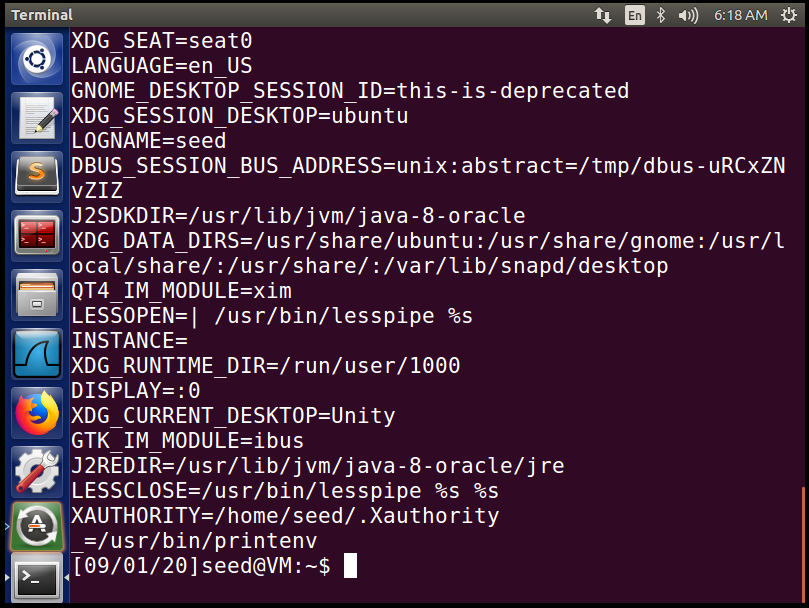
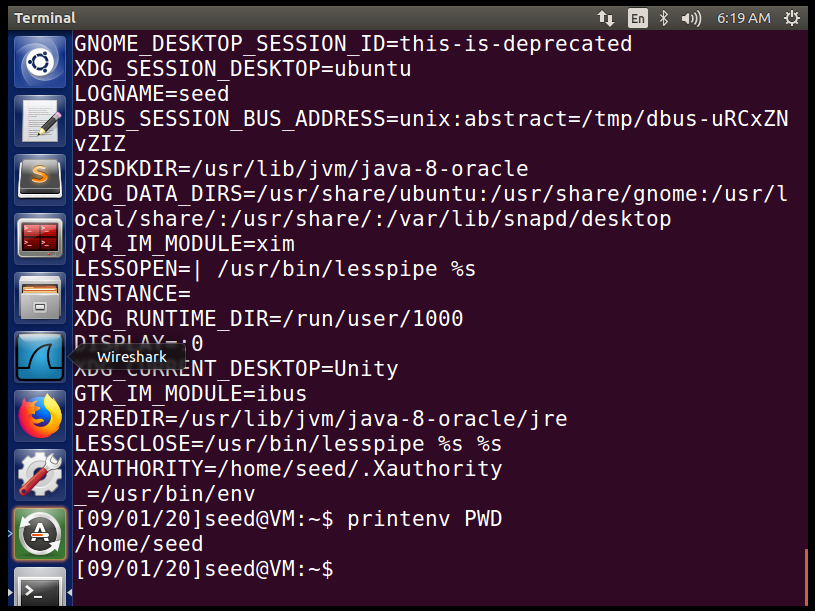
**57117226 华旻旭**

**TASK1**

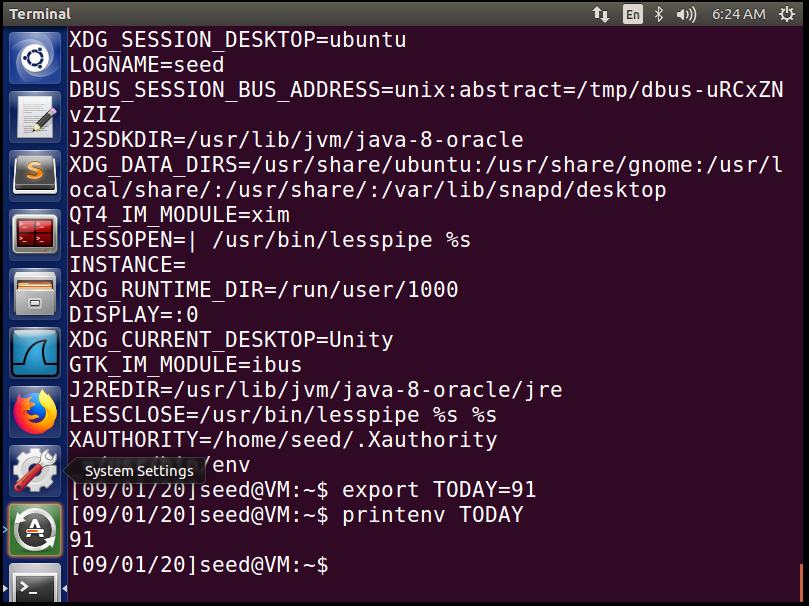
**使用printenv命令查看环境变量**



**查看PWD的环境变量**

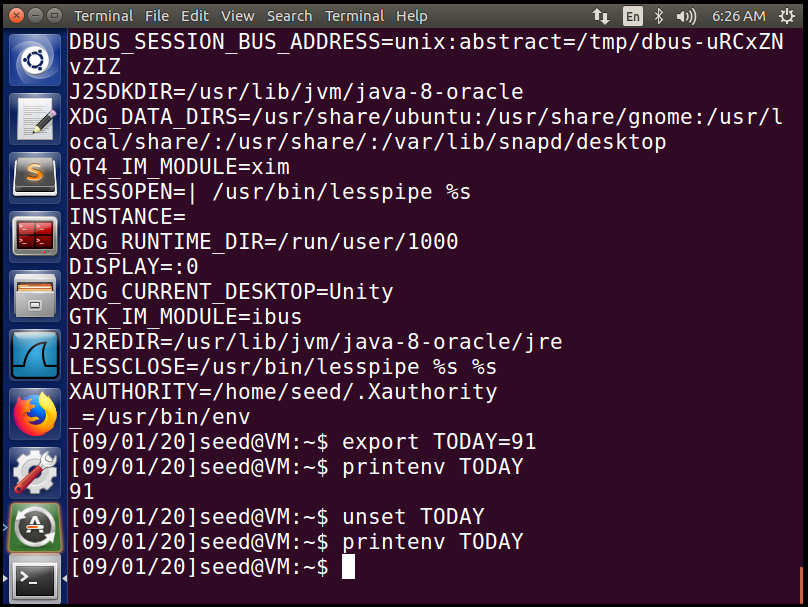


**使用export命令设置环境变量**

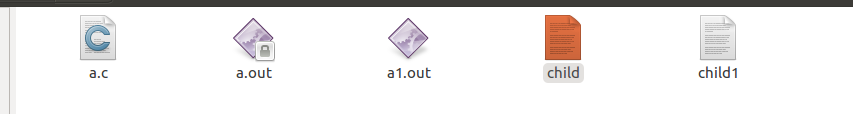


设置了TODAY变量的值为91，并打印

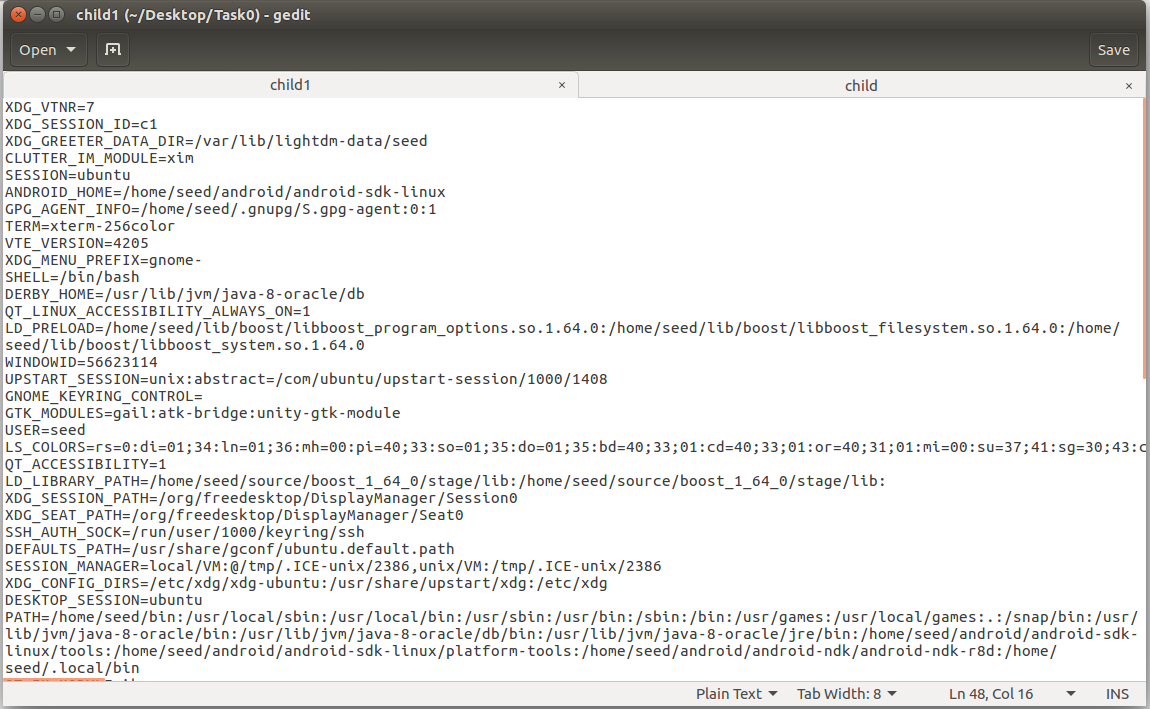
**取消TODAY变量的设置，并打印，发现已经没有了TODAY变量**

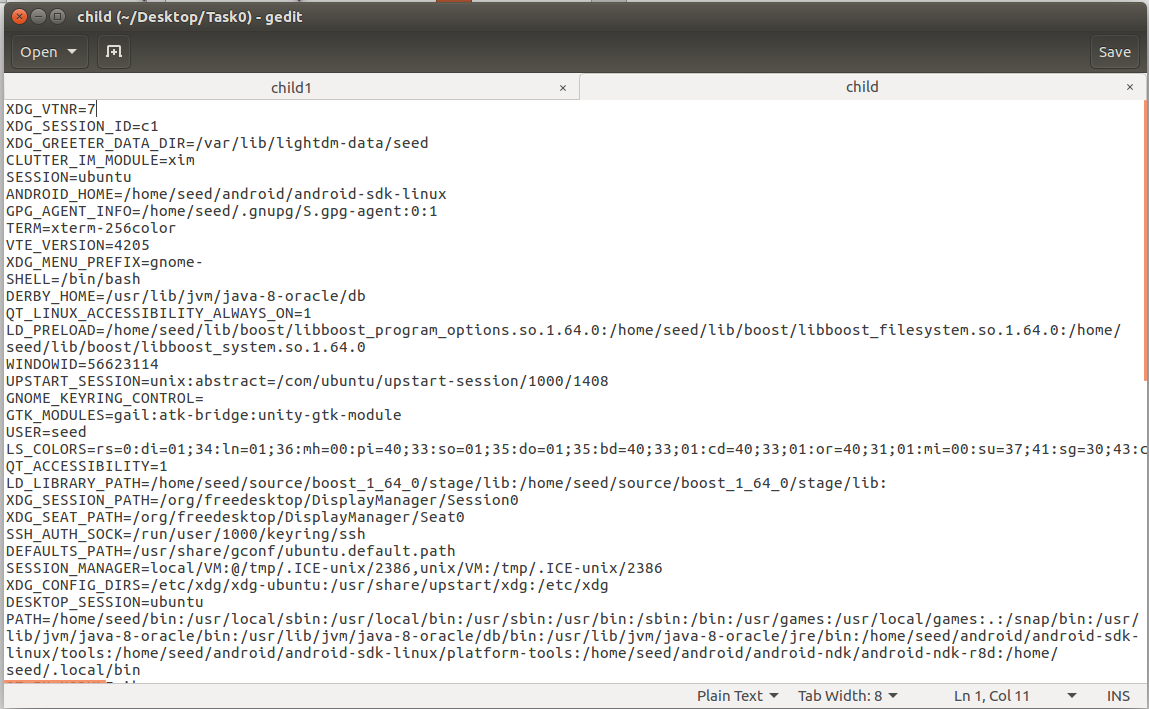


**TASK2**



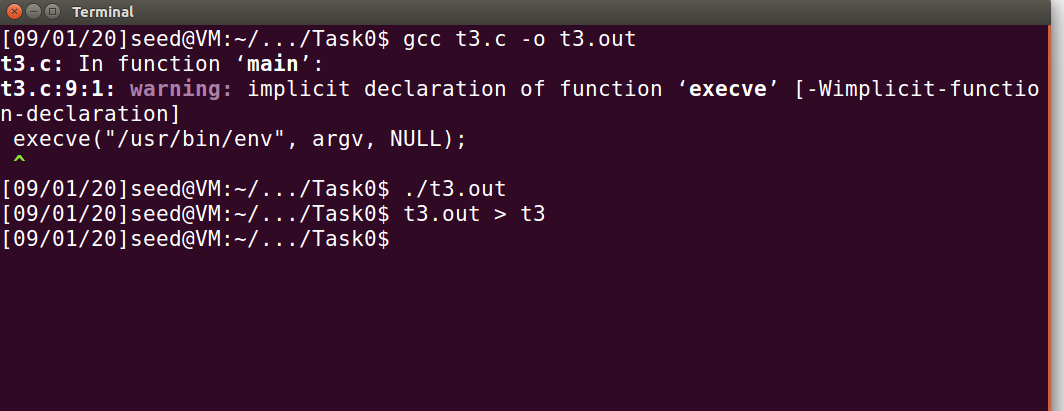
编译手册中的代码，得到注释前的执行文件a.out与注释后的执行文件a1.out，以及两次得到的输出结果child与child1.



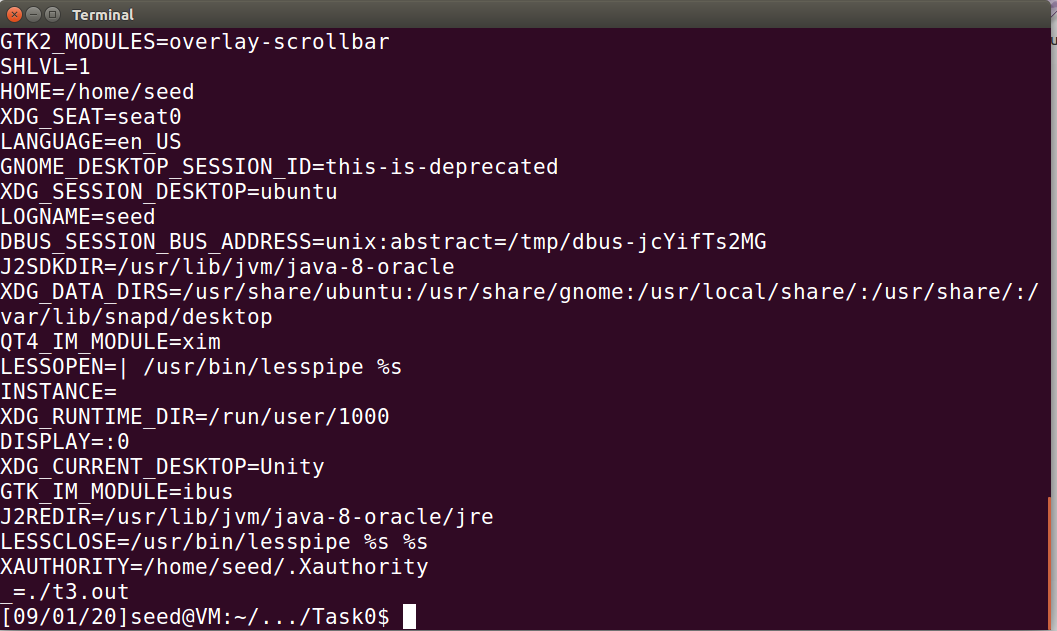


对比两次输出，发现结果一致，证明子进程与父进程环境变量一致。

**TASK3**



编译并执行t3，发现没有输出

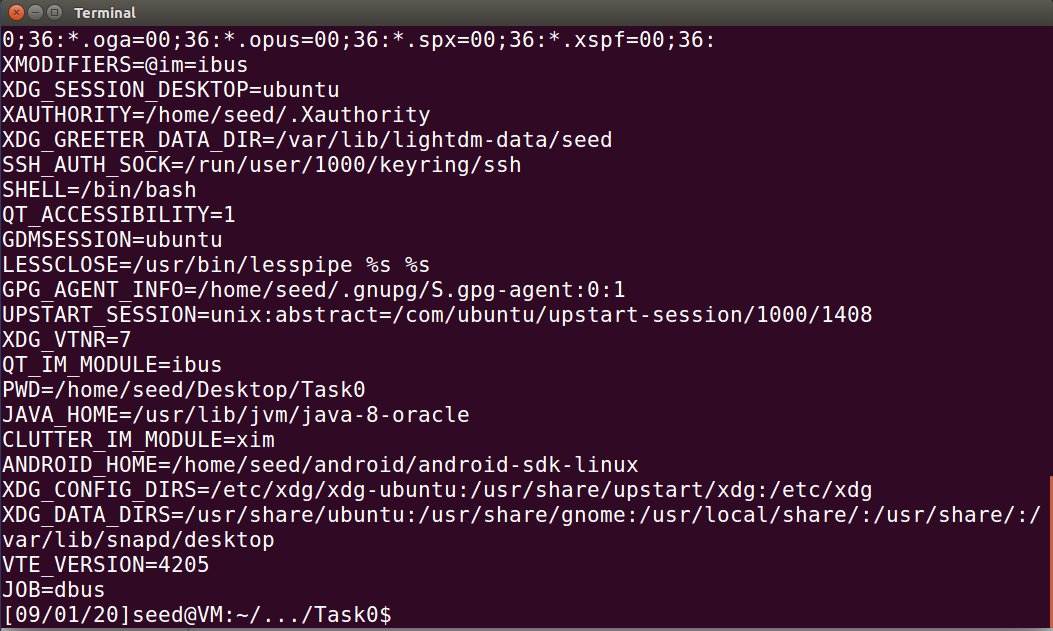


修改源代码之后，发现输出了系统的环境变量

结论：当环境变量并不在执行execve时被自动继承

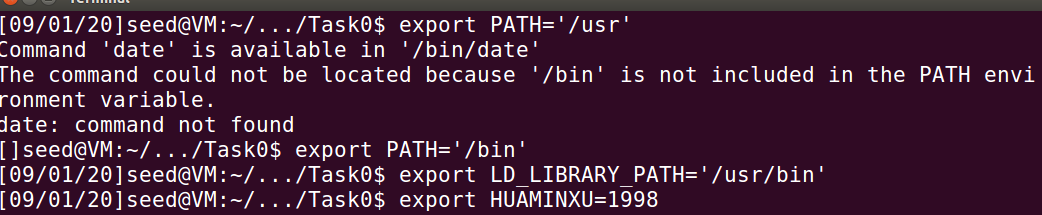
**TASK4**

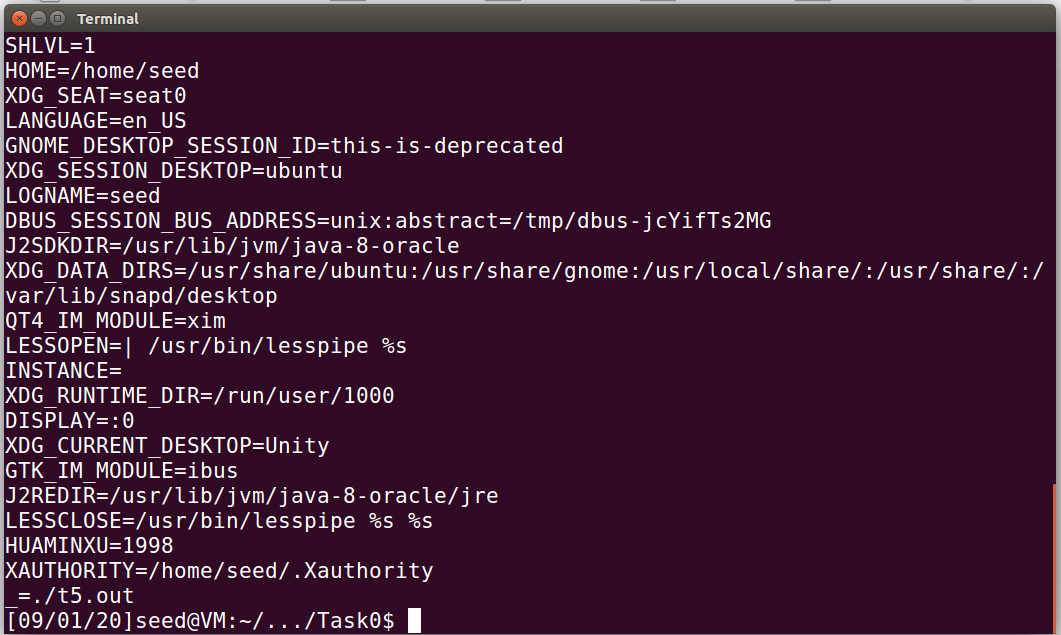
按照手册要求编译并执行，得到如下结果，可以看到



**TASK5**

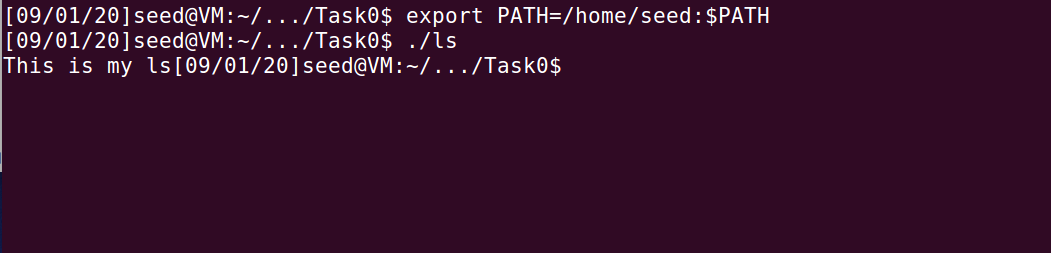
按照实验手册的要求编译程序，加入环境变量，运行发现并不是所有的环境变量都成功export了,LD\_LIBRARY\_PATH没有成功export





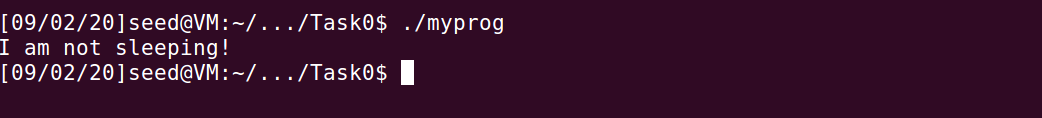
**TASK6**

将/home/seed路径添加到PATH变量的最前面，并在/home/seed中加入自己替换的ls，即可得到输出“This is my ls”

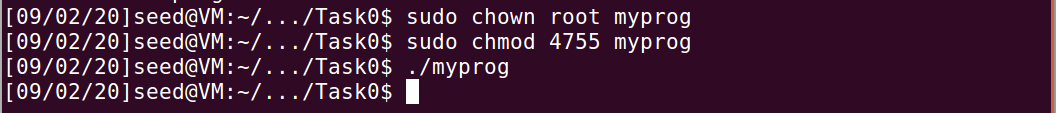


**TASK7**

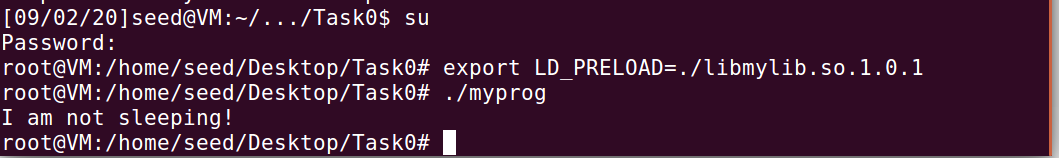
不进行任何修改运行myprog



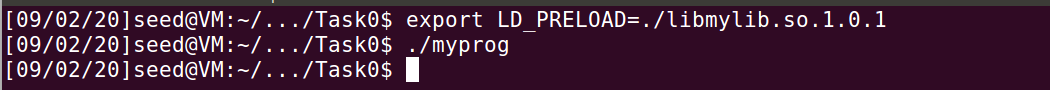
将它修改为root用户的Set-UID程序，用普通用户权限运行



在root用户下exportLD\_PRELOAD，并运行



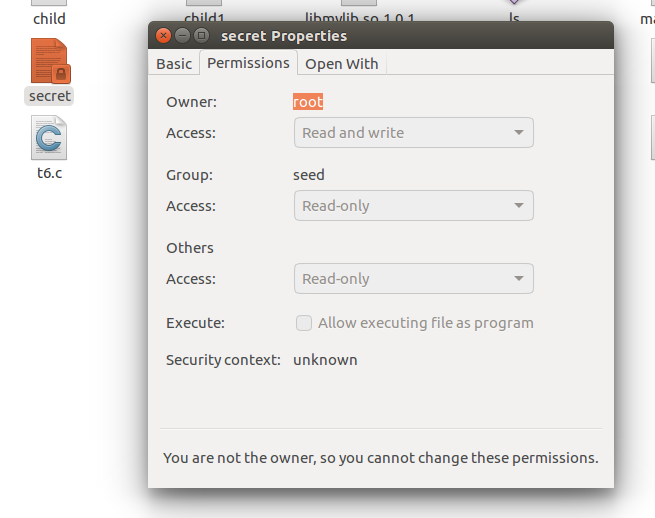
将myprog设定为user1的程序，在seed中exportLD\_PRELOAD环境变量并运行



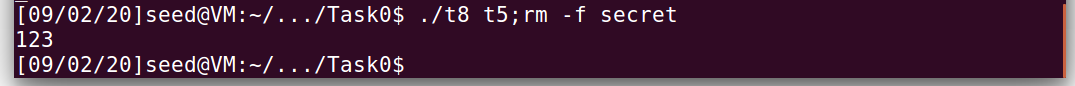
猜测是由于系统内建的某种保护机制对LD\_PRELOAD环境变量进行了保护

**TASK8**

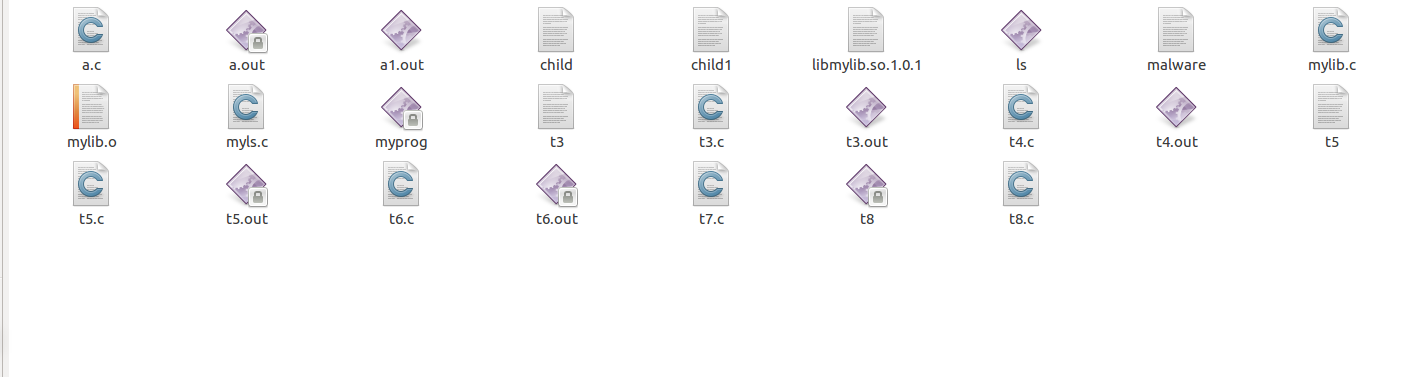
我创建了一个名为secret的文件，并将其权限改为644



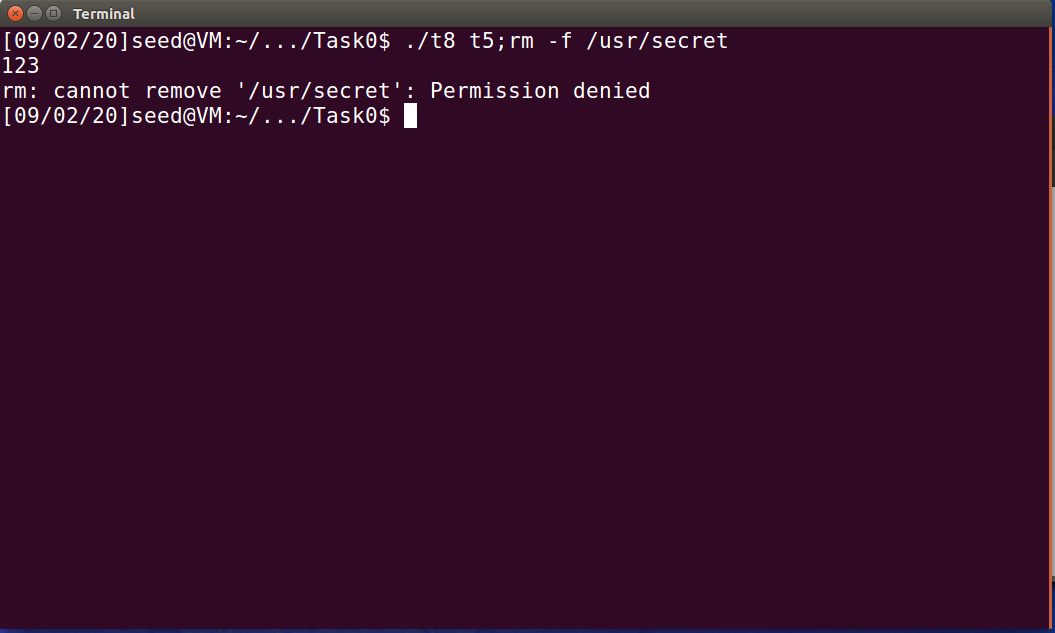
在向文件注入时，我们使用;rm -f 命令，删除系统中seed用户组只有只读权限的文件



最后发现文件被成功删除



而将system改为execve后，发现删除失败，没有权限



**TASK9**

将源代码编译并执行，可以看到zzz中被加入了Malicious Data ，说明普通用户得到了root

权限

