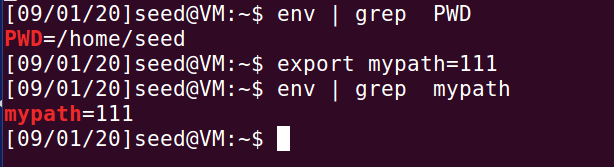
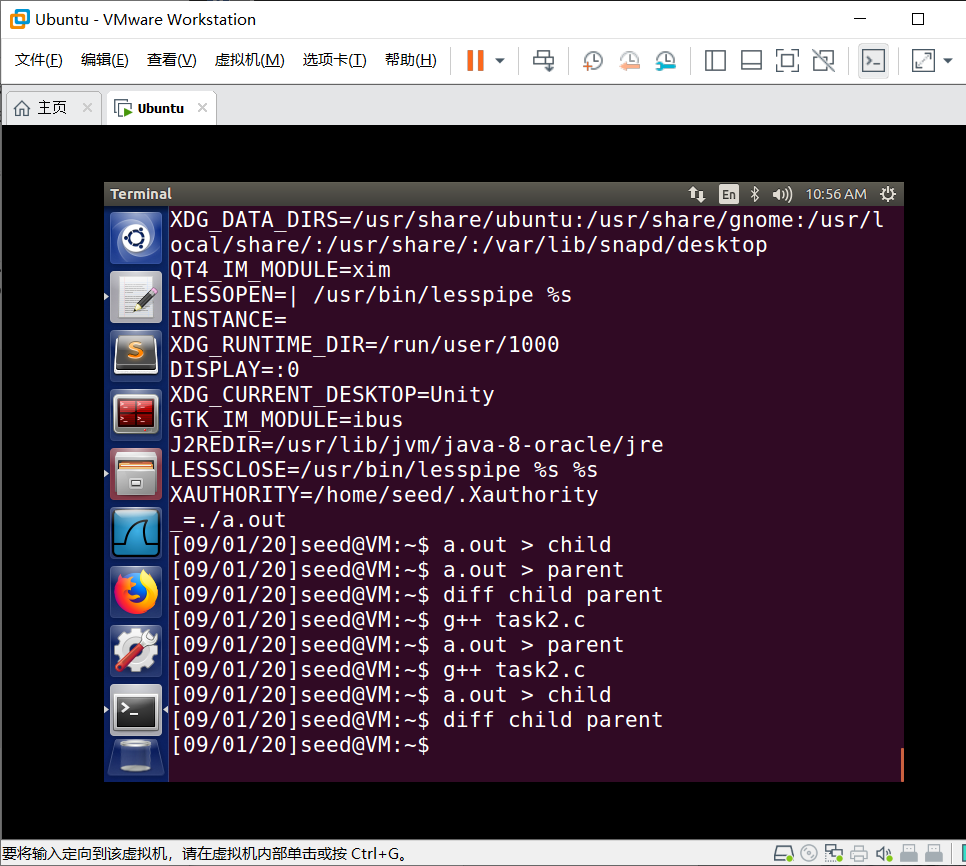
Task1: Manipulating Environment Variables

使用env命令查看相对应的环境变量，并使用export命令定义新的变量mypath，如图所示：



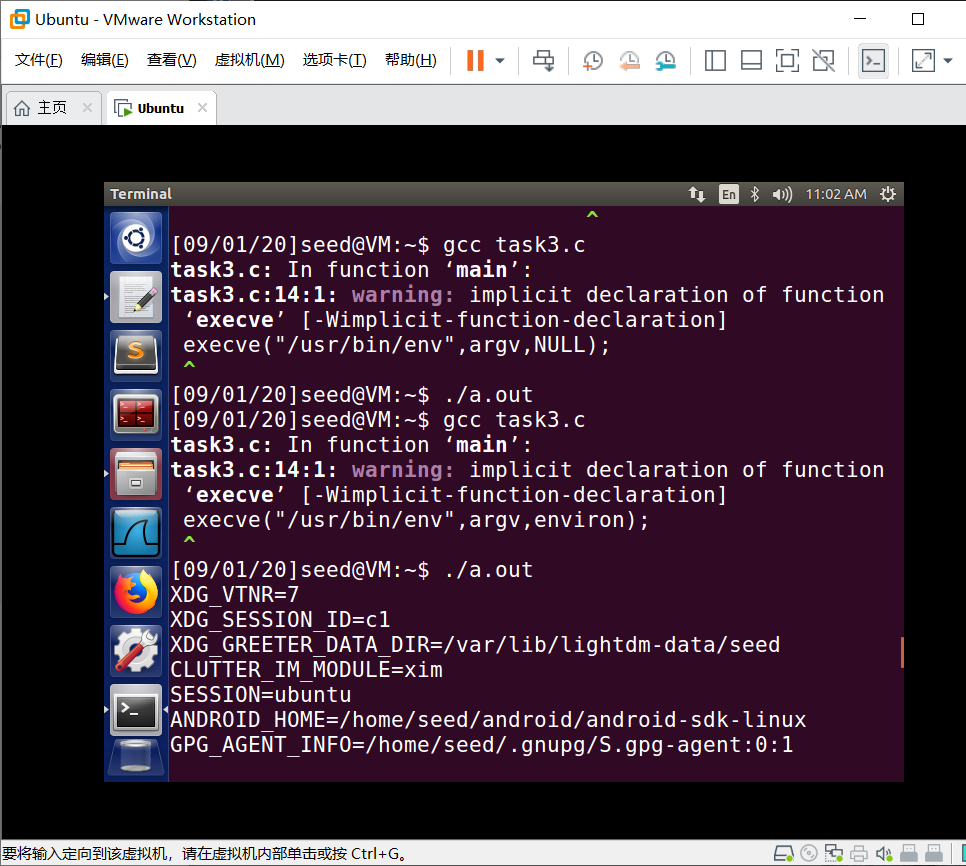
Task 2: Passing Environment Variables from Parent Process to Child Process

编译task2.c，分别使子进程和父进程的环境变量写入child和parent文件中，并用diff命令比较两个文件，结果发现两者完全一样，说明子进程继承了父进程的环境变量。



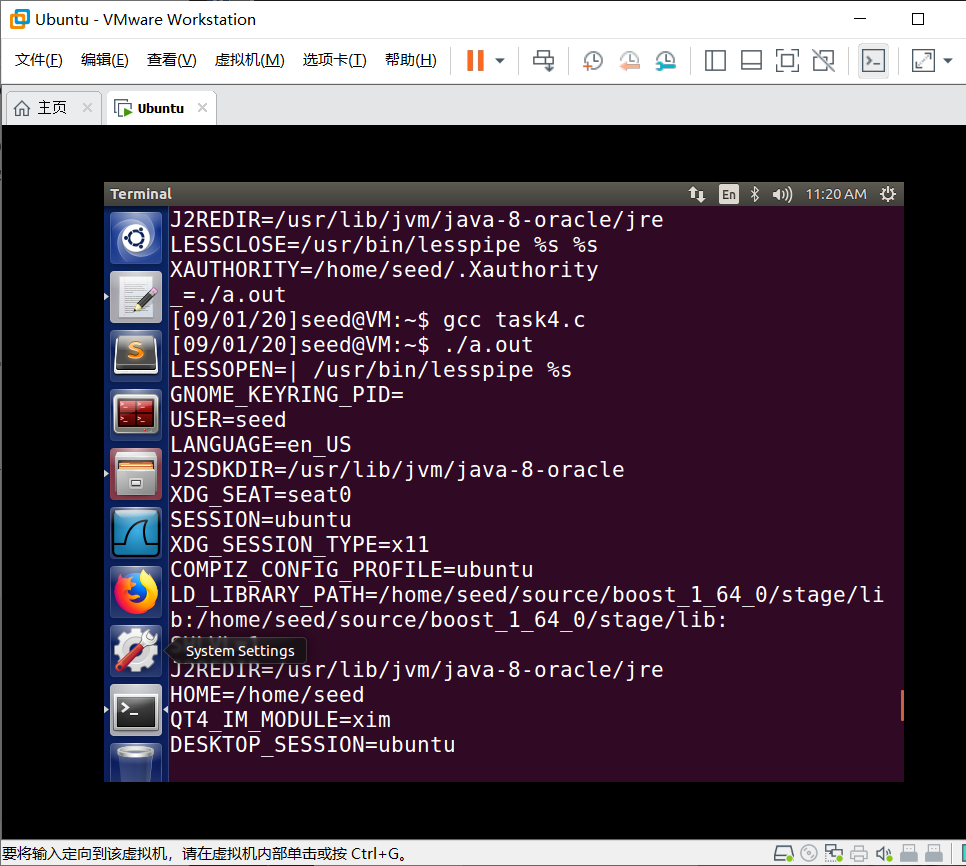
Task 3: Environment Variables and execve()

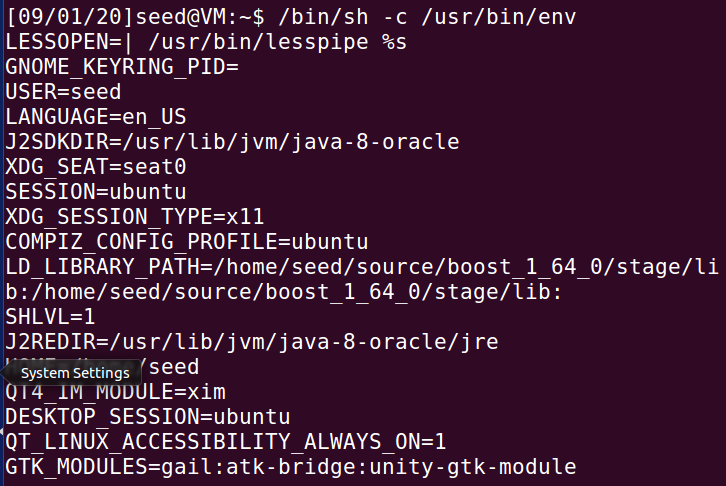
首先将execve的第三个参数设置为NULL，执行后并没有执行；将参数设置为当前的环境变量后，执行了/usr/bin/env的命令。



Task 4: Environment Variables and system()

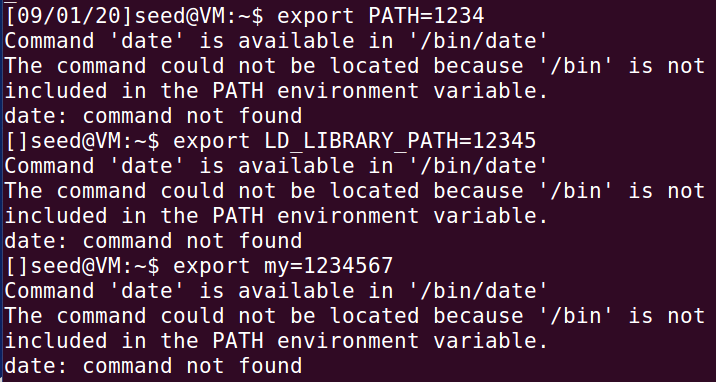
程序使用system调用执行/usr/bin/env的命令，该函数首先调用/bin/sh，再执行对应的命令。



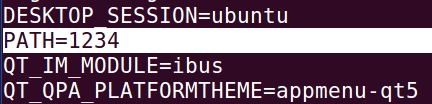


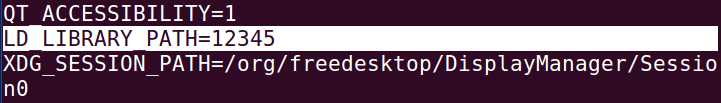
Task 5: Environment Variable and Set-UID Programs

设置PATH变量和LD\_LIBRARY\_PATH变量为1234和12345，新定义变量my为1234567，执行Set-UID程序task5.c，可以发现相关变量得到了更新。



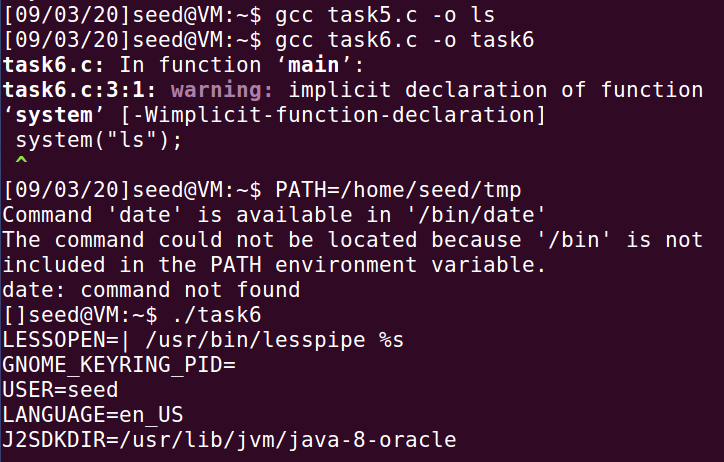






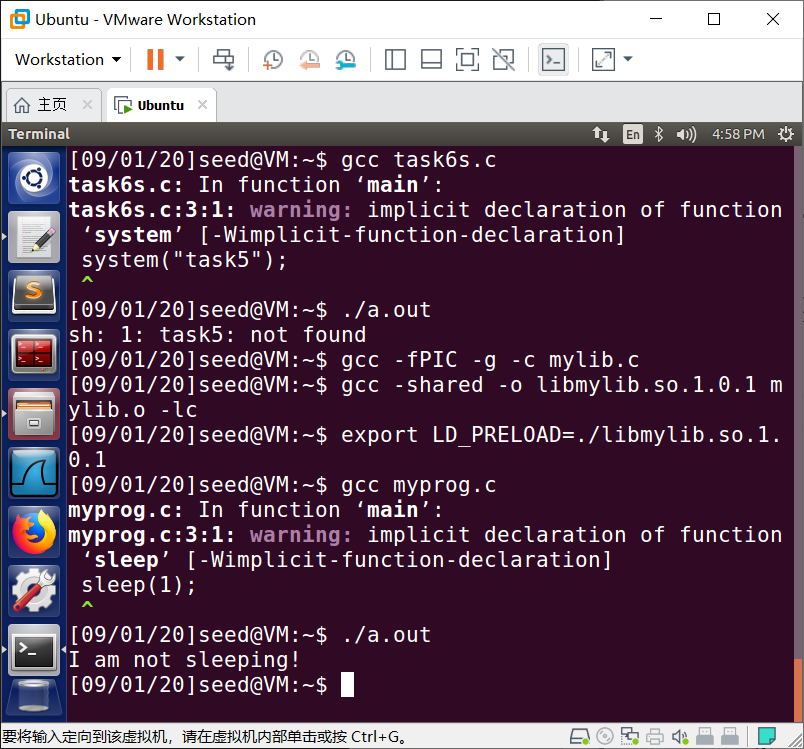
Task 6: The PATH Environment Variable and Set-UID Programs

将task5.c编译命名为成ls文件，并将环境变量PATH改为ls文件所在的目录，执行task6可见程序并不是列出当前目录，而是执行task5的程序。即输出所有环境变量。

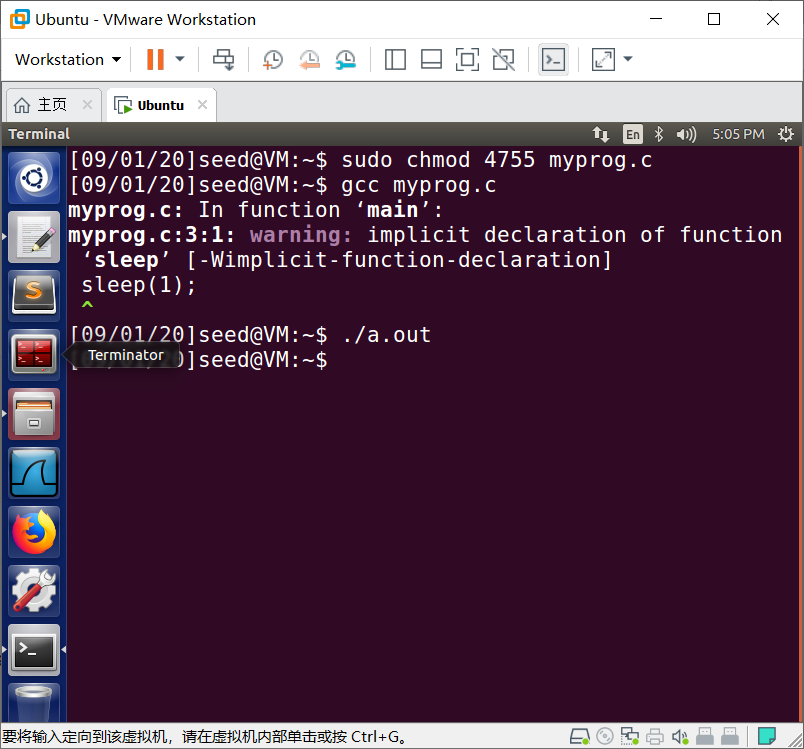


Task 7: The LD PRELOAD Environment Variable and Set-UID Programs

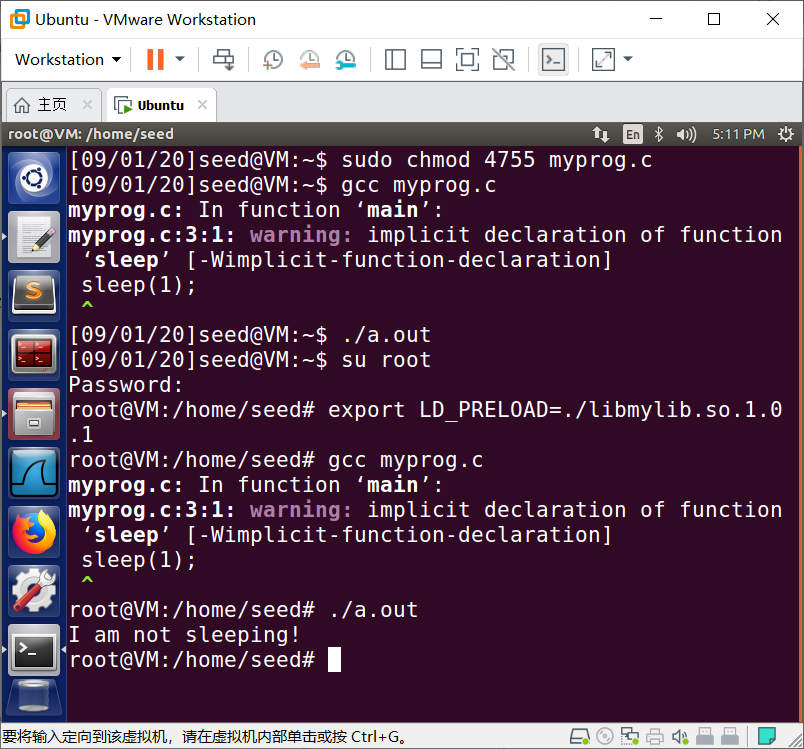
首先在seed用户下运行输出：I am not sleeping!



以seed用户运行拥有root权限的myprog程序时，无输出。

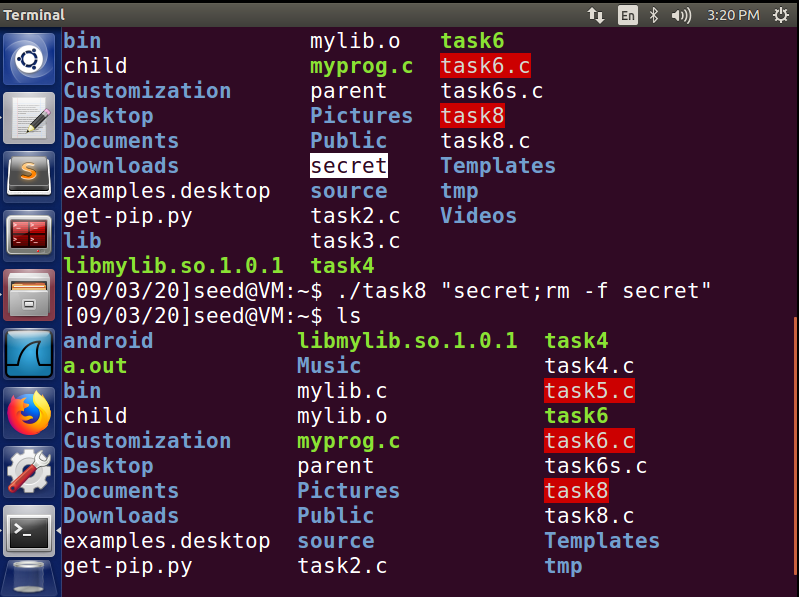


在root用户中再次设置LD\_PRELOAD环境变量并运行myprog程序时，输出：I am not sleeping! 只有用户自己创建的程序自己去运行才会链接到自己的sleep函数。

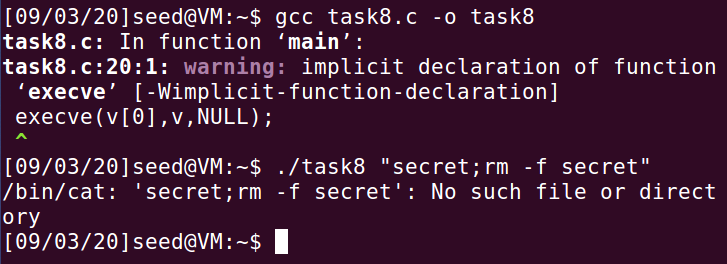


Task 8: Invoking External Programs Using system() versus execve()

编译task8.c并设置成set-UID，执行程序时输入secret; rm -f secret，可见secret文件就可以被删除。



如果将程序改为execve调用，攻击不会有效。可见system的工作方式是将后续的命令传递给shell，这样就可能被利用shell漏洞攻击。而execve是创建子进程，参数是传给子进程的，因此可以防止此类攻击。



Task 9: Capability Leaking

在etc目录下创建zzz文件，执行后可以发现zzz文件在子进程中被修改了。子进程拥有父进程的数据，堆栈的拷贝，同时拥有的父进程的setuid权限，因此有权限修改。

