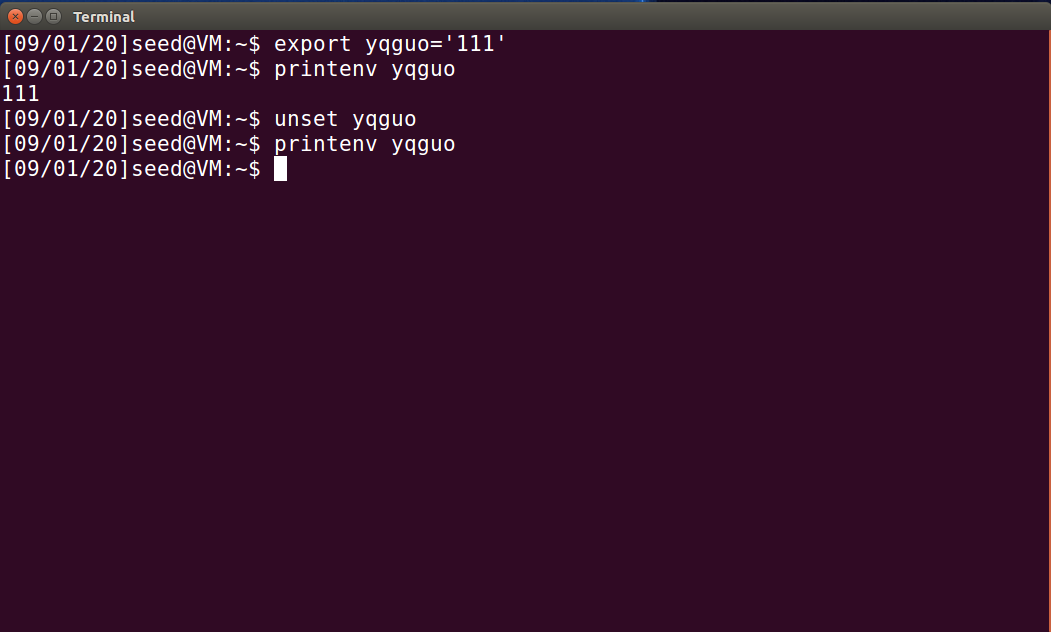
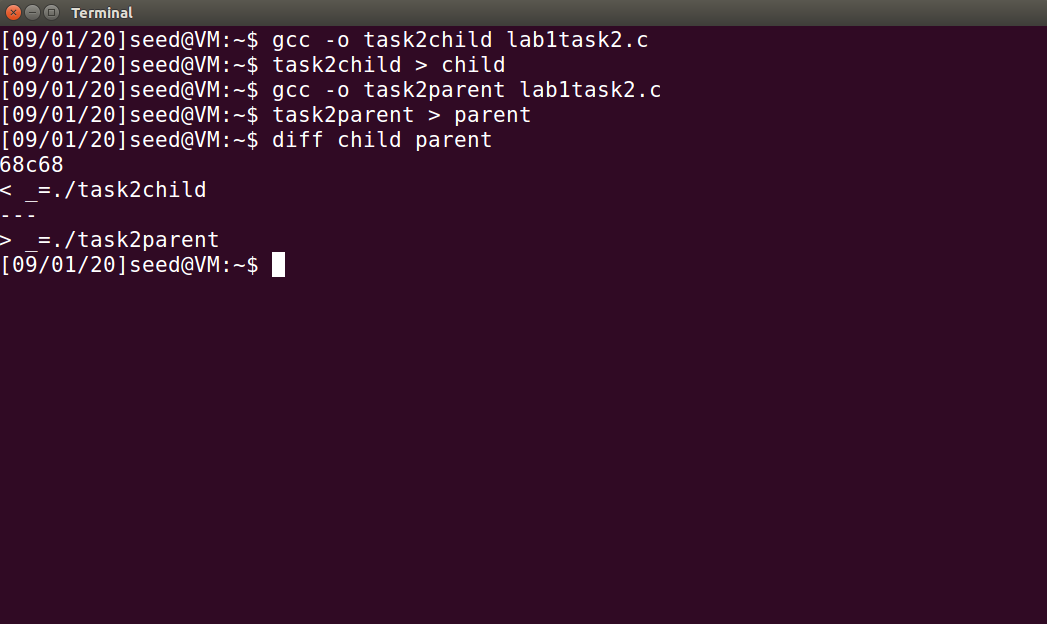
Lab1 Report

姓名：郭雅琪 学号：57118104

Task1: Manipulating Environment Variables

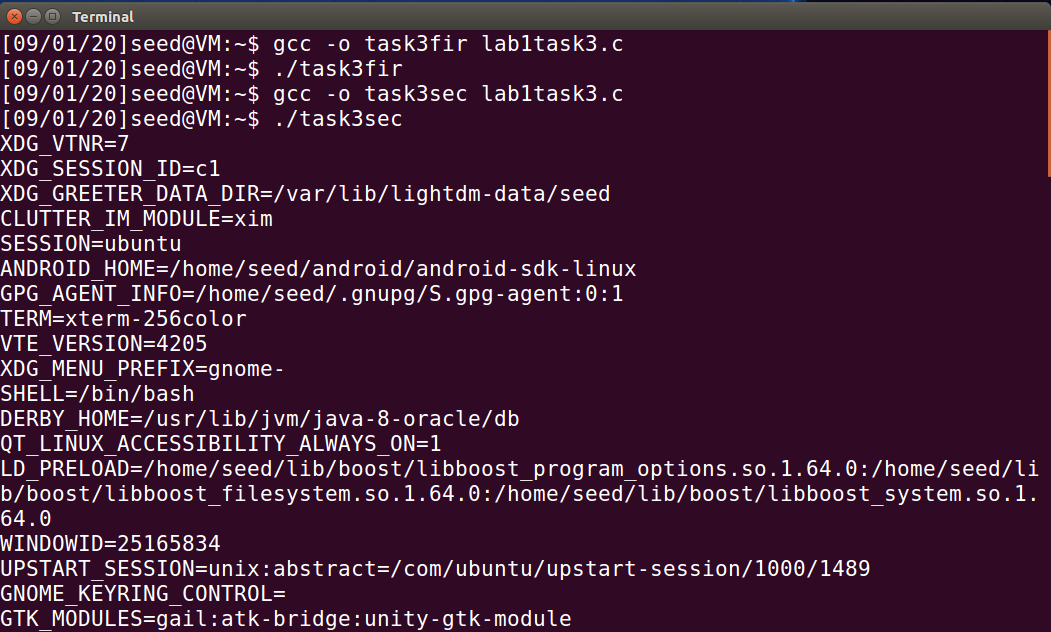


Task2: Passing Environment Variables from Parent Process to Child Process



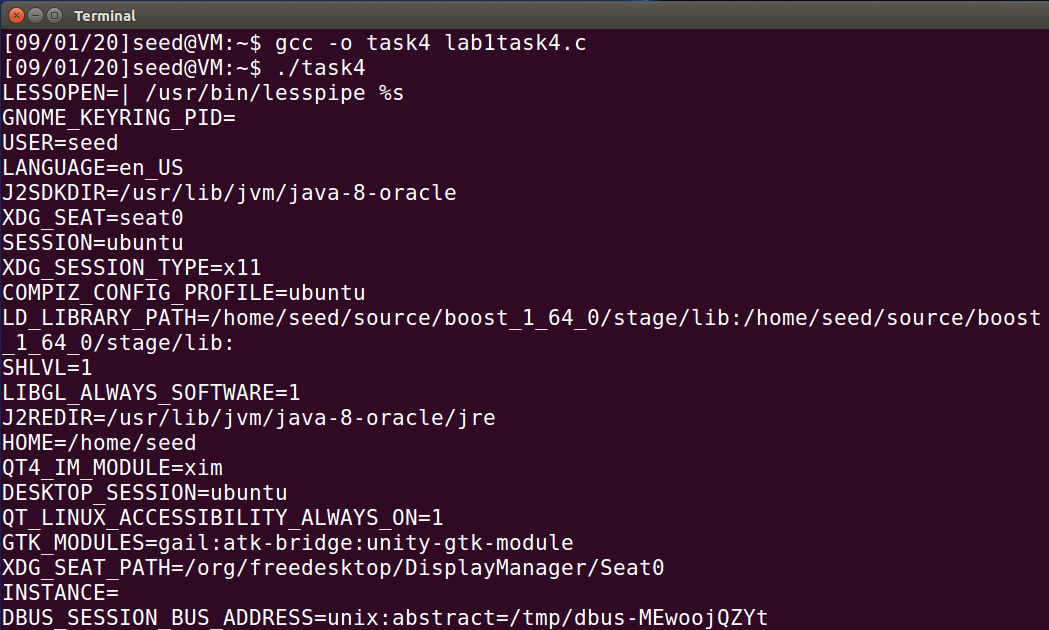
通过diff命令比较两个文件发现，子进程和父进程的环境变量完全相同。这表明使用fork函数派生出的子进程直接继承父进程的环境变量。

Task3 Environment Variables and execve()



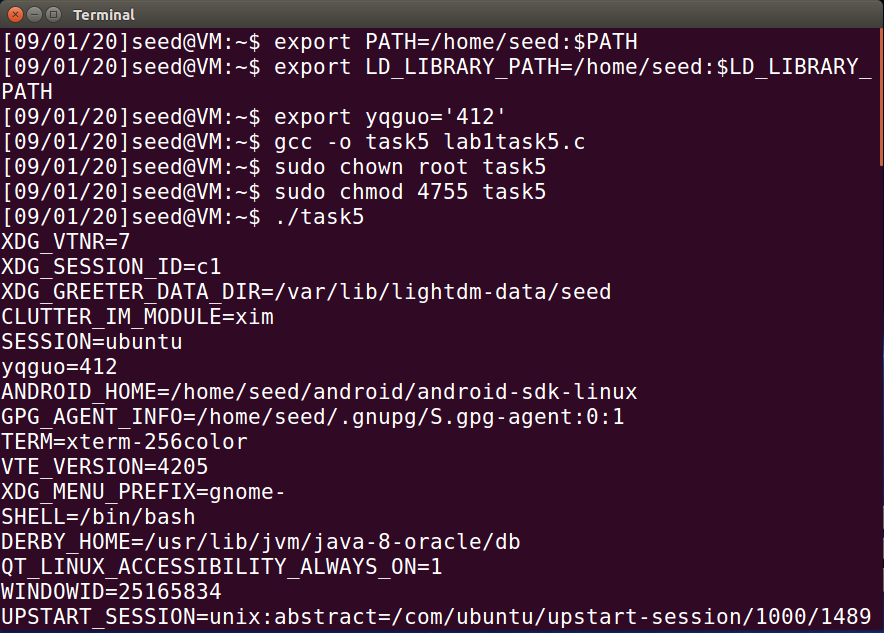
通过输出我们可以判断出，使用execve()函数派生出的子进程内存空间完全被覆盖，所有旧的环境变量都丢失。子进程调用execve()时，通过第三个参数来继承环境变量。第一次调用时，因为第三个参数为NULL，所以没有输出。第二次调用时，传入了环境变量，因此程序有输出。

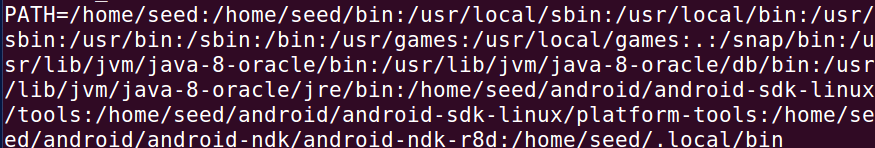
Task4: Environment Variables and system()



通过运行程序，我们发现调用system的进程的环境变量传给了新进程。

Task5: Environment Variable and Set-UID Programs

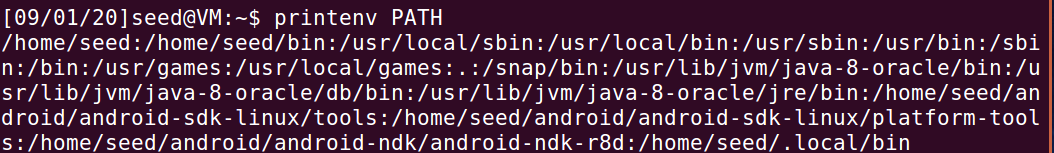




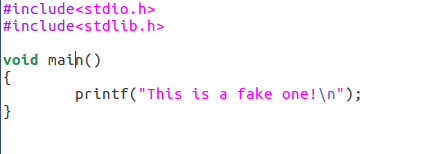
通过比对发现，在seed用户模式中修改的环境变量yqguo，PATH被Set-UID程序继承了。LD\_LIBRARY\_PATH在子进程中并没有被打印出来,说明子进程并没有继承LD\_LIBRARY\_PATH。

Task6: The PATH Environment Variables and Set-UID Programs

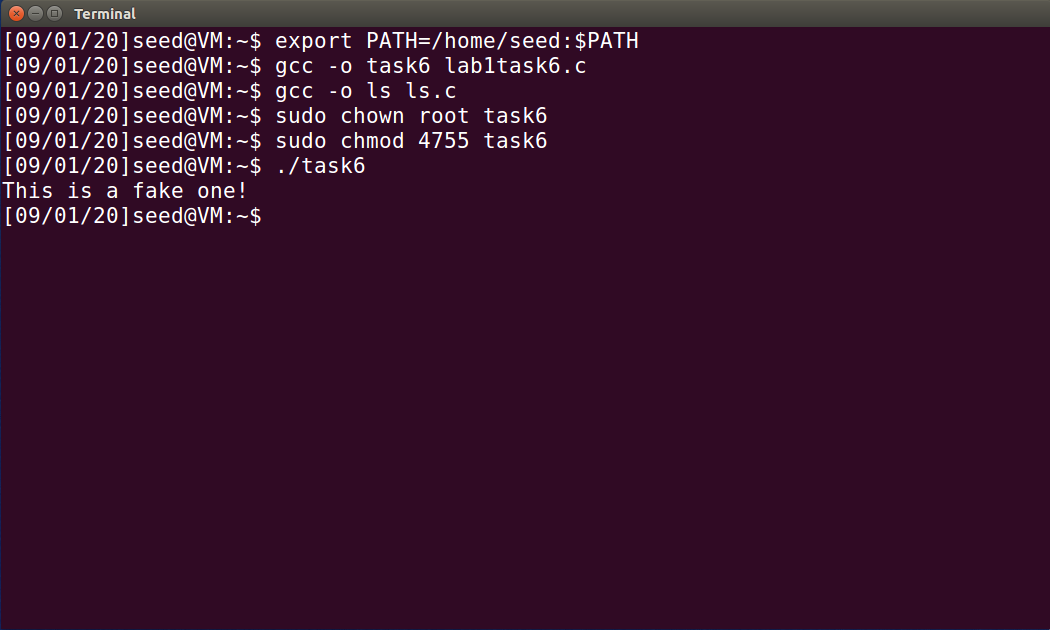
首先修改环境变量PATH，在最前面添加绝对路径/home/seed。



然后，在/home/seed目录下编写自己的ls函数并且对其进行编译，具体如下：

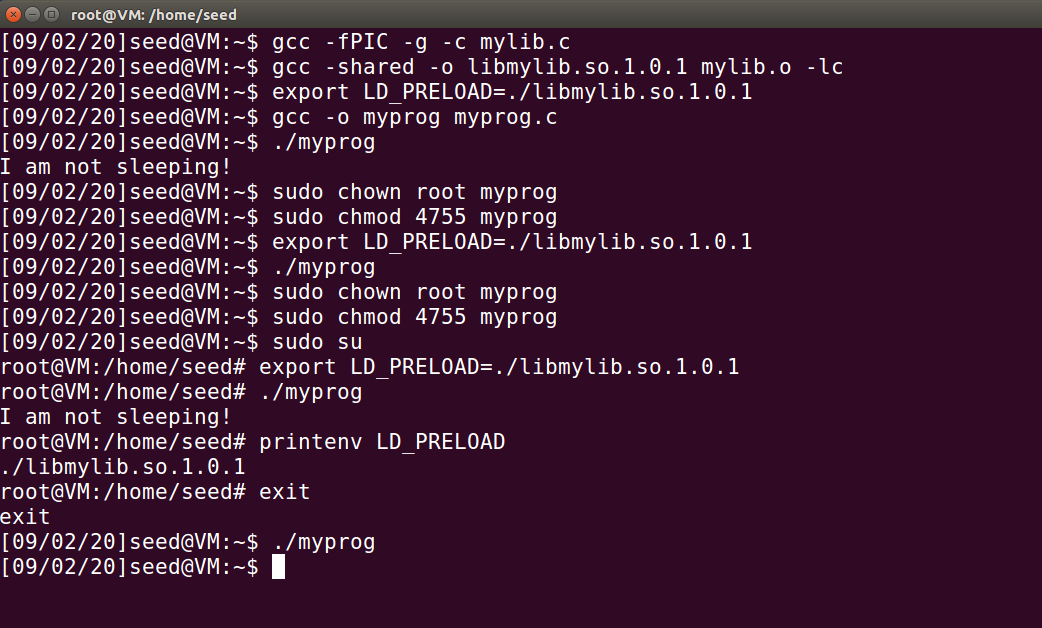


最后，把题目提供的代码进行编译并且修改其拥有者、将其改为Set-UID程序，最后运行。结果如下：

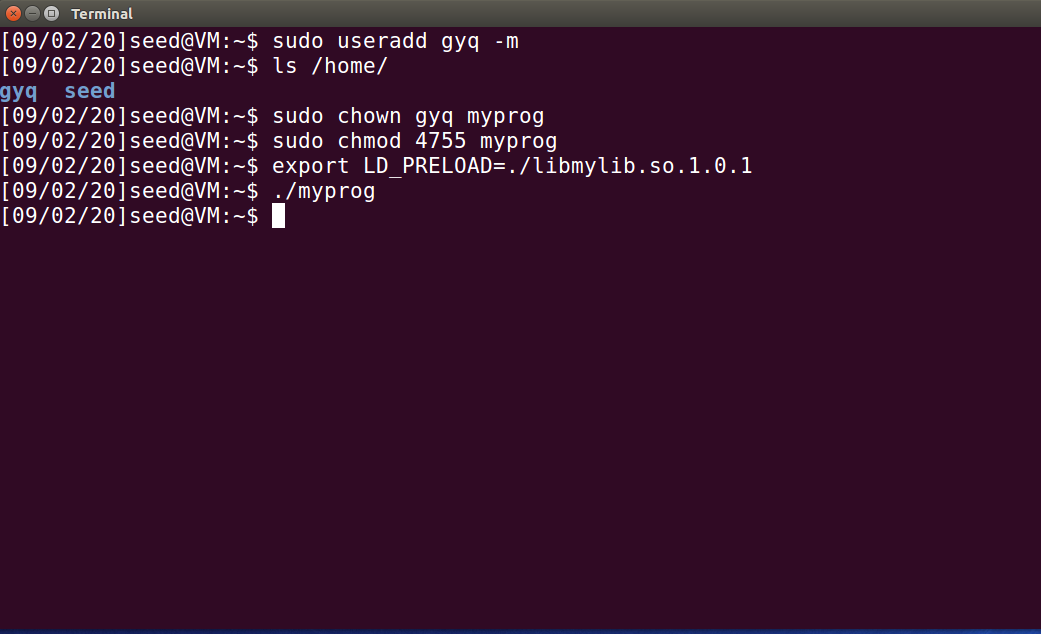


发现此时调用的ls函数已经变成了自己编写的ls函数，而非/bin/ls。原因是程序在调用ls函数时，使用的是它的相对路径，并不是绝对路径。我们在PATH环境变量中把/home/seed添加到/home/seed/bin之前，编译器在/home/seed目录下找到ls的可执行文件后就不会继续在之后的目录中寻找，因此最终调用了seed用户自己编写的ls函数。

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



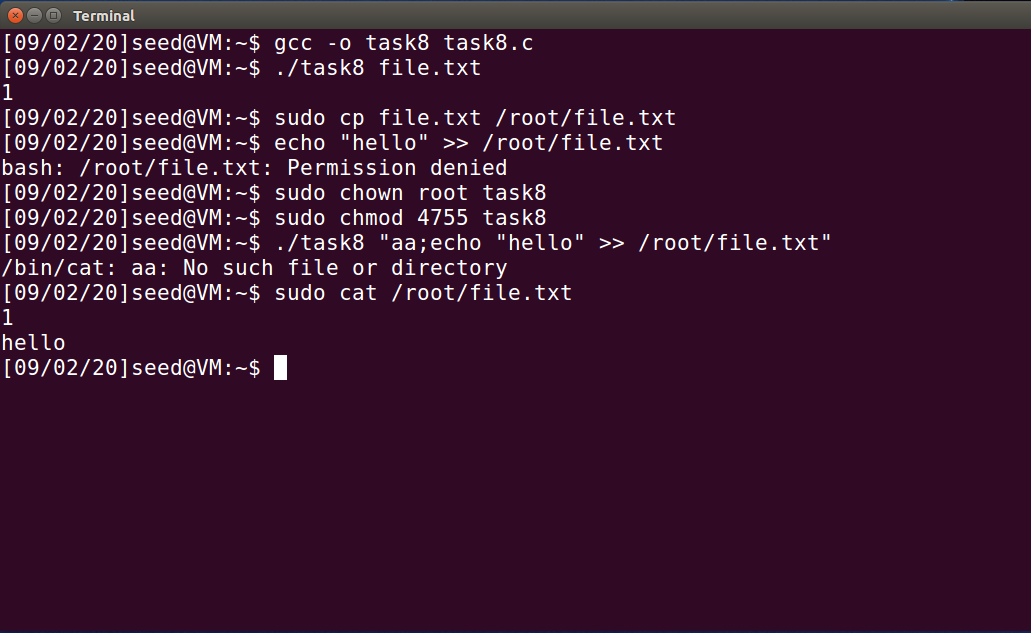
通过观察输出我们发现，当myprog是一个普通程序，并在seed用户下运行时，调用自己编写的动态链接库。当myprog成为Set-UID程序并且root用户是其所有者时，在seed用户下运行调用原有的sleep函数，在root状态下运行，调用自己编写的动态链接库。



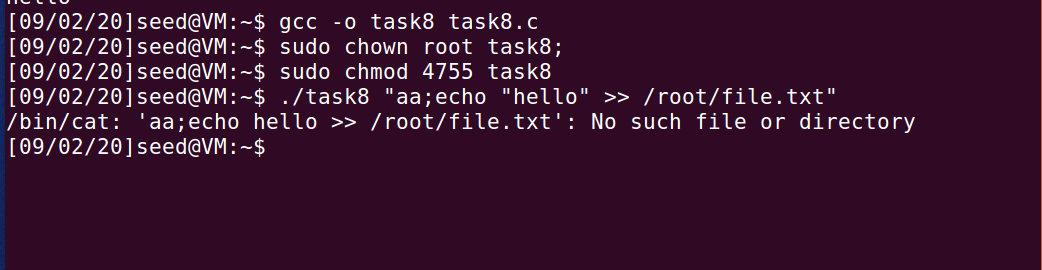
在此情况下，我们新建了一个用户gyq，将myprog设为Set-UID程序但所有者是gyq（非root用户），此时我们在seed用户下更改环境变量LD\_PRELOAD并运行myprog，发现调用的是原有的sleep函数。

通过以上4种情况我们发现，只有在myprog拥有者的状态下更改LD\_PRELOAD并且运行，才可以调用自己编写的动态链接库。

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



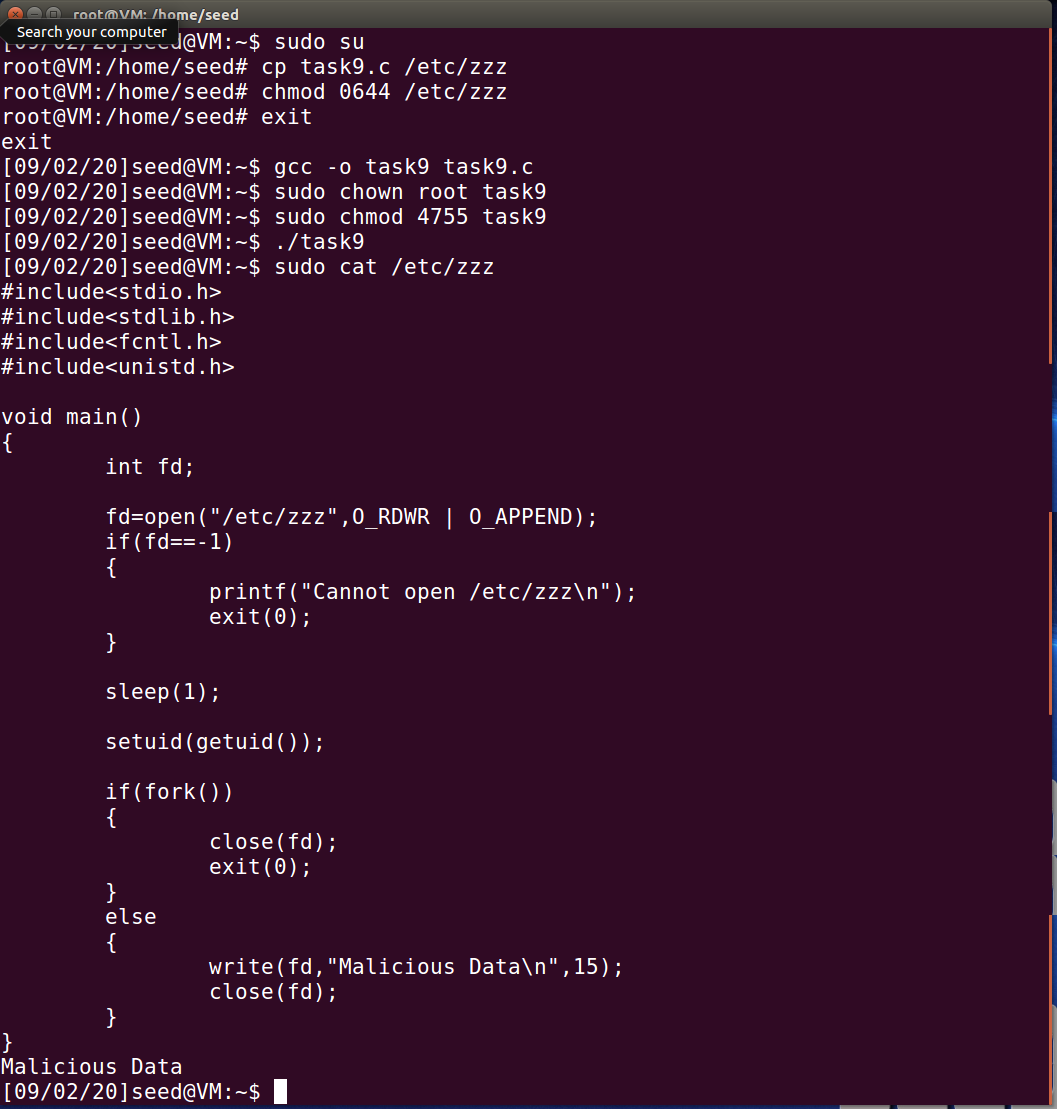
对于使用system语句执行命令的程序。首先用task8打开file.txt，查看其中内容。之后，将其复制到root文件夹下之后，发现普通用户不能向其中写入内容。在将task8的拥有者变为root且转变成Set-UID程序之后，对其进行攻击。在输入之后加入向文件中写文字的代码，成功在文件中写入内容。



而对于使用execve语句执行命令的程序，使用上述方法无法实现攻击。原因在于，system函数调用时只有一个参数，它会把参数当做命令全部直接执行。而execve函数调用时有三个参数，用户的输入为第一个参数，也是就是文件名，这样可以有效阻止上述攻击。

Task9: Capability Leaking

首先，在root权限下，将一个文件复制为/etc/zzz，并且将其权限改为0644。然后回到seed用户状态下，把task9改为Set-UID程序，并将其拥有者改为root。



通过上图我们可以看出，在程序已经将ruid和euid改为一致的情况下，进程仍旧成功修改了/etc目录下的文件，说明发生了权限泄露。原因是没有及时关闭句柄，导致之后的操作命令仍旧可以修改普通用户状态下的只读文件。