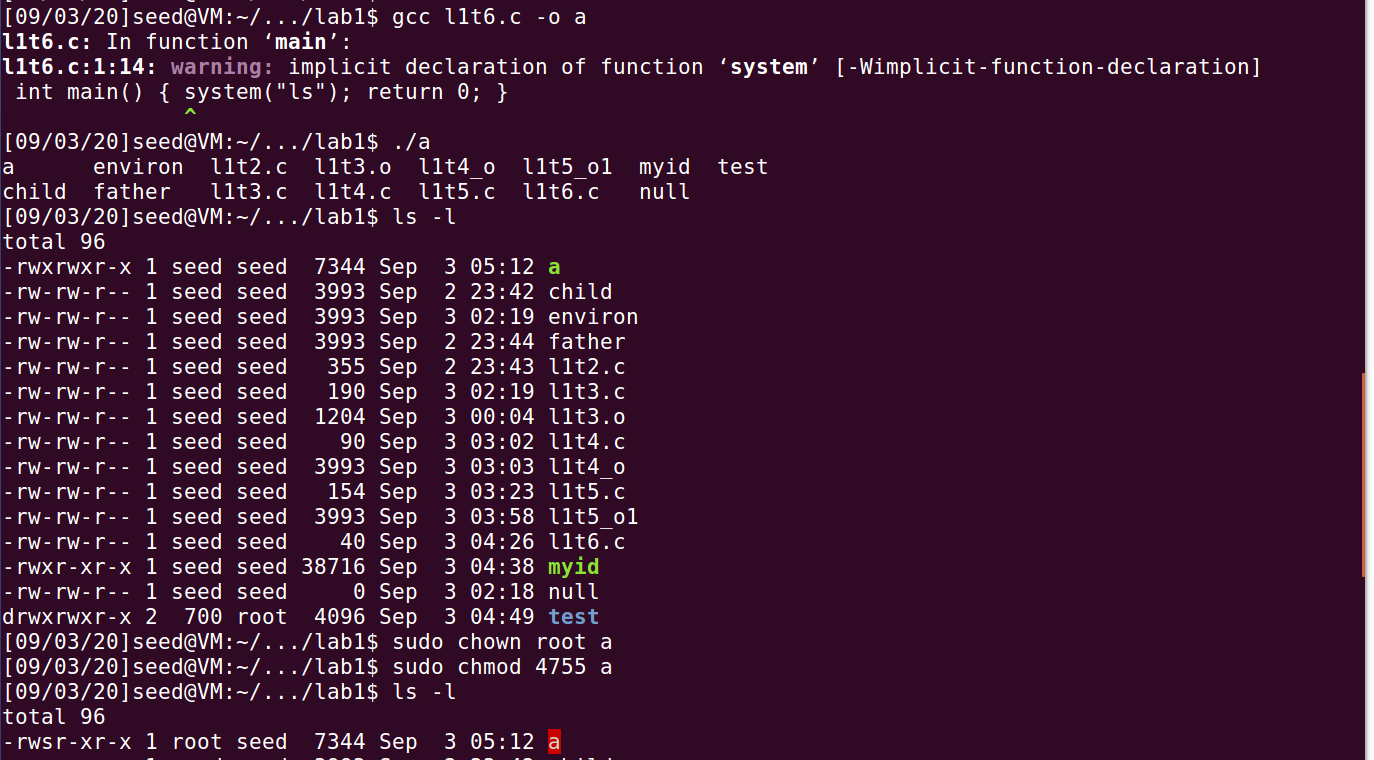


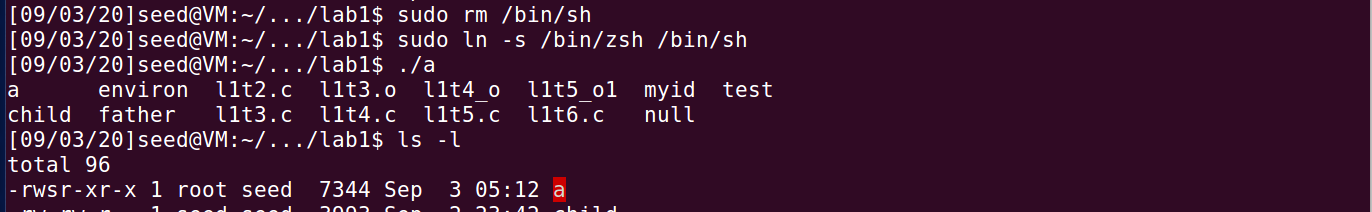
设置的环境变量没有都进入Set-UID子进程，进入子进程的环境变量为PATH和ANY\_NAME。

**Task6**

使程序成为Set-UID 程序后，运行结果不变。代码以root特权运行。

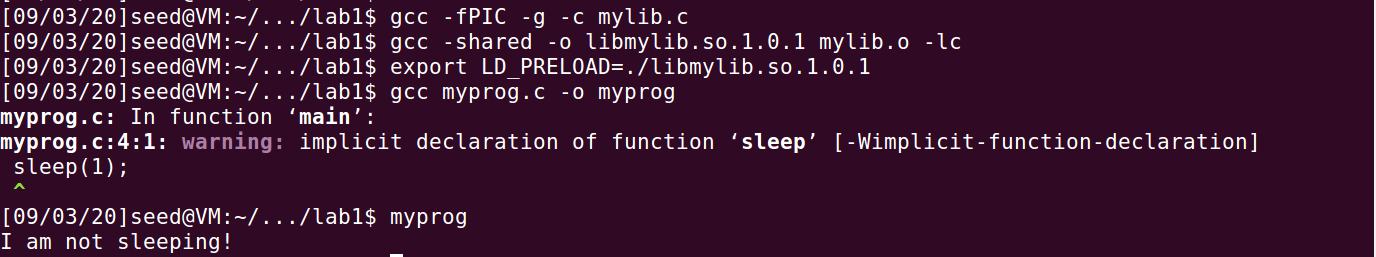
将/bin/sh链接到zsh后，运行结果不变。





**Task7**

**Step1：**

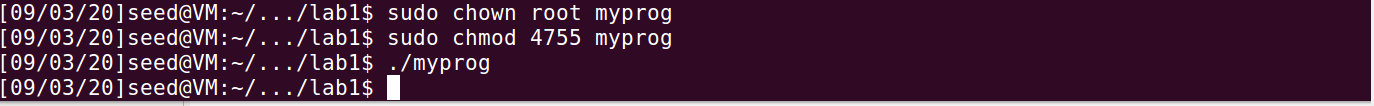


**Step2：**

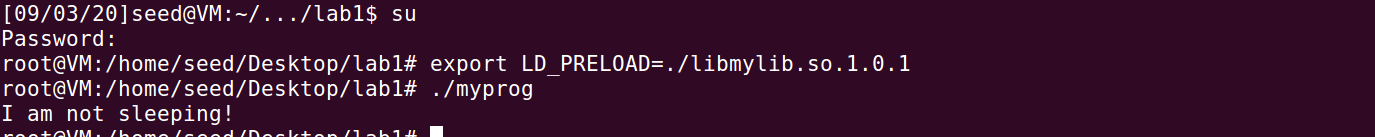
（1）Make myprog a regular program, and run it as a normal user：



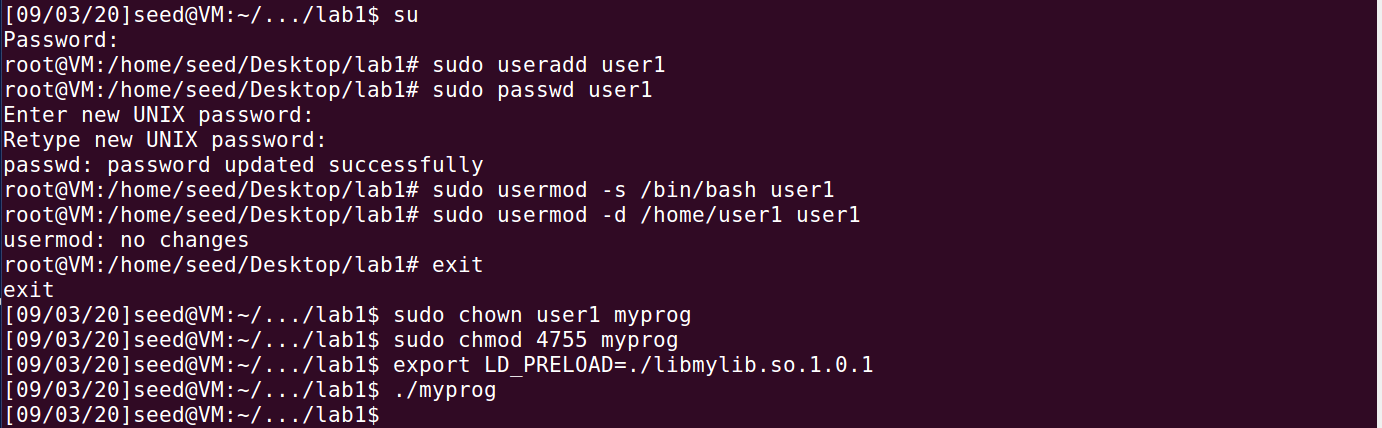
（2）Make myprog a Set-UID root program, and run it as a normal user：



（3）Make myprog a Set-UID root program, export the LD\_PRELOAD environment variable again in the root account and run it：

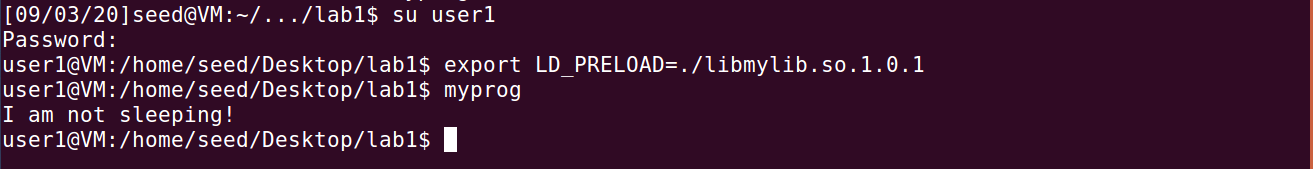


（4）Make myprog a Set-UID user1 program (i.e., the owner is user1, which is another user account), export the LD\_PRELOAD environment variable again in a different user’s account (not-root user) and run it：



**Step3：**

根据上述四种情况，推测只有当ruid与suid相等时，才可以继承LD\_PRELOAD变量，输出结果。简易验证如下：

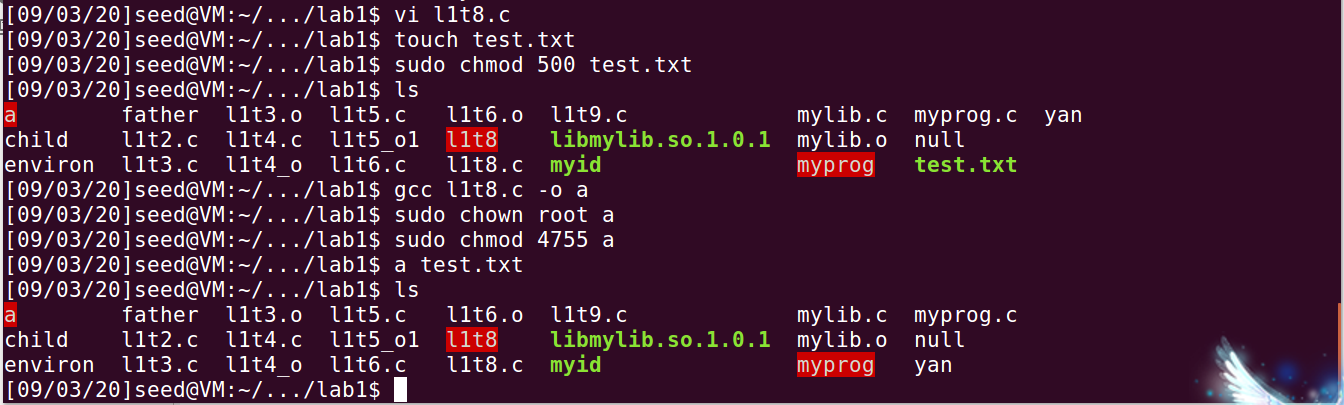


在（4）的基础上，改为在user1中运行，输出了结果。

**Task8**

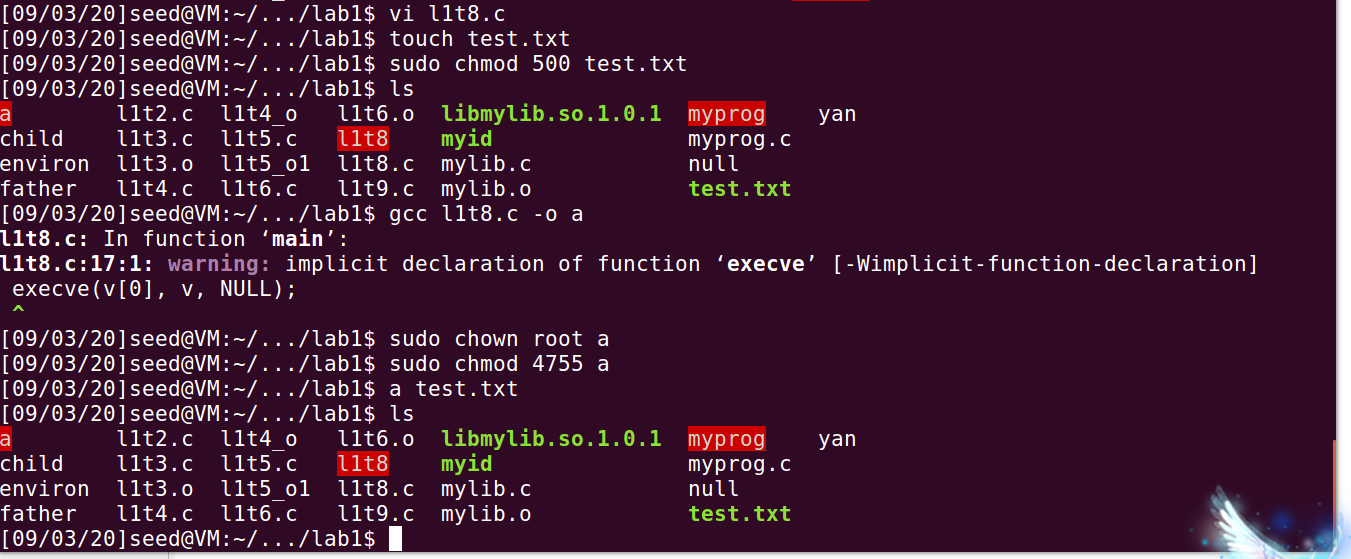
如下图所示，test.txt为不可写文件，采用system()函数时，Bob方可以删除该文件：





采用execve()函数时，Bob方不可以删除该文件：





execve()不调用shell，不受环境变量的影响，因此在Set-UID中更安全。

**Task9**

如图所示，/etc/zzz文件会被修改：



在执行fork()函数或进行输出操作之前，fd并未及时关闭。因此，特权依然被fd继承了下来，能够对/etc/zzz进行修改。

