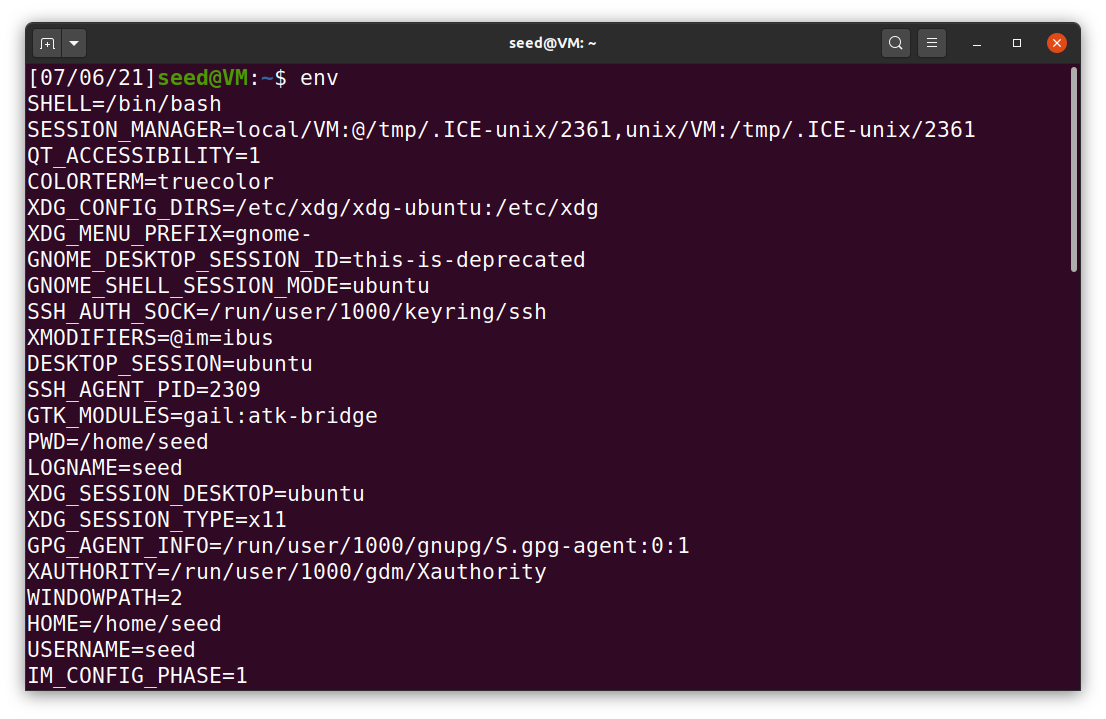
**Task1:操作环境变量**

1.使用printenv输出环境变量。

结果截图:



1. 使用export和unset设置或取消设置环境变量

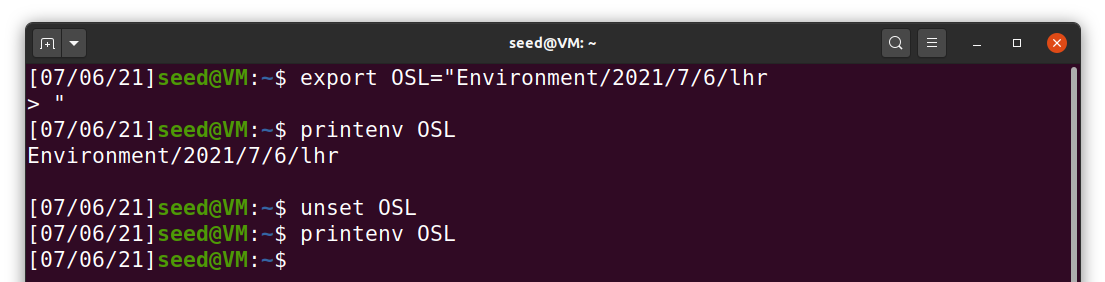
第一步：使用 exportOSL=Environment/Variable/and/Set-UID/Program/Lab” 命令设置环境变量OSL。

第二步：使用 printenv OSL 命令输出环境变量OSL的值。

第三步：使用 unset OSL 命令删除环境变量OSL。

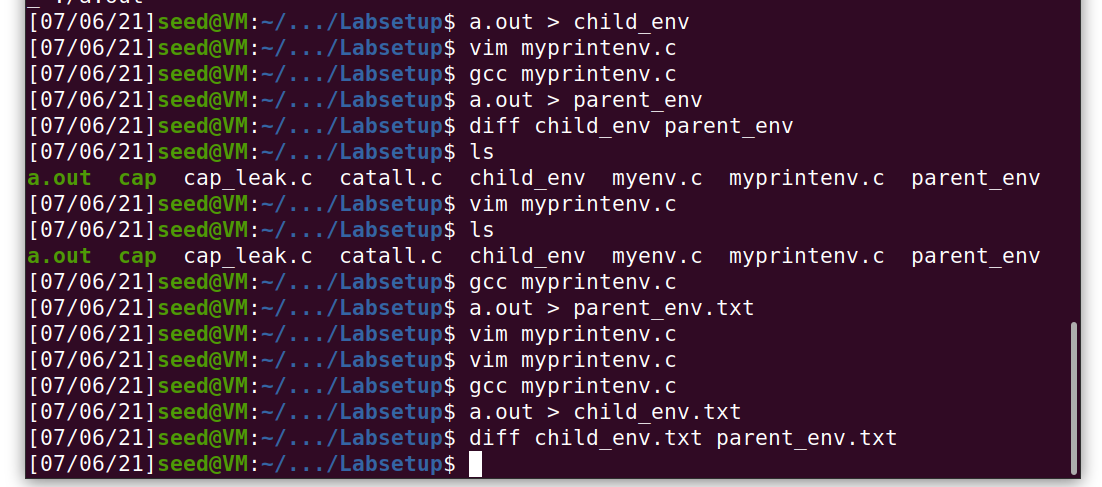
第四步：使用 printenv OSL 检验环境变量OSL是否已经删除。

结果截图：



**Task2:从父进程向子进程传递环境变量**

结果截图：

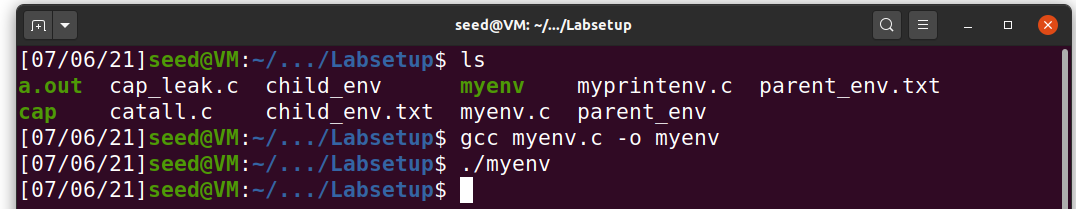


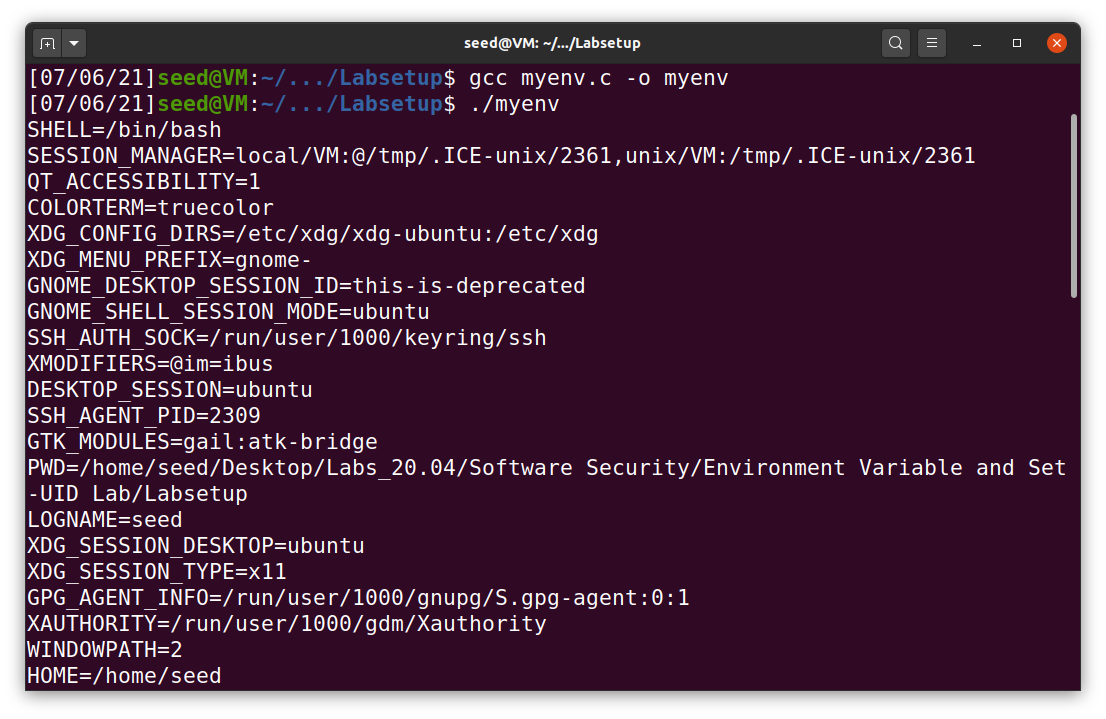




结论：子进程与父进程的环境变量完全相同，说明环境变量是可以从父进程传递给子进程的。

**Task3:环境变量和execve()**

没有传入参数：  


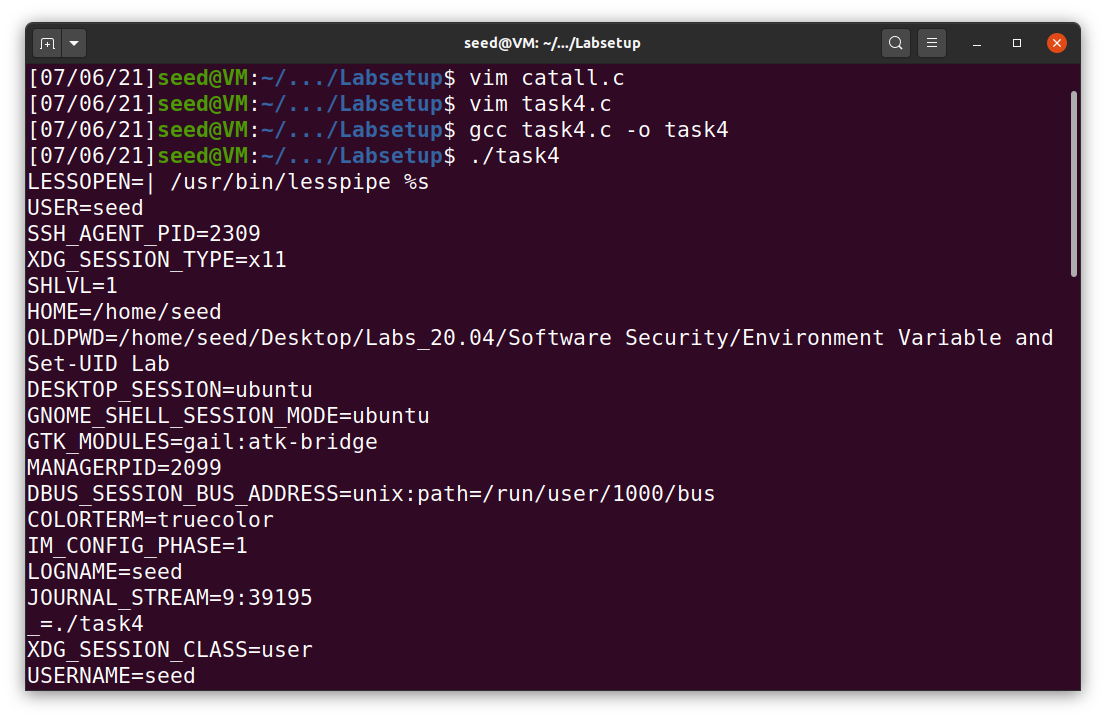
传入参数：  


实验现象：execve()中第三个参数为NULL时，程序没有输出，第三参数为environ时，程序输出进程的环境变量。

结论：此处新程序的环境变量不是自动继承当前进程的环境变量，需要传入相应参数才能继承环境变量。

**Task4:环境变量和system()**

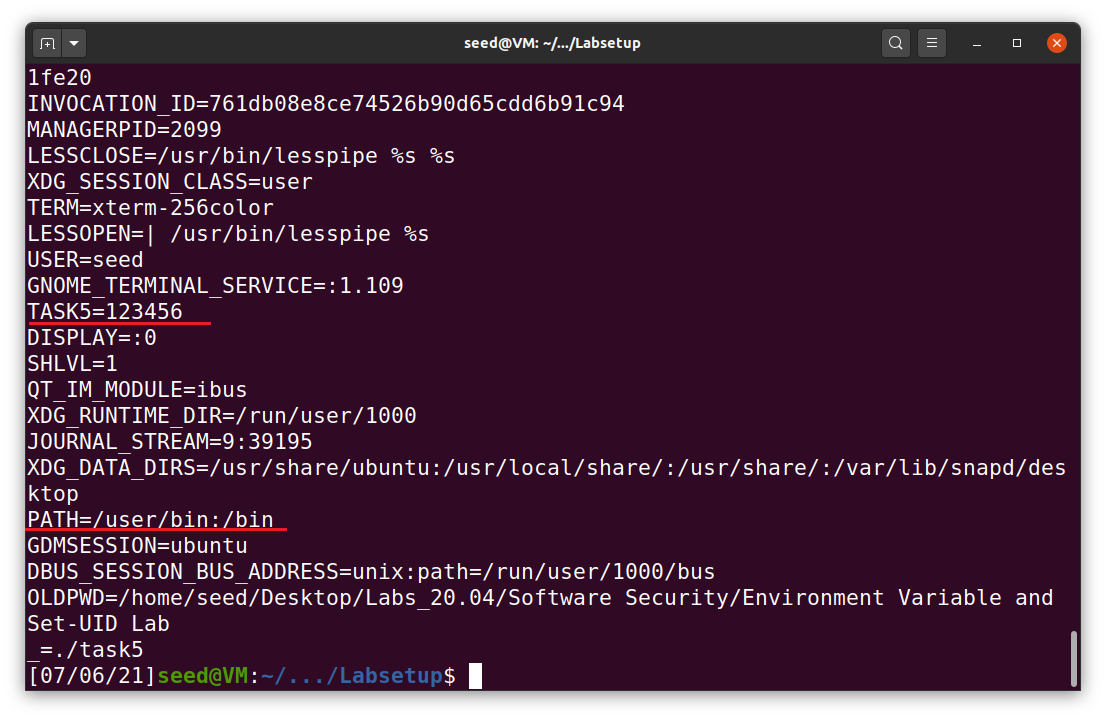
结果截图：



结论:程序输出了进程的环境变量，可以证实，通过使用system()，进程的环境变量传递给了新的程序/bin/sh。

**Task5:环境变量和Set-UID程序**

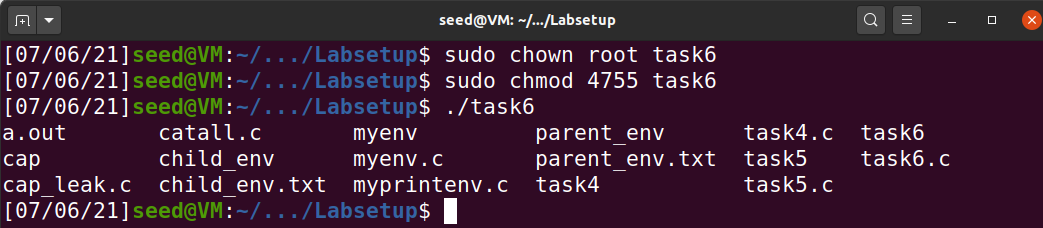
结果截图：

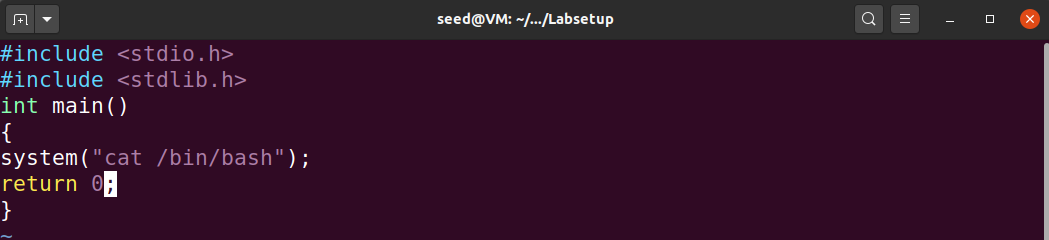


实验结果：在Set-UID程序进程的环境变量中只找到了向用户进程的环境变量中添加的PATH和TASK5，且变量值相同。并没有发现向用户进程的环境变量中添加的LD\_LIBRARY\_PATH环境变量。

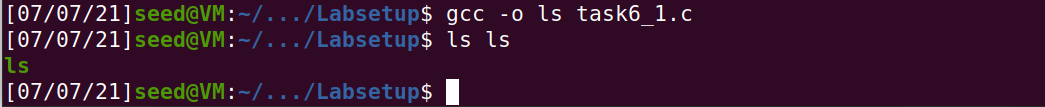
**Task6:PATH环境变量和Set-UID程序**

PATH环境变量未改变时：

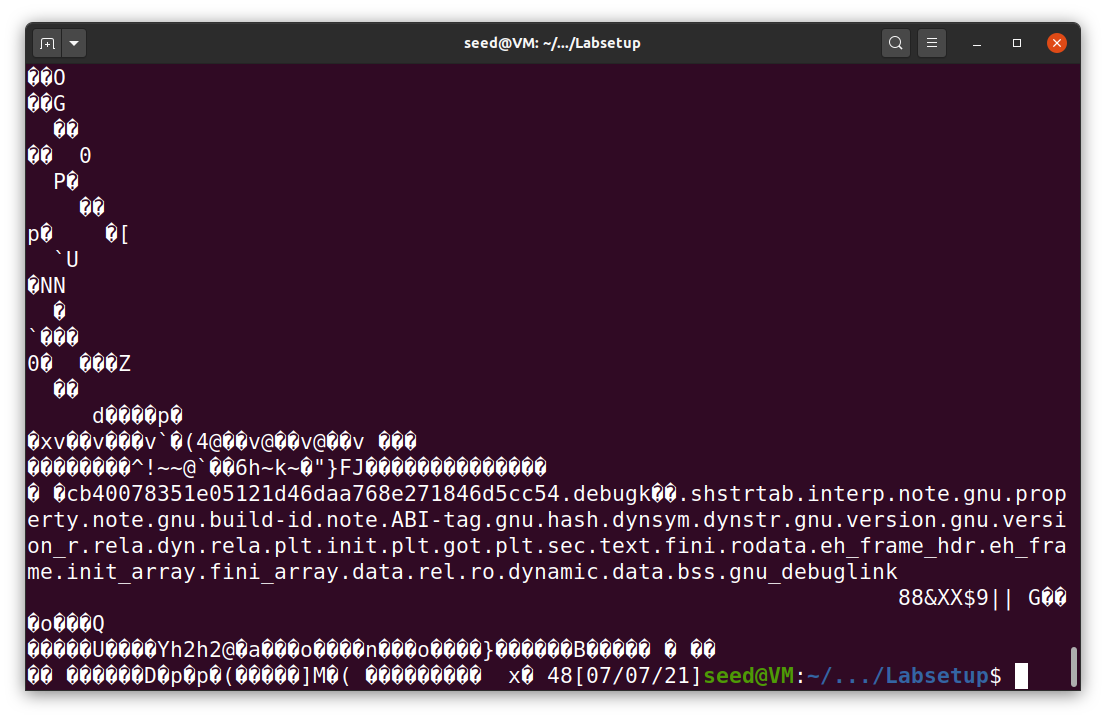


自己编写的ls:  


编译并确认存在ls:



PATH路径改变后：

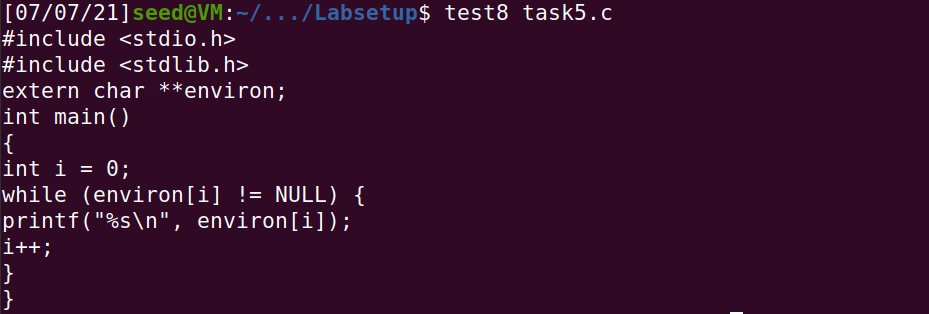


结论：set-uid程序的运行会受到环境变量的影响。system()从PATH变量设置的路径中寻找对应文件。

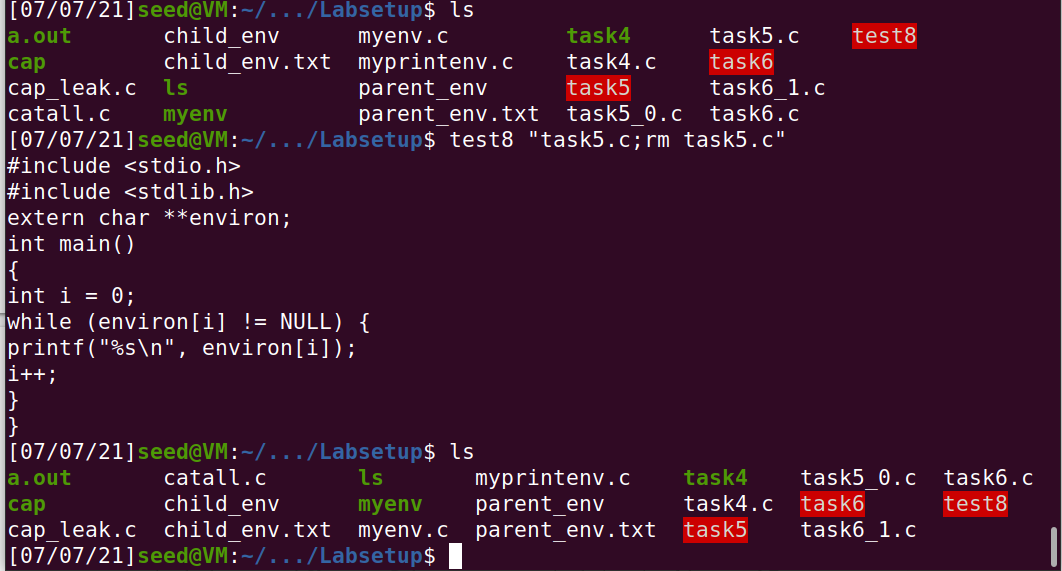
**Task8:使用system（）与execve（）调用外部程序。**

使用system()时：

首先，可以实现显示功能。

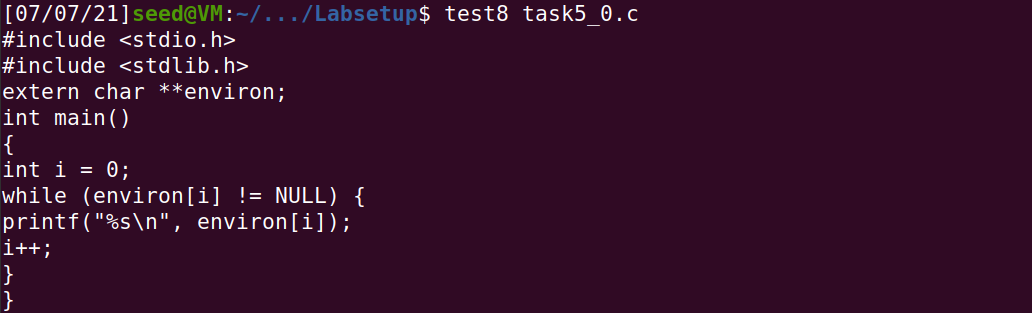


如下图所示，当在输入中加入“rm task5.c”时，就将文件task5.c删除。



使用**execve（）时：**

依然能够实现显示功能。



但是，当在输入中加入“rm task5.c”时，就会报错。



结论：system会将输入按照‘;‘对指令逐个执行，execve会将“”中的所有参数当作一个文件路径，因此不能逐个执行命令。

**实验体会：**

本次实验是第一个用Linux系统进行c语言编程的实验，在编译的过程中遇到不少错误，比如缺少函数的头文件，程序中有中文字符等等。通过解决这些错误，我自主学习，搜集资料的能力得到了提升。本次实验锻炼了我使用linux命令行的能力，学习了一些使用的命令，加深了对进程，环境变量，Set-UID程序等概念的理解。

在实验过程中，有时会出现因为操作不熟练导致实验失败的情况，实验技能与linux命令行的熟练度还需要提高。