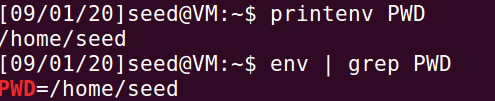
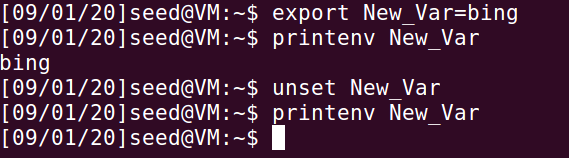
**Task1: Manipulating Environment Variables**

（1）分别使用printenv和env命令查看PWD。

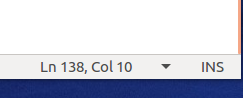


（2）使用export新建环境变量New\_Var，再使用unset删除。

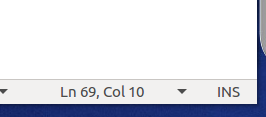


**Task2: Passing Environment Variables from Parent Process to Child Process**

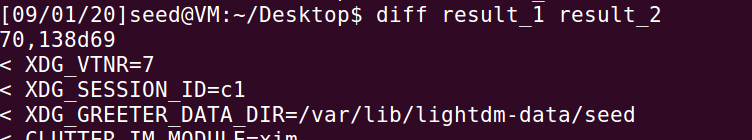
（1）运行程序，打印出138行环境变量



（2）注释掉①保留②，打印出了69行环境变量

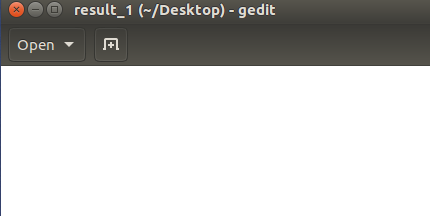


（3）使用diff命令，发现result\_2要与result\_1一样，需要删除[70,138]这个闭区间的69行内容，,说明父进程与子进程打印出的环境变量数量和内容都相同，子进程完全继承了父进程的环境变量

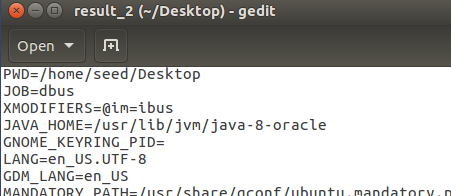


**Task3: Environment Variables and** execve()

（1）打印结果为空



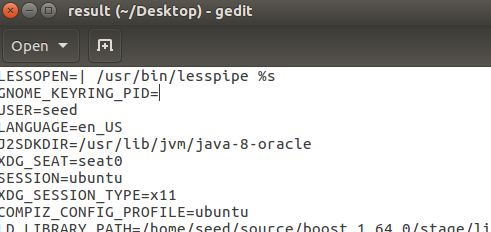
（2）打印出了当前的环境变量



（3）使用execve（）创建子进程可以通过设置参数获得环境变量

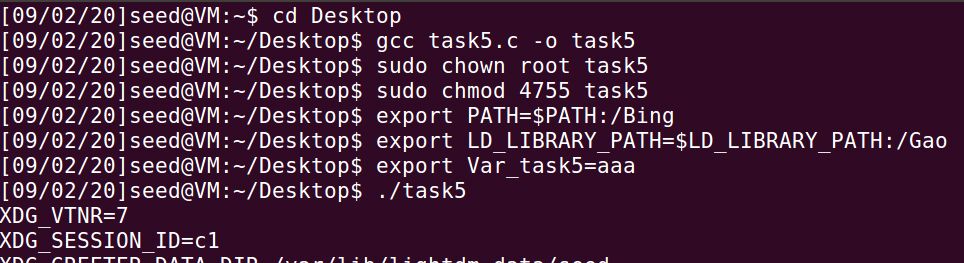
**Task4: Environment Variables and** system()

打印出了环境变量



**Task5: Environment Variable and Set-UID Programs**

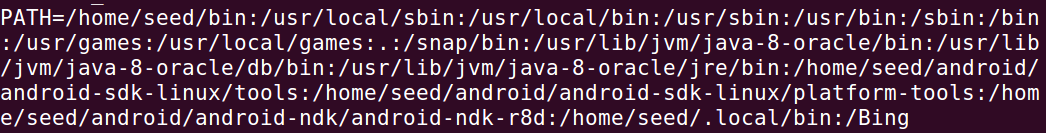
按照实验步骤，输入如图所示的命令



找到了Var\_task5



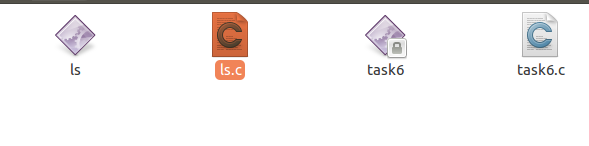
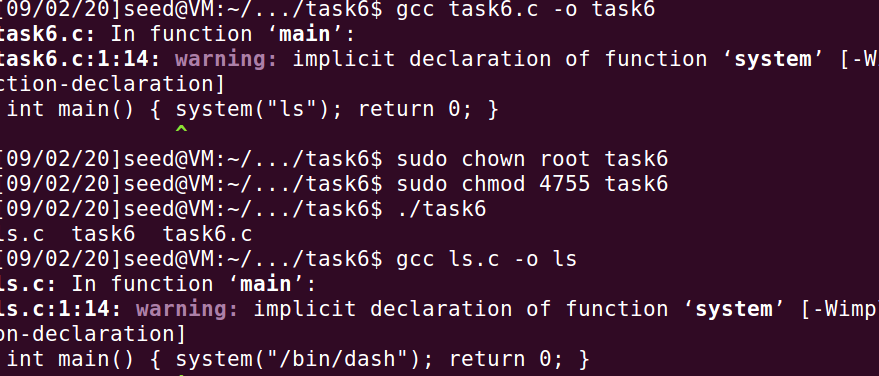
找到了新设置的PATH



没有找到LD\_LIBRARY\_PATH

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

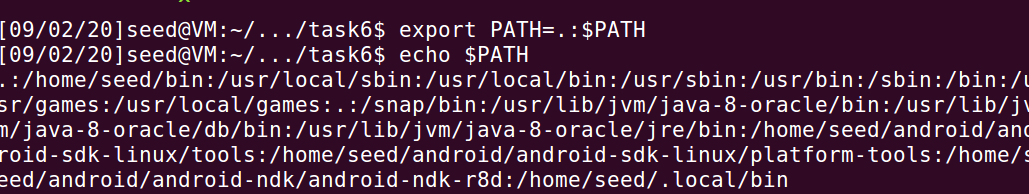
在task6.c所在文件目录下新建ls.c，并编译



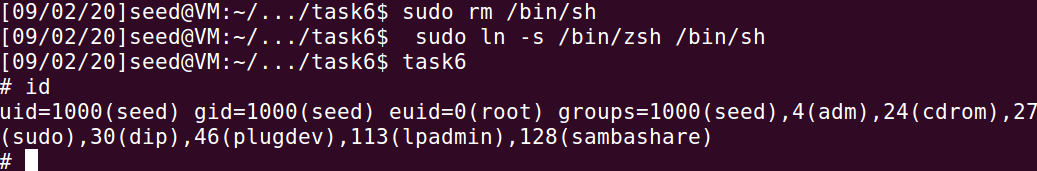
*/\*ls.c\*/*

*int main() { system("/bin/sh"); return 0; }*

更改环境变量PATH并查看是否成功



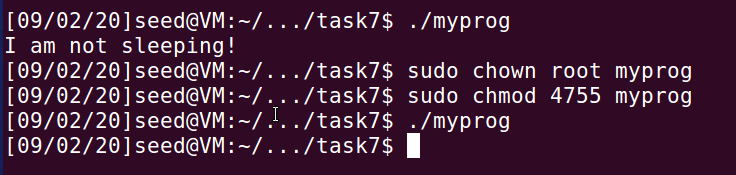
运行task6，获得了root权限



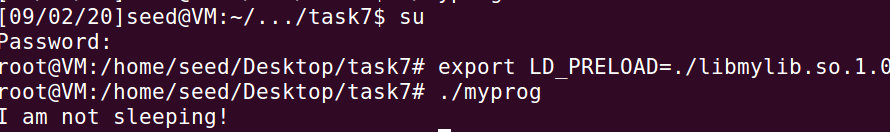
**Task7: The** LD\_PRELOAD **EnvironmentVariableand** Set-UID **Programs**

（1）myprog为普通程序，普通用户运行，打印出“I am not sleeping!”

（2）myprog为set uid程序，普通用户运行，输出空



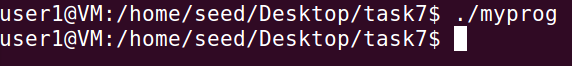
（3）设置root用户环境变量LD\_PRELOAD，在root权限下执行，打印出 “I am not sleeping!”



（4）把myprog设为seed用户的set uid程序



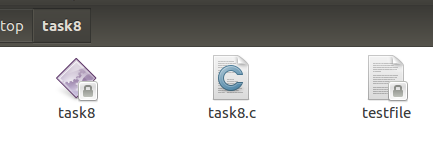
在user1用户下，先设置环境变量LD\_PRELOAD，再运行myprog，打印结果为空。



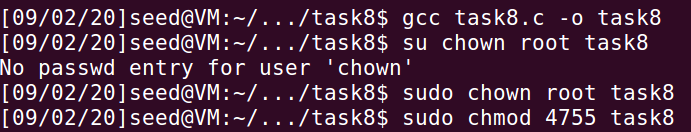
**Task8: Invoking External Programs Using** system() **versus** execve()

设置文件testfile只可读

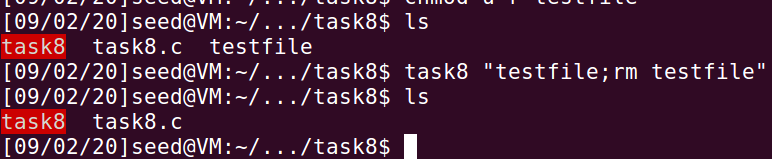




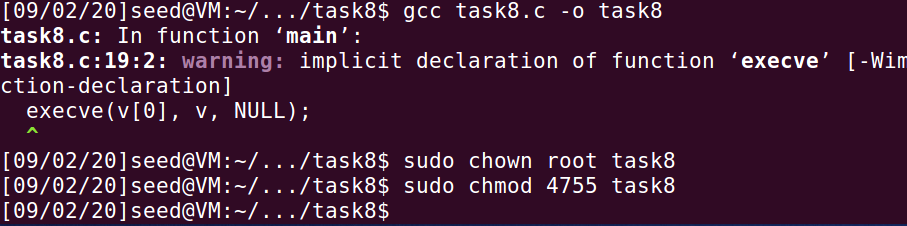
编译task8.c，并将task8改为root用户的set uid程序



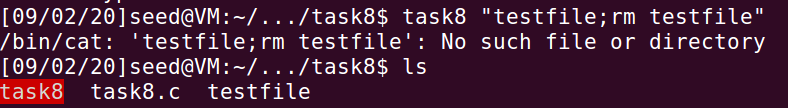
运行task8，发现删除了不可写文件testfile



注释掉system（），使用execve（），重新编译，并设为root用户的set uid程序

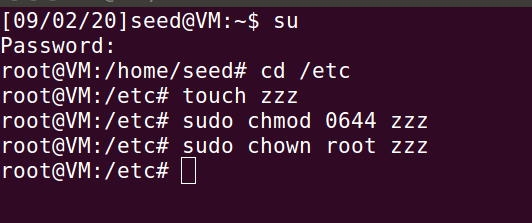


再次尝试删除文件，提示不存在改文件，删除失败



**Task9: Capability Leaking**

创建/etc/zzz，并设为root拥有，0644权限



编译task9.c并把它设为root拥有的set uid程序，执行task9，发现文件被写入了“Malicious Data”

