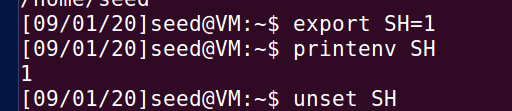
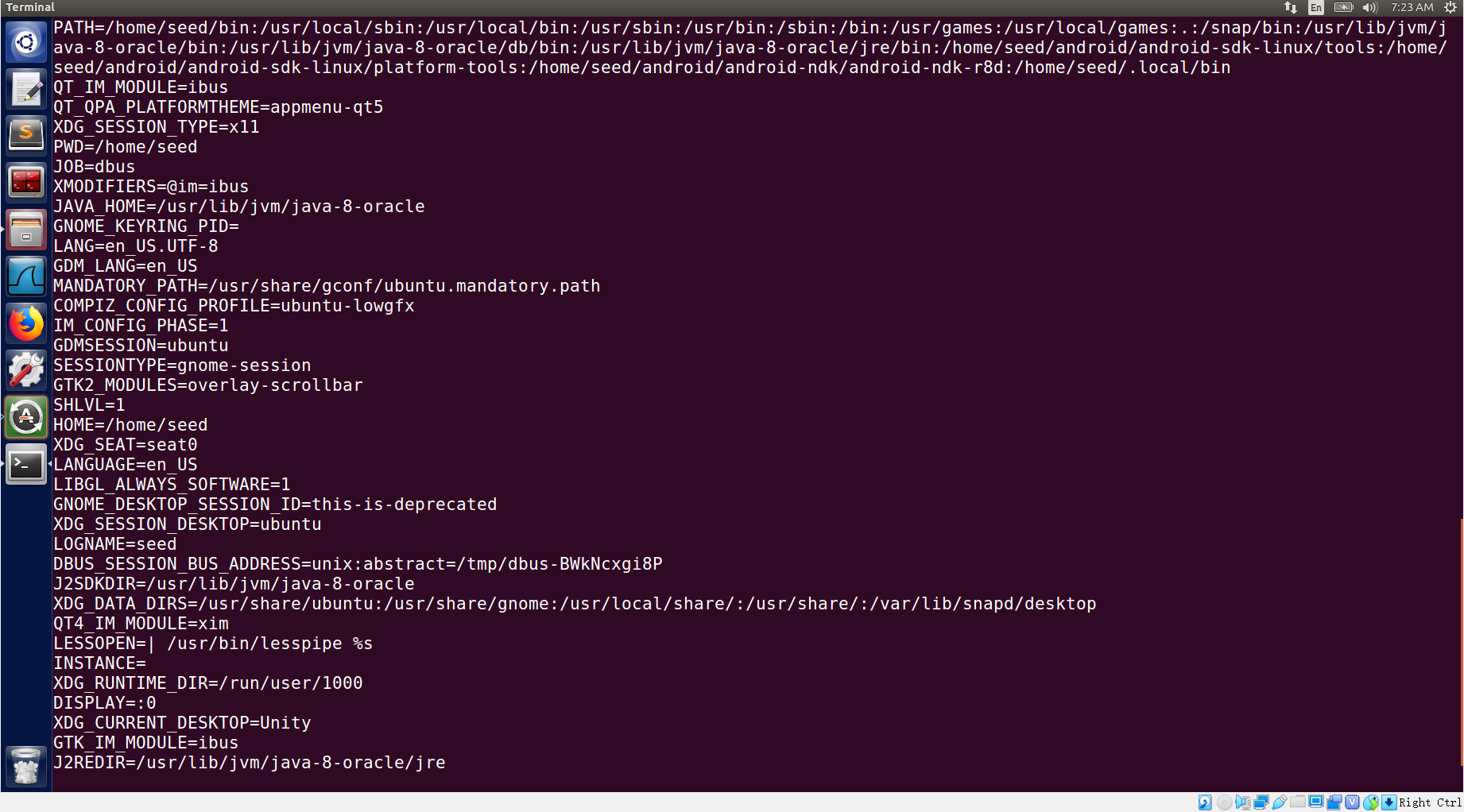
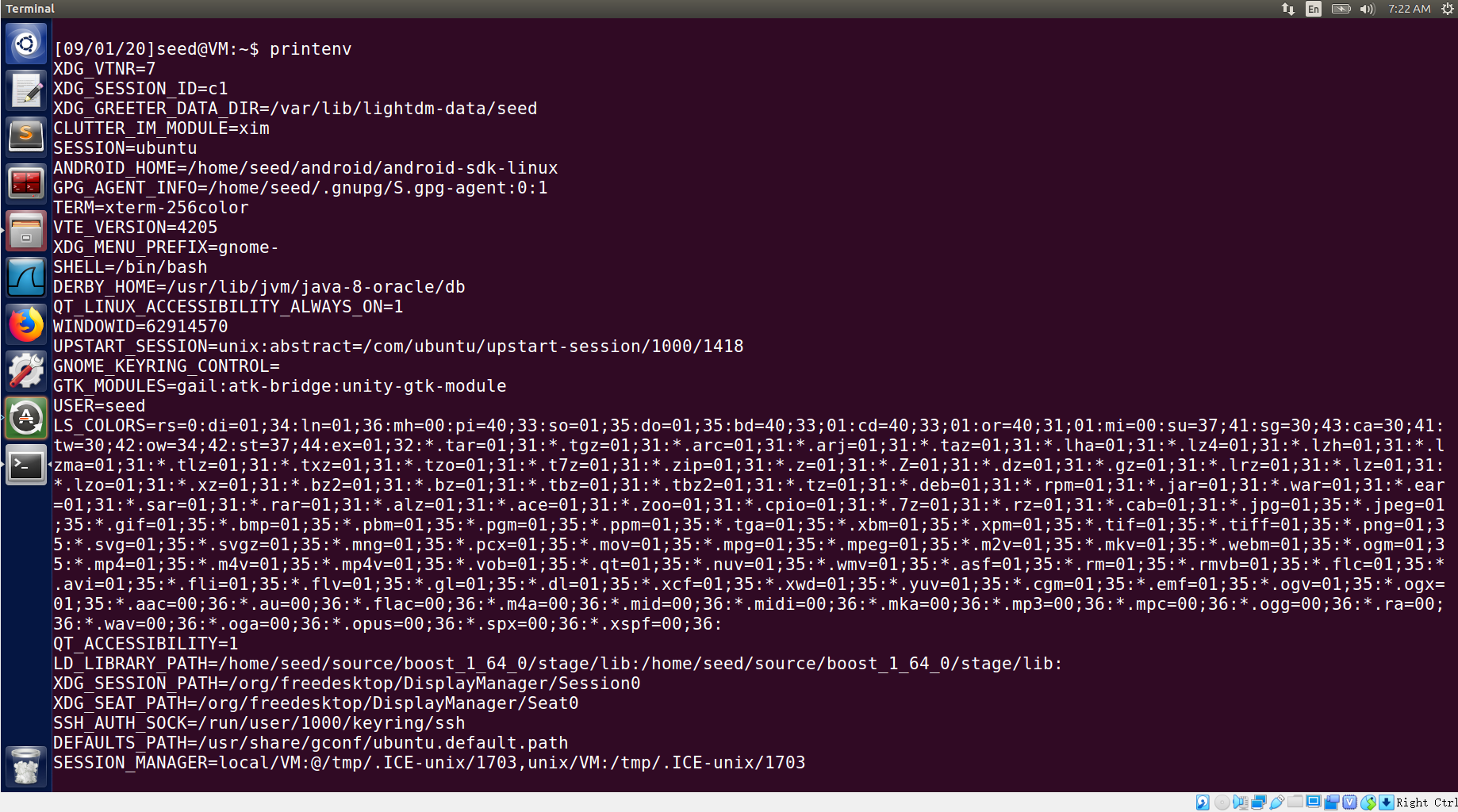
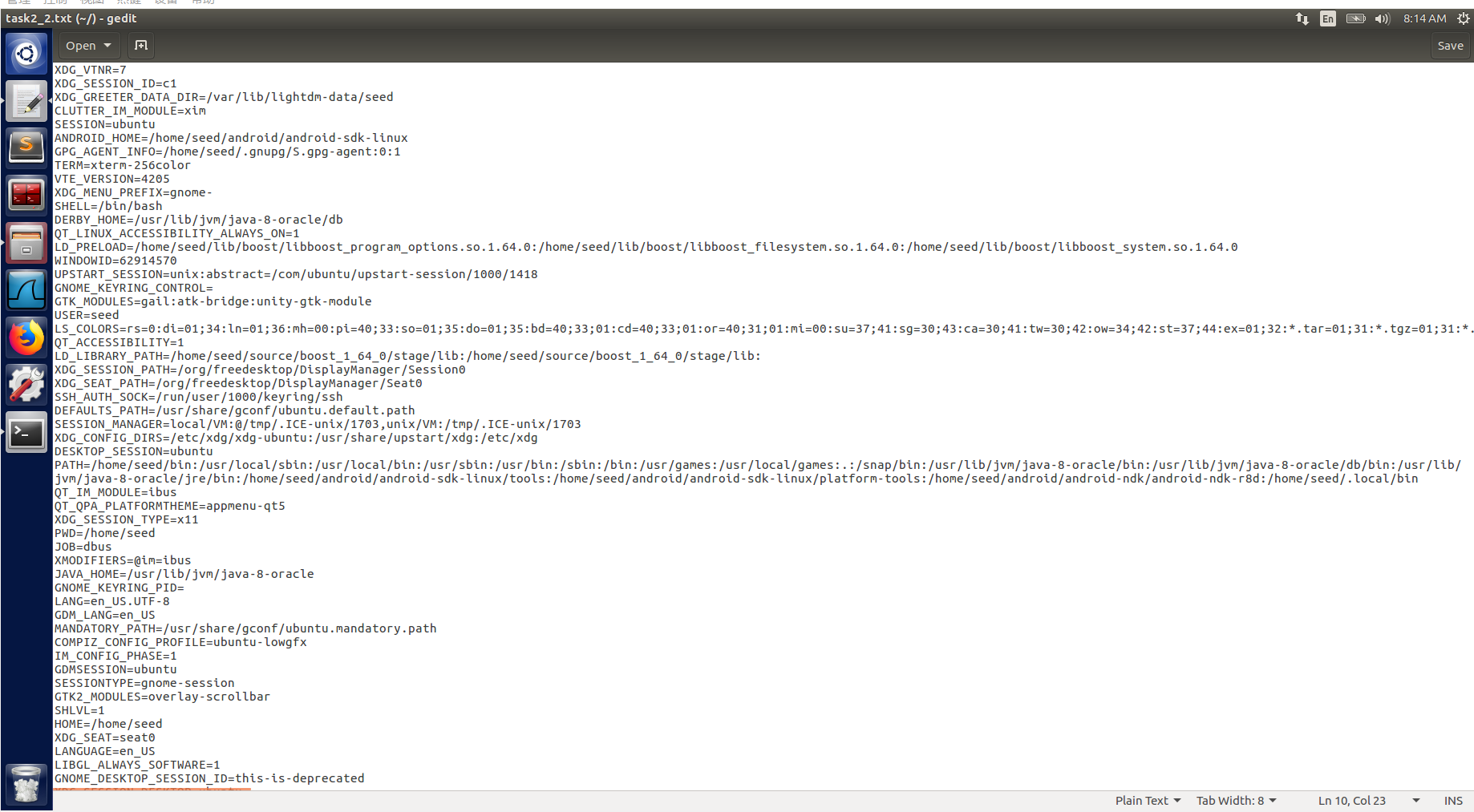
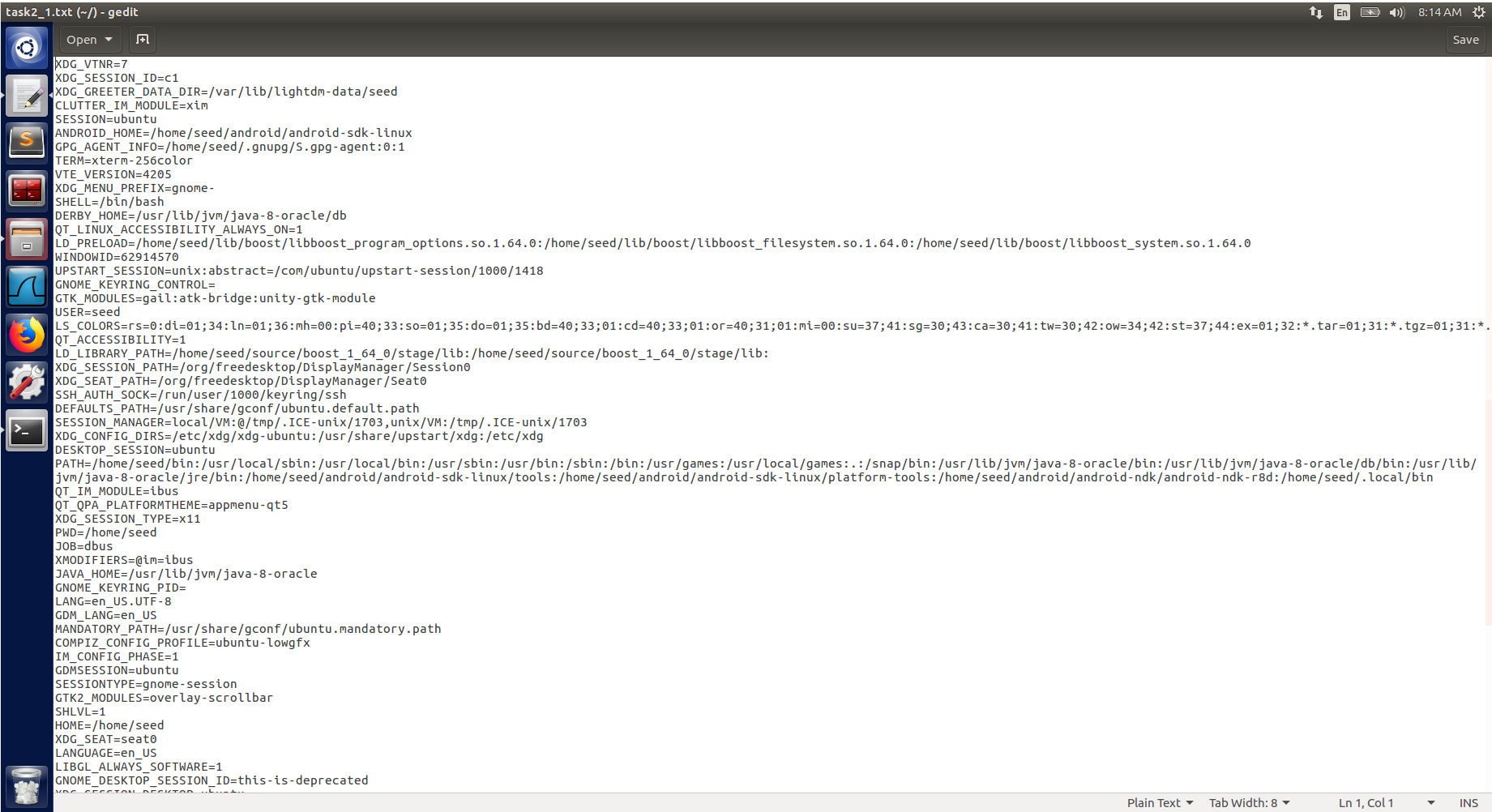
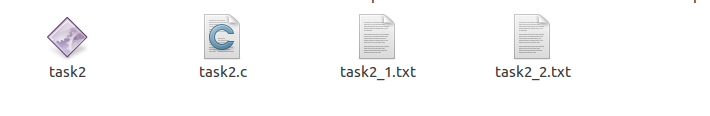
**Task1.**

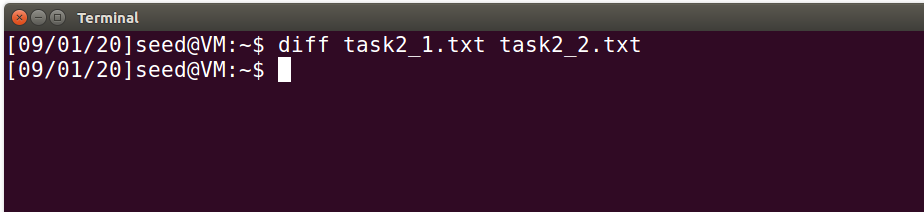


**Task2.**

将父子进程的环境变量分别保存在task2\_2.txt和task2\_1.txt中



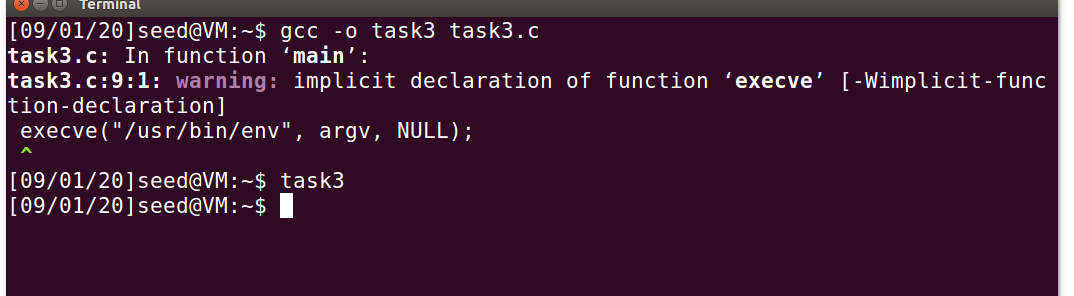
使用diff命令比较，发现没有不同



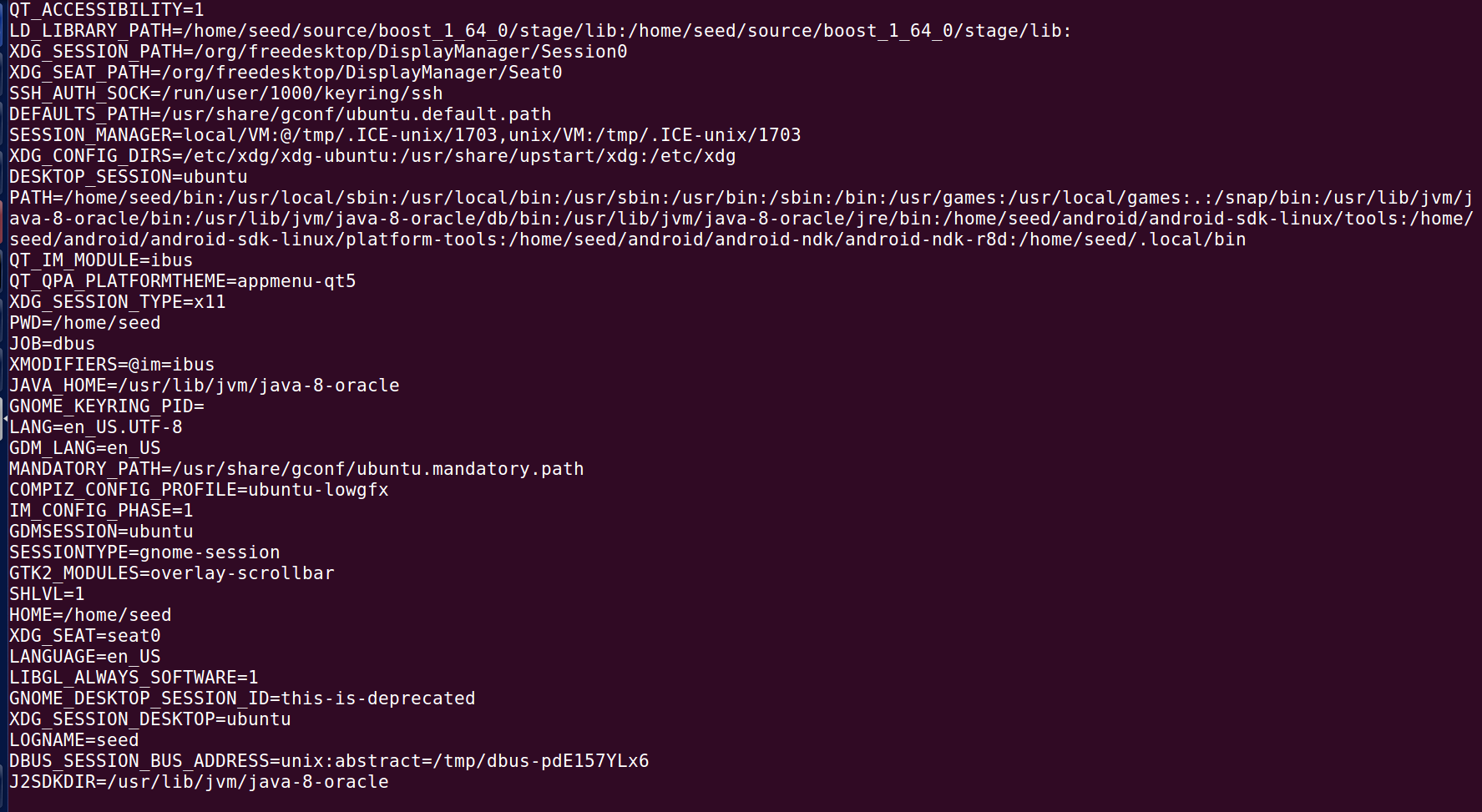
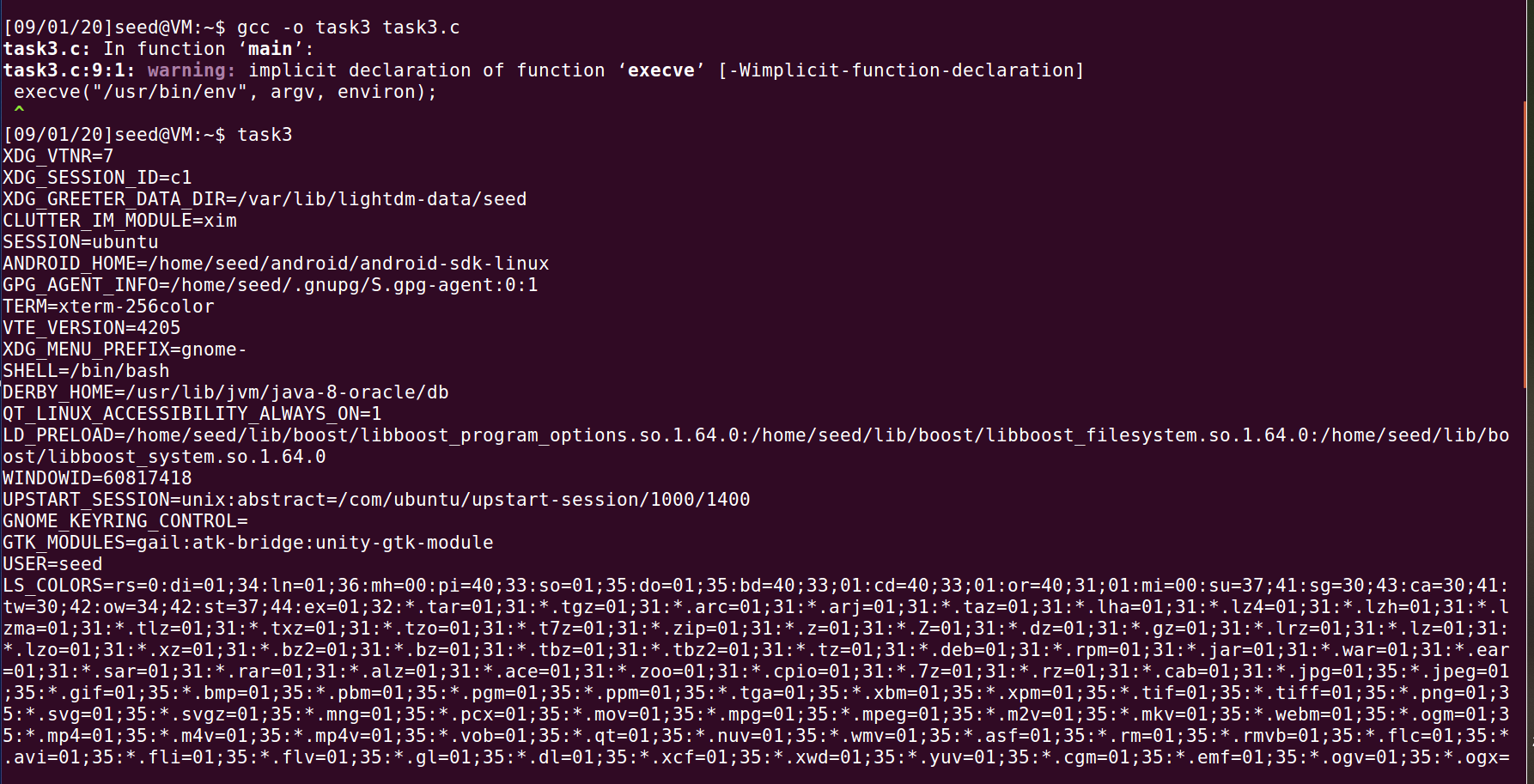
因此可以得出结论：fork()产生的子进程与父进程的环境变量完全相同

**Task3.**

Step1没有输出，如图

****

Step2输出如图

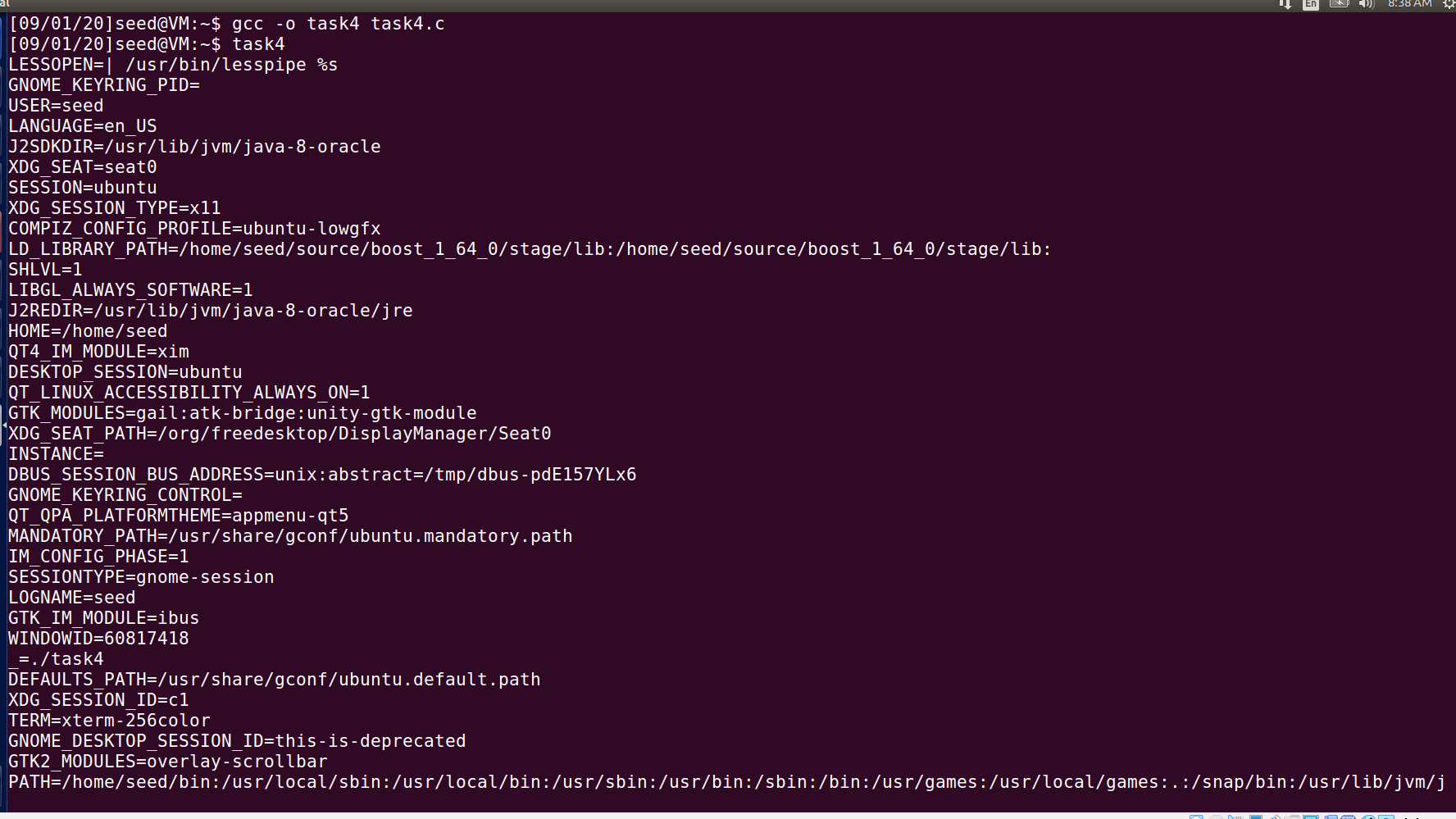
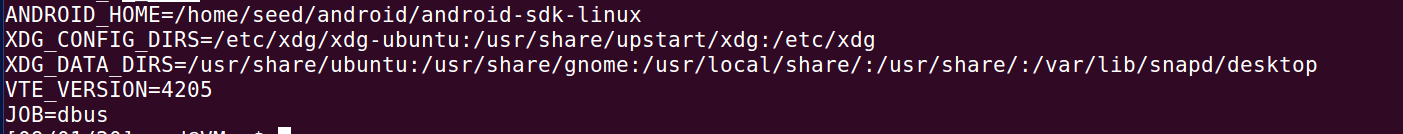
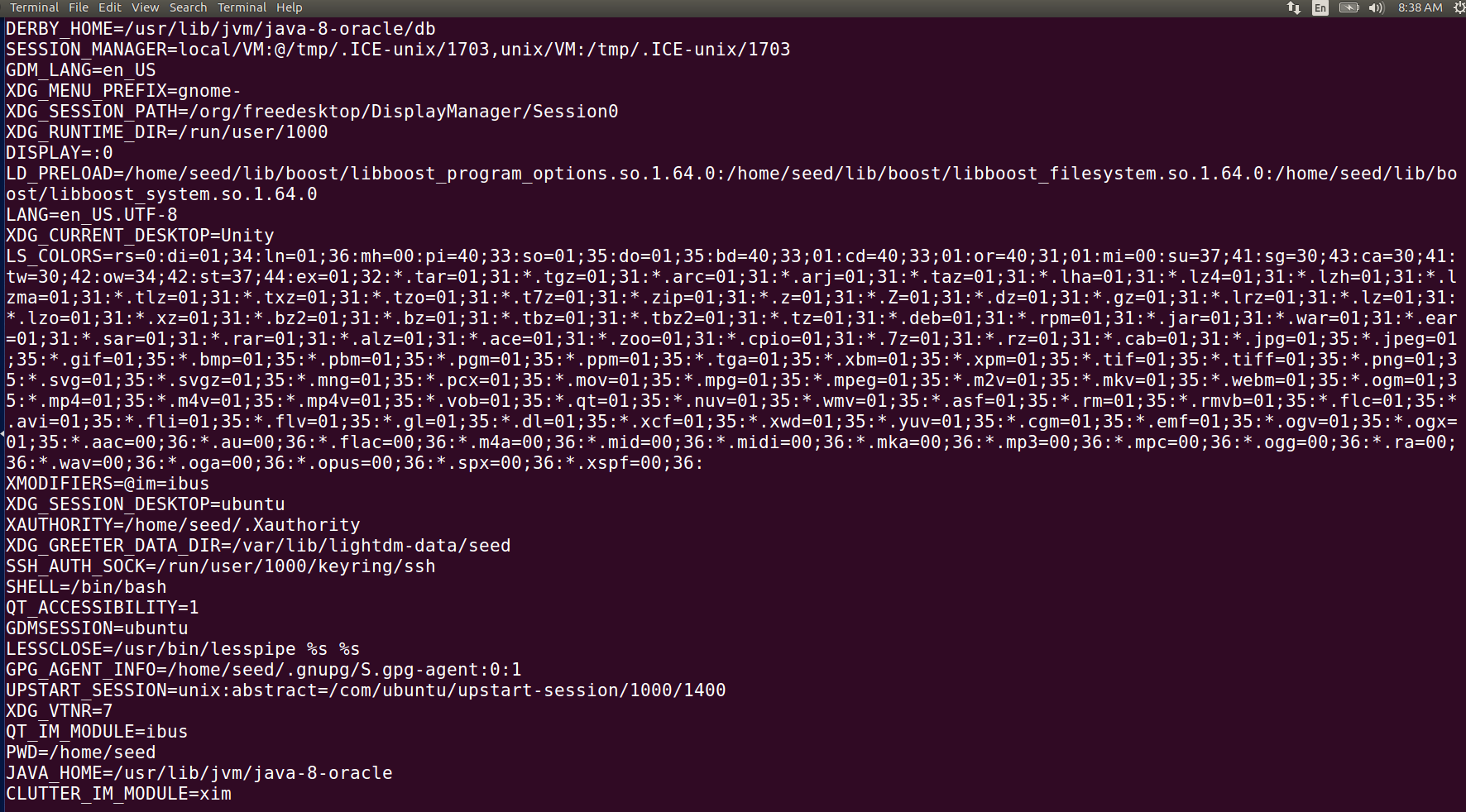


因此可以得出结论：

当execve()调用新的进程后，环境变量会直接继承于新的进程

**Task4.**

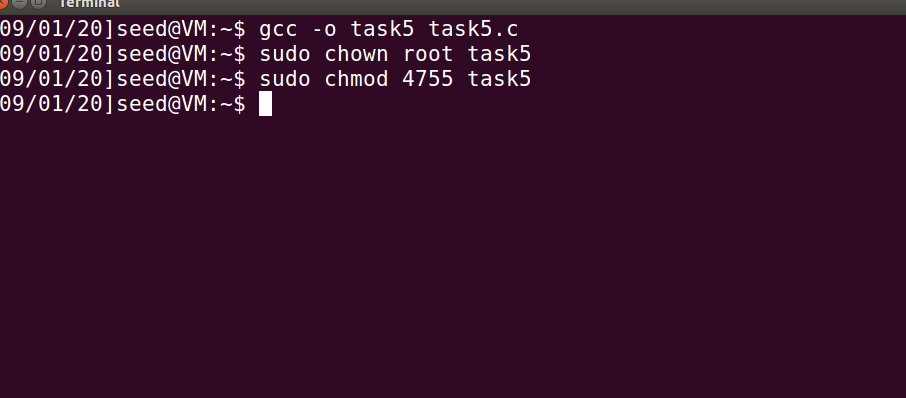
输出如下



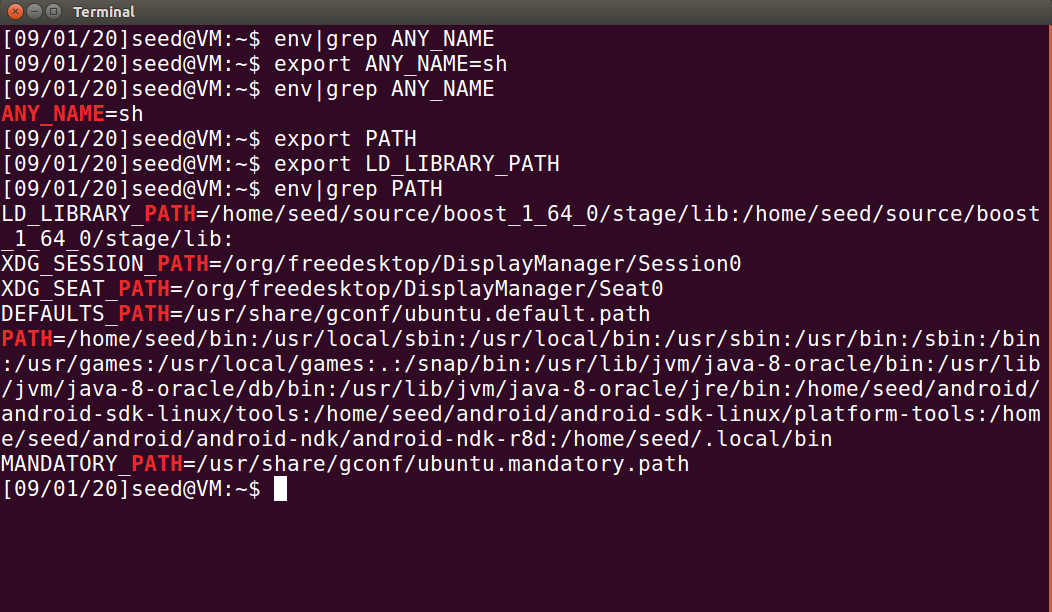
存储在目录中的环境变量通过system()命令打印在了shell中，验证了system()的实现是通过传递变量给shell实现的

**Task5.**

编译及前期步骤如图

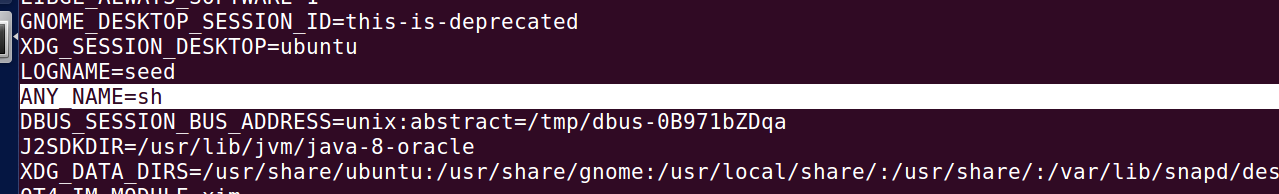
****

如下图所示设置环境变量并使用env|grep命令查看初始值

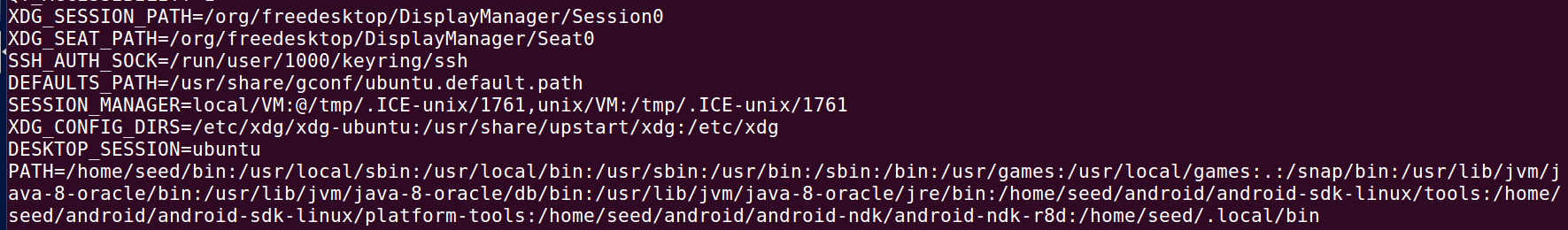
****

运行可执行文件后在输出中寻找相同名称的环境变量

下图中选中行是输出中的ANY\_NAME环境变量，可以发现与原先设置的值相同

****

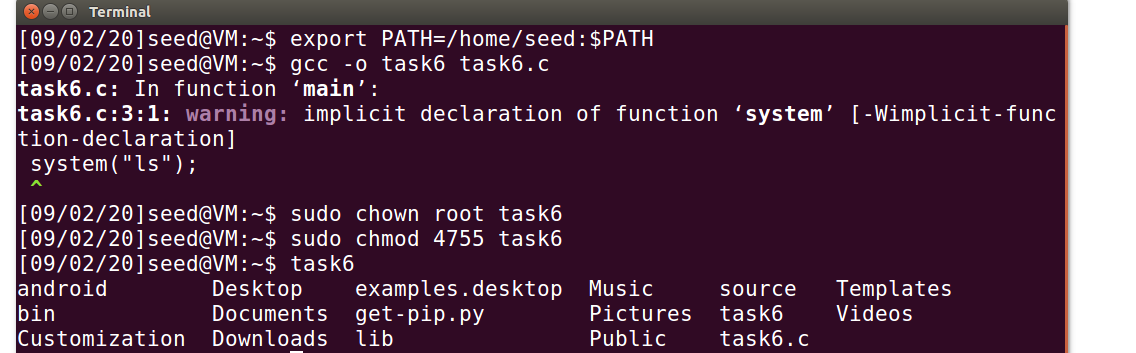
下面两张图是输出中的PATH环境变量，可以发现与原先的值也相同

****

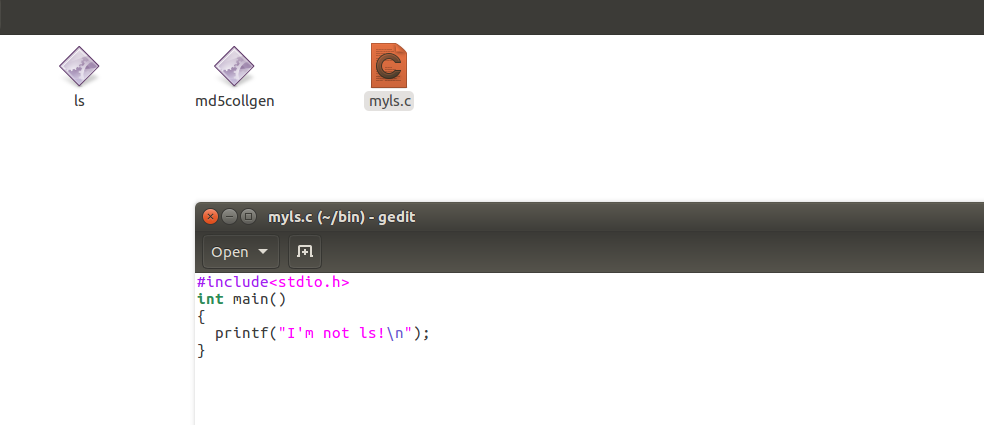
从以上实验结果可以发现，set-uid进程的环境变量继承于用户进程的环境变量

**Task6.**

修改环境变量PATH，编译含有system(“ls”)的程序task6，修改其owner为root并将其设为set-uid程序，执行正常的ls命令输出当前目录下的文件。

****

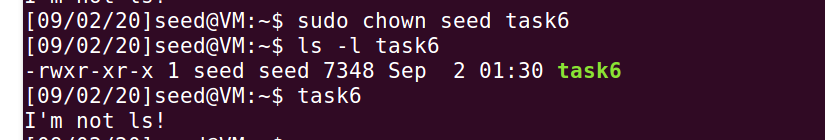
在home/seed/bin中写自己的myls.c文件并编译为可执行文件ls，具体内容为输出一句话：I‘m not ls!

****

然后再执行task6，发现输出了I‘m not ls！，说明成功的用自己的ls指令替代了原先的ls指令。

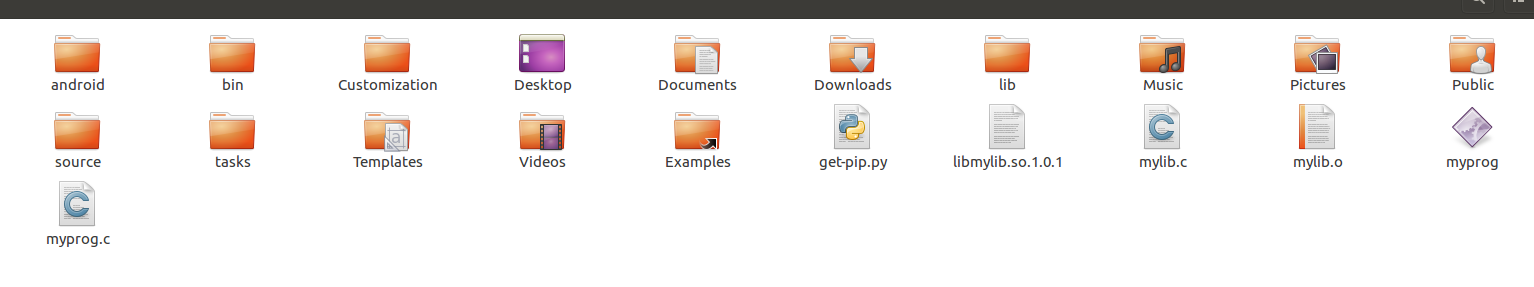
****

新的ls并非以root特权运行，如下图所示，即使将task6的owner变为seed仍然执行了新的ls

****

**Task7.**

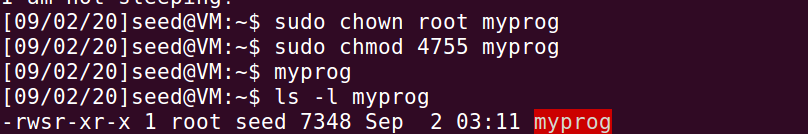
首先编译动态库及myprog程序，并通过export设置临时环境变量LD\_PRELOAD

****

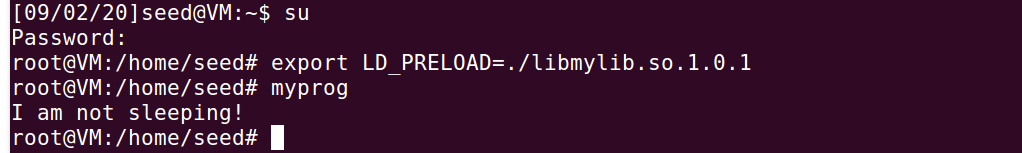
以普通用户身份运行非set-uid的myprog，结果如下，执行了新的sleep()

****

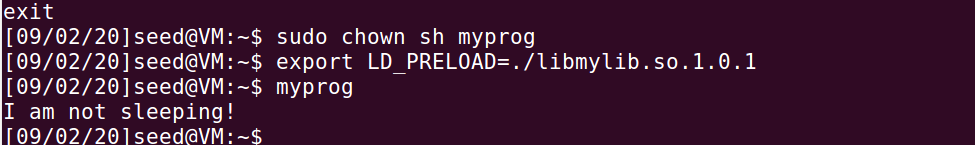
以普通用户身份运行set-uid且owner为root的myprog，结果如下，执行了原本的sleep()

****

以root身份运行set-uid且owner为root的myprog，结果如下，执行了新的sleep()

****

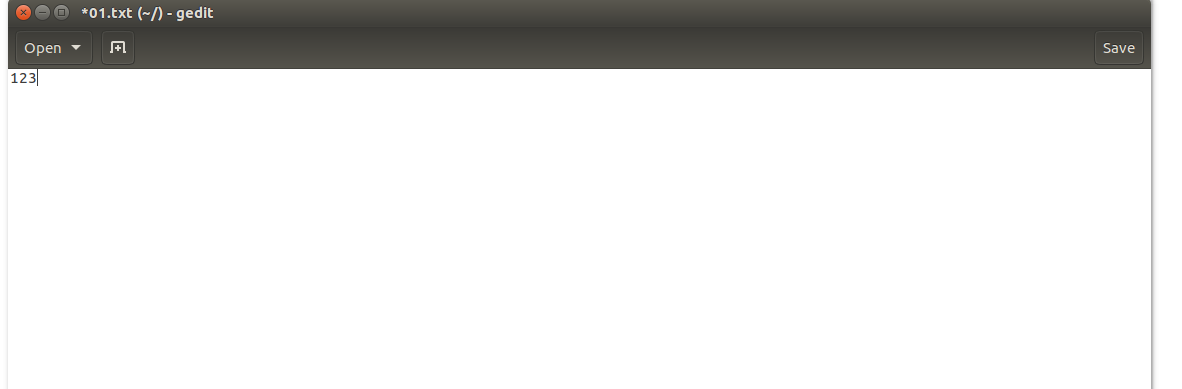
以seed身份运行set-uid且owner为sh的myprog，结果如下，执行了新的sleep()

****

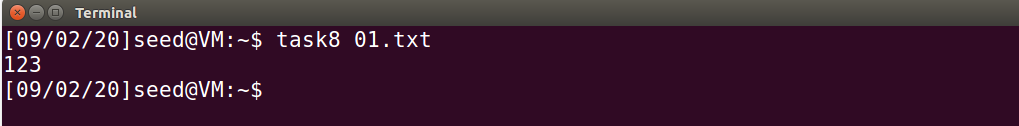
经过多次不同用户不同权限以及是否配置临时环境变量LD\_PRELOAD的测试，得出以下结论：当前用户的环境变量LD\_PRELOAD决定了程序运行时优先加载的动态链接库的目录，export命令可以在此次shell的运行中临时改变环境变量，root用户的环境变量在每次切换用户后就会回到默认状态，而普通用户的环境变量则不会因切换用户而变化。

**Task8.**

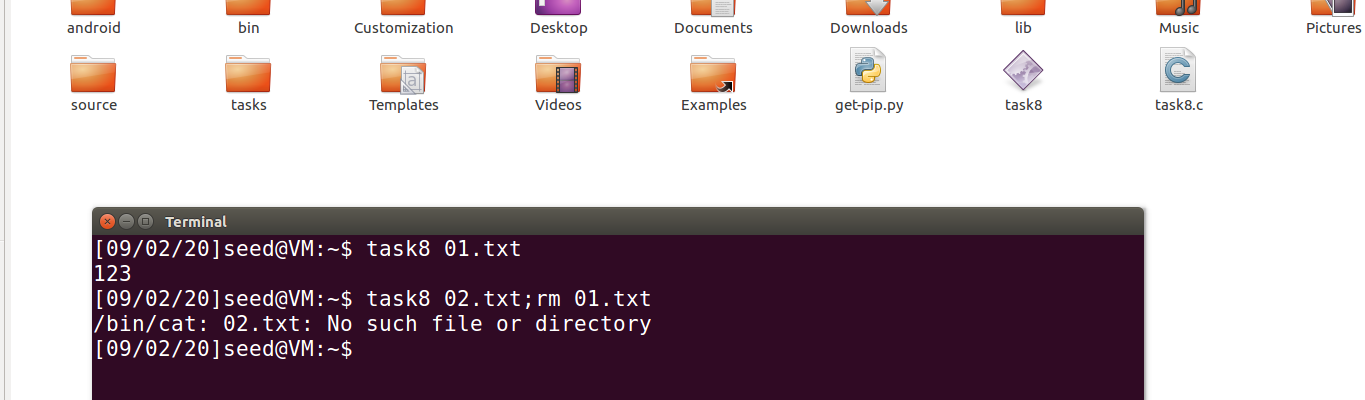
先编译使用system()命令版本的代码，如下图所示创建测试文件01.txt

****

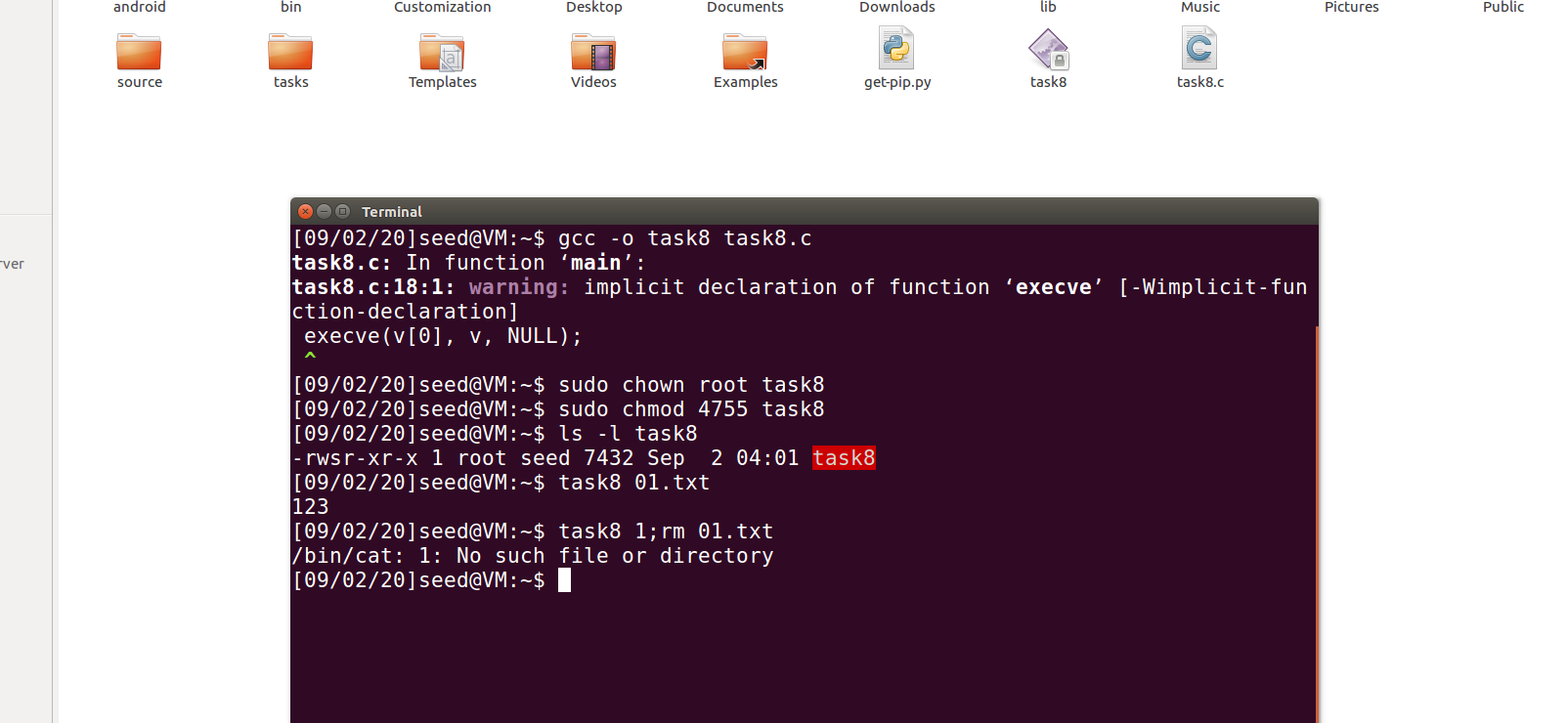
如图所示，可以正常使用

****

若使用不正当的输入可以删除文件，如下图所示01.txt被删除了

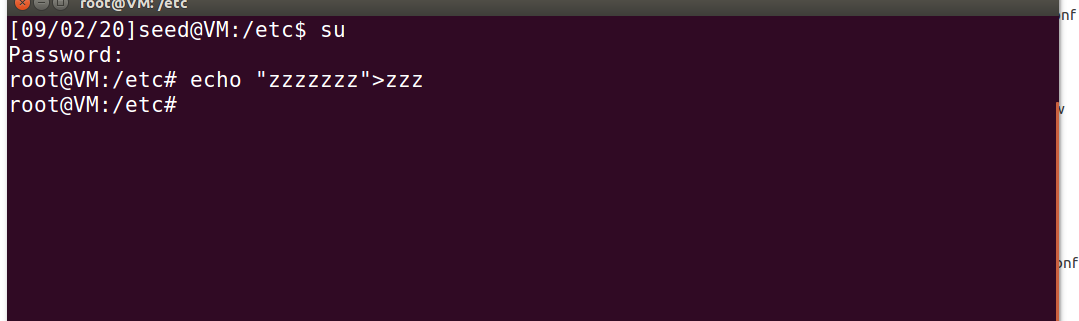
****

编译使用execve命令版本的代码，创建相同的测试文件01.txt，发现依然可以通过输入删除文件？？？？？陷入困惑，还是本来就是这样

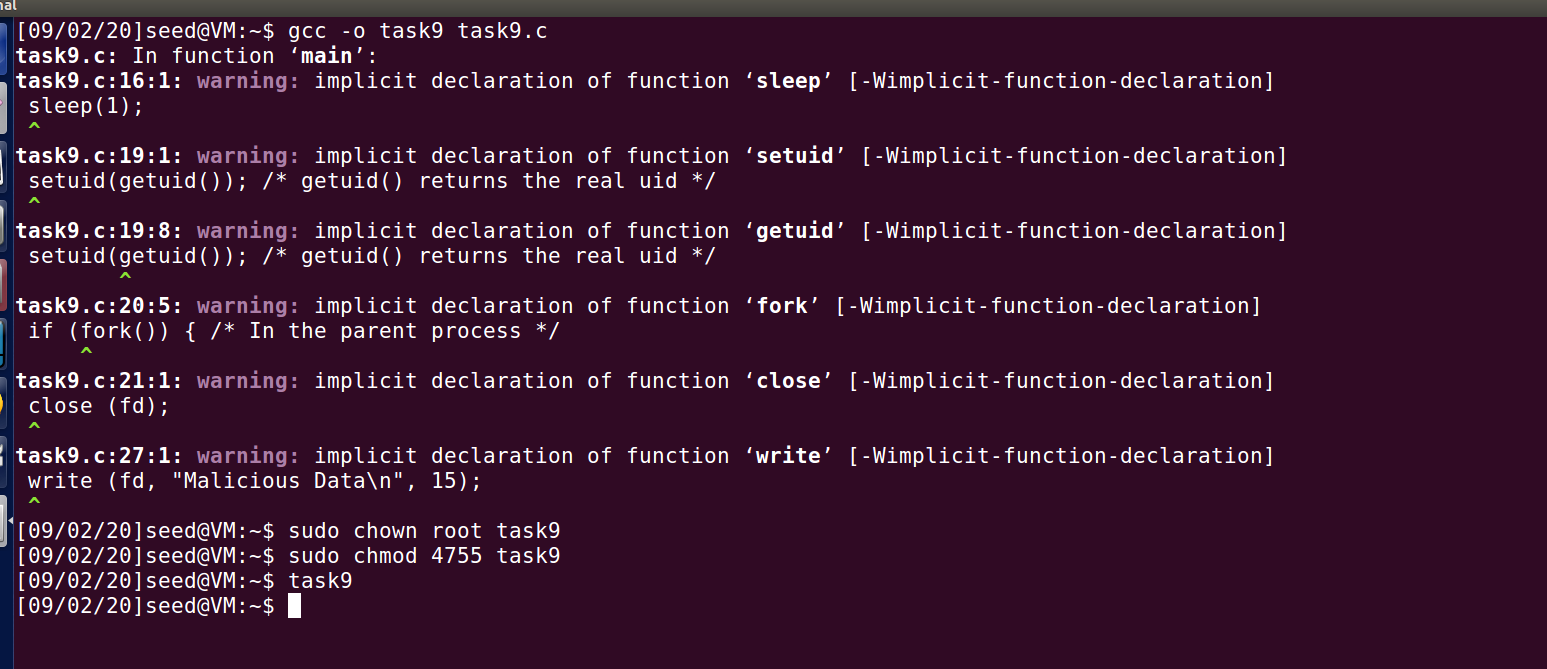


**Task9.**

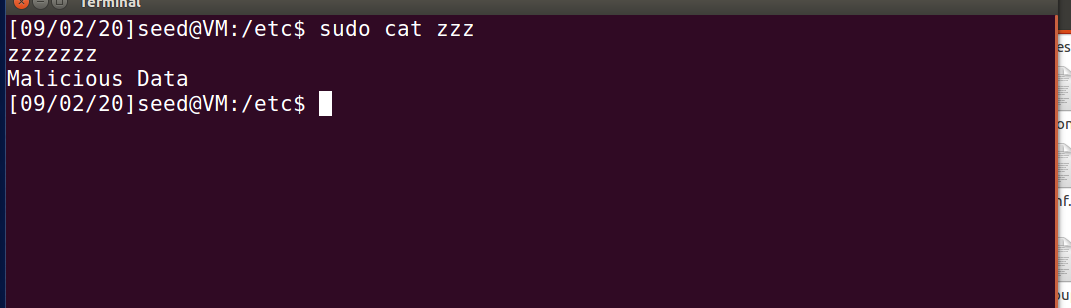
首先在/etc目录下创建zzz文件并随便输入一些字符，如下图

****

编译程序，修改owner并改为set-uid，执行程序

****

打印zzz文件发现内容已经被修改，说明在setuid()操作之后虽然用户的权限降级了但fd并未关闭，用户依然可以修改其中的内容

****