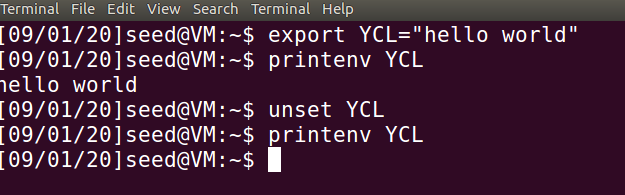
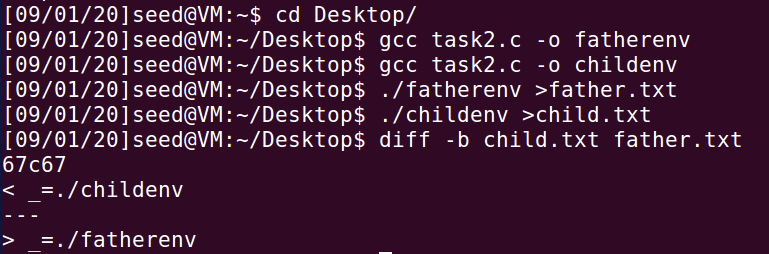
Task1



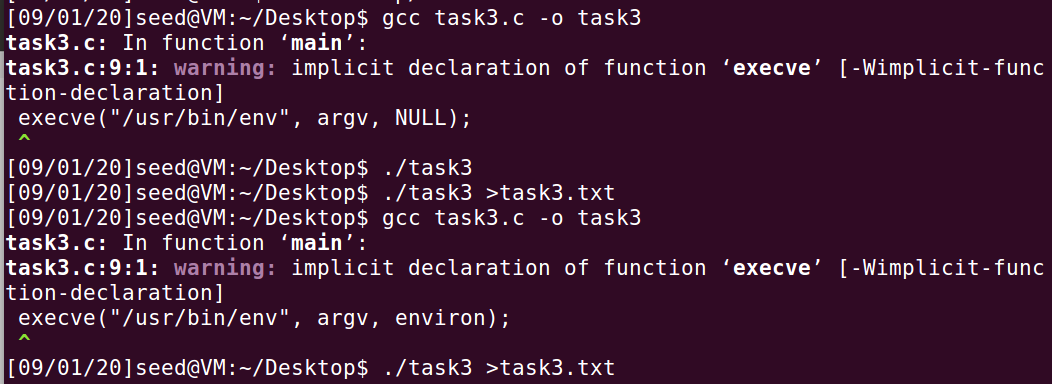


Task2



通过两次注释生成的两个程序分别输出父进程的环境变量与子进程的环境变量，可以发现这两个进程的输出是完全一致的 除了上图所示的表示进程名字的。所以可以下结论子进程的环境变量可以继承父进程。

Task3





在第一步中由于传入的环境变量为NULL，所以程序输出为空，第二步传入环境变量后程序输出了正确的环境变量。所以执行execve并不会继承之前的环境变量，会使用传入的环境变量参数覆盖。

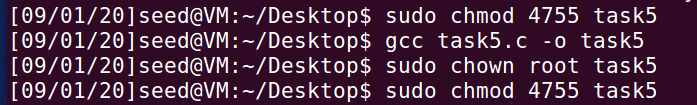
Task4

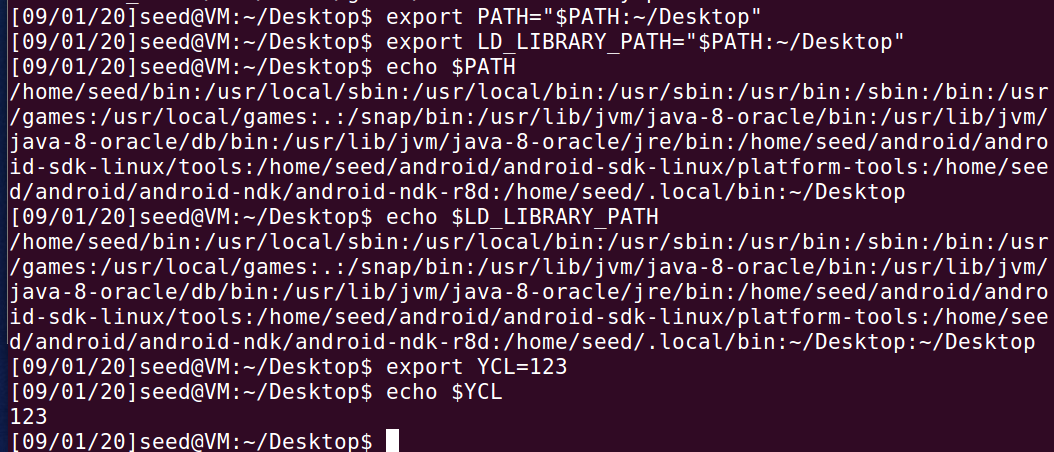


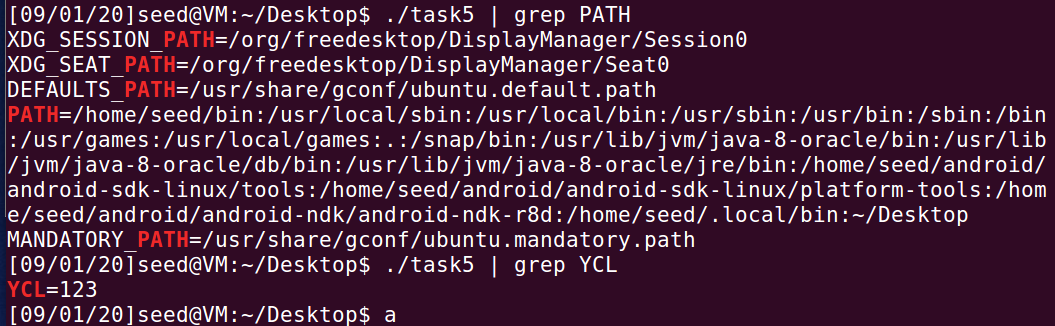


通过对比这个程序的运行结果与在sh中输入env后的结果对比可知，他调用了/bin/sh，并将环境变量传递给了他。

Task5



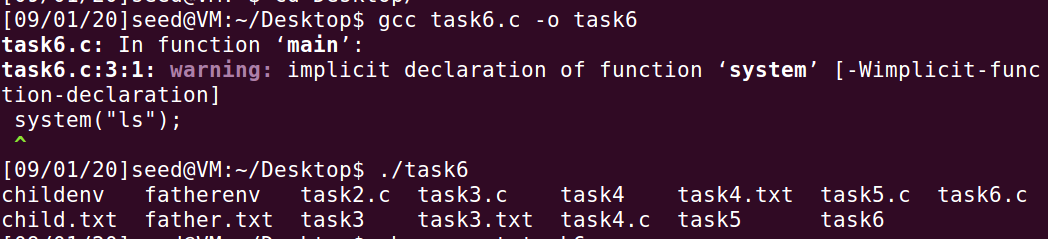




LD\_LIBRARY\_PATH不存在。

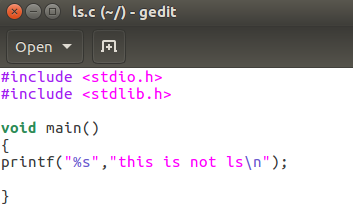
如图所示，PATH与我自己定义的YCL进入了该进程的环境变量中，但是LD\_LIBRARY\_PATH被忽略了。

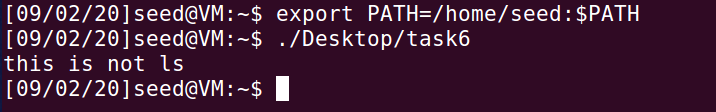
Task6





首先编译并将其转化为Set-UID程序，然后执行，正确的显示了ls的结果。

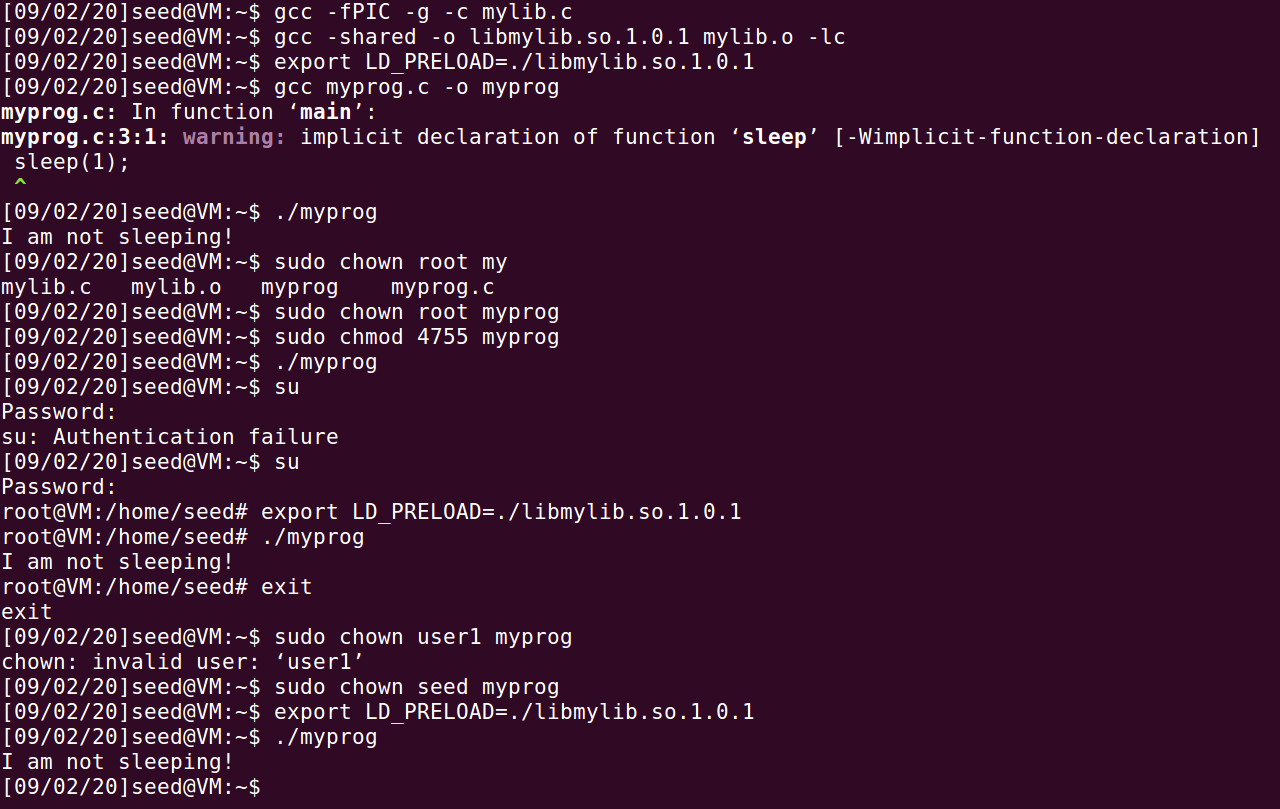




随后编写一个上图的代码，并将其添加到了PATH的头部，再次执行task6，随后执行了我写的ls代码。如此拥有了root权限

如task5实验所示，Set-UID程序的环境变量会继承PATH，将我编写的程序所在目录添加至PATH头部，如此该程序就会执行我的代码。但是如此是拿不到root权限的，因为执行shell会被自动降权，需要将原来的shell链接到zsh

Task7





第一种情况程序执行重写的sleep函数

第二种情况程序执行原本的sleep函数

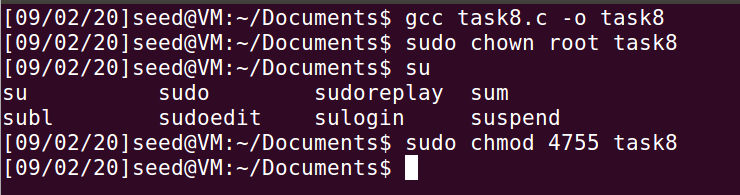
第三种情况程序执行重写的sleep函数

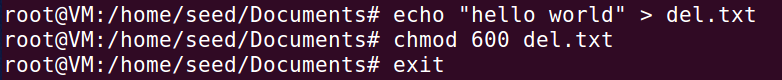
第四种情况程序执行原本的sleep函数

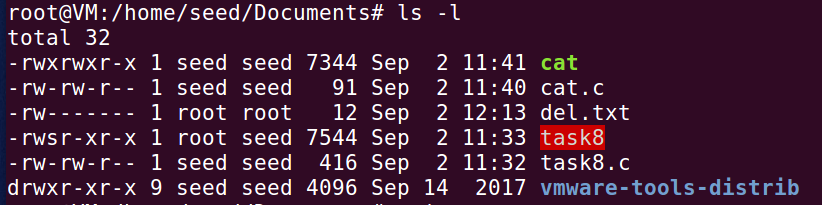
由task5可知，set-UID程序不会继承父进程的LD LIBRARY PATH，所以这里会执行原来的sleep函数，本问题的关键还是在于程序的环境变量是否被我们更改成了LD\_PRELOAD=./libmylib.so.1.0.1

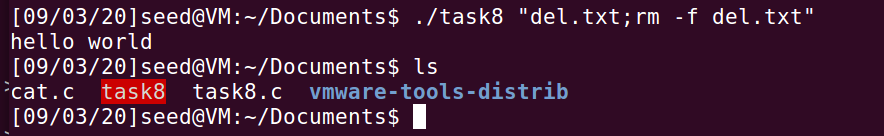
第一种情况显然更改了，第三种情况相当于在root用户的环境变量写入上文，第四种情况由于拥有者是seed，同第二种不继承。

Task8

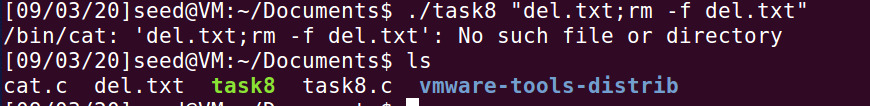








在语句后加；程序就会继续执行后面的删除命令。

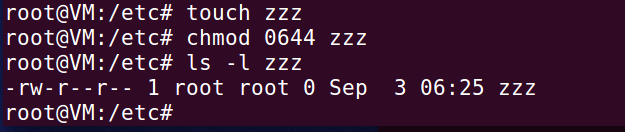


但是注释掉system使用execve后便会发现原来的攻击失效了。

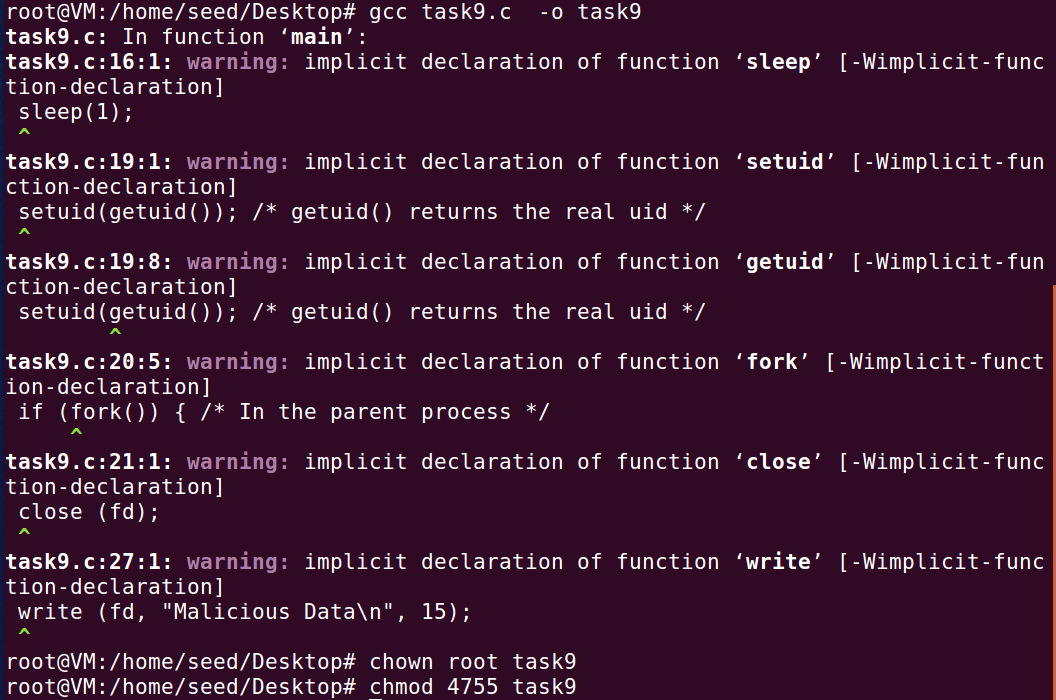
这是因为execve分别传入命令与参数，在参数中用分号间隔并不能让他成为两条命令。

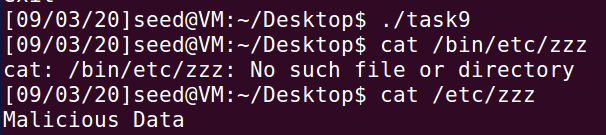
Task9

首先创建zzz文件



编译设置setuid程序





在普通用户下运行该程序可以发现Malicious Data已被写入。

原因在于设置uid前，该文件已被打开，所以进程对该文件仍具有root权限，如果在文件打开前设置uid就会提示没有权限打开文件，印证了之前的原理。