SET-UID Programs

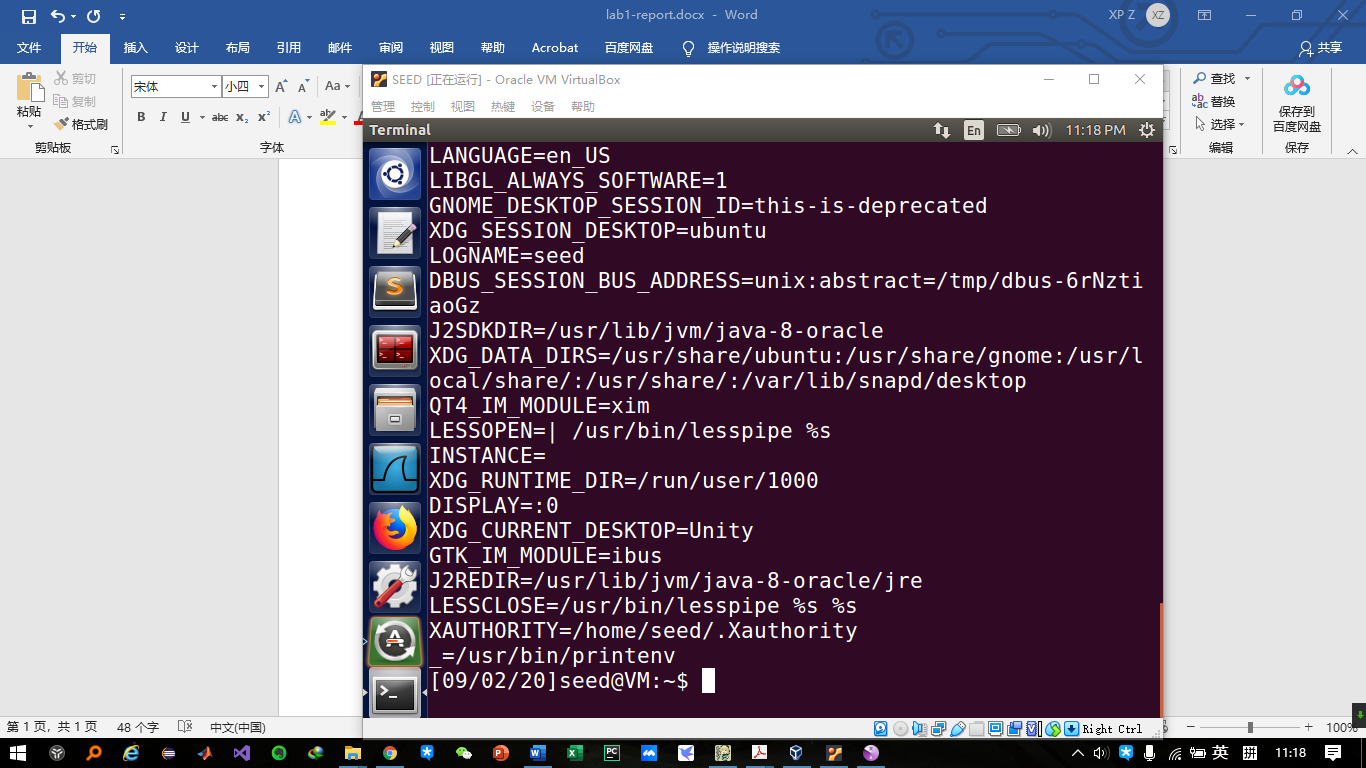
57117131 周煦彭

TASK 1: Manipulating Environment Variables

实验过程：

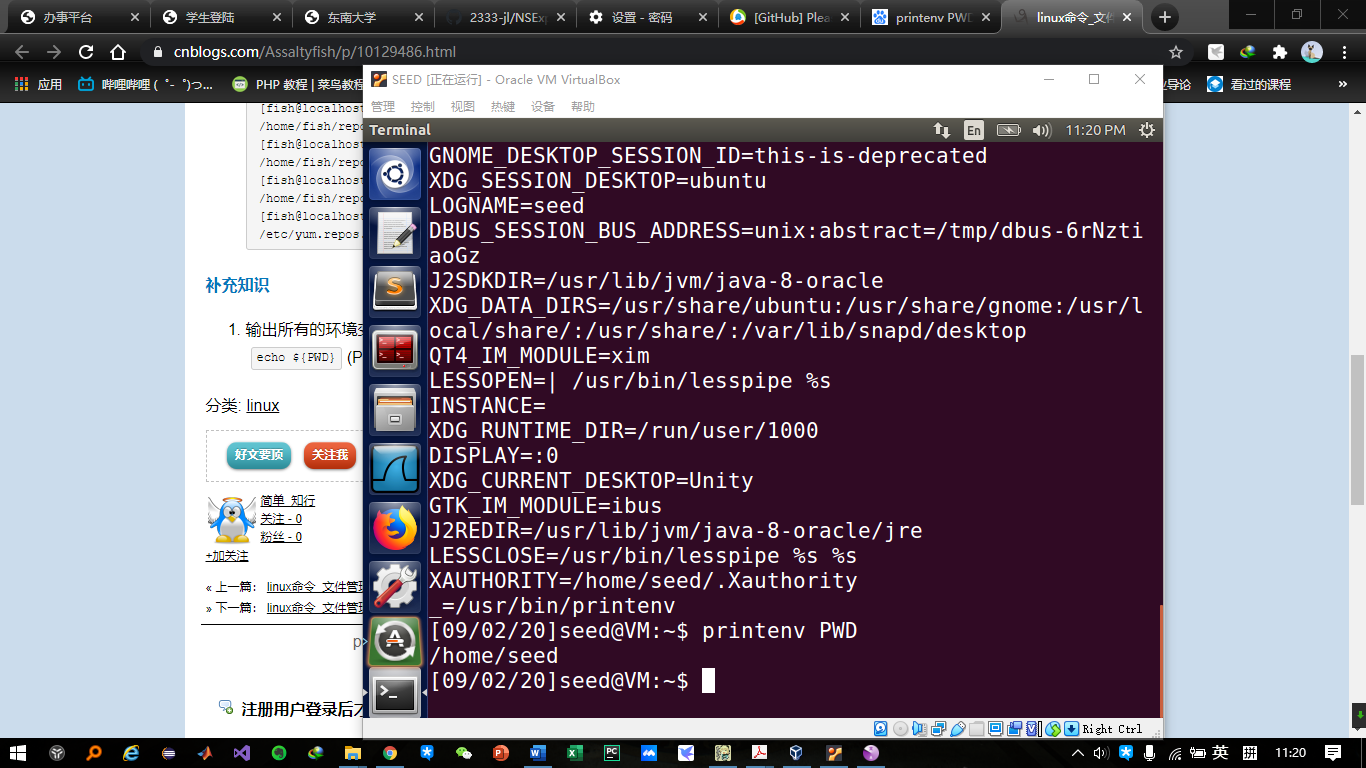
本实验研究设置和取消设置环境变量的命令。

输入“printenv”命令，结果如下



打印出系统的环境变量。

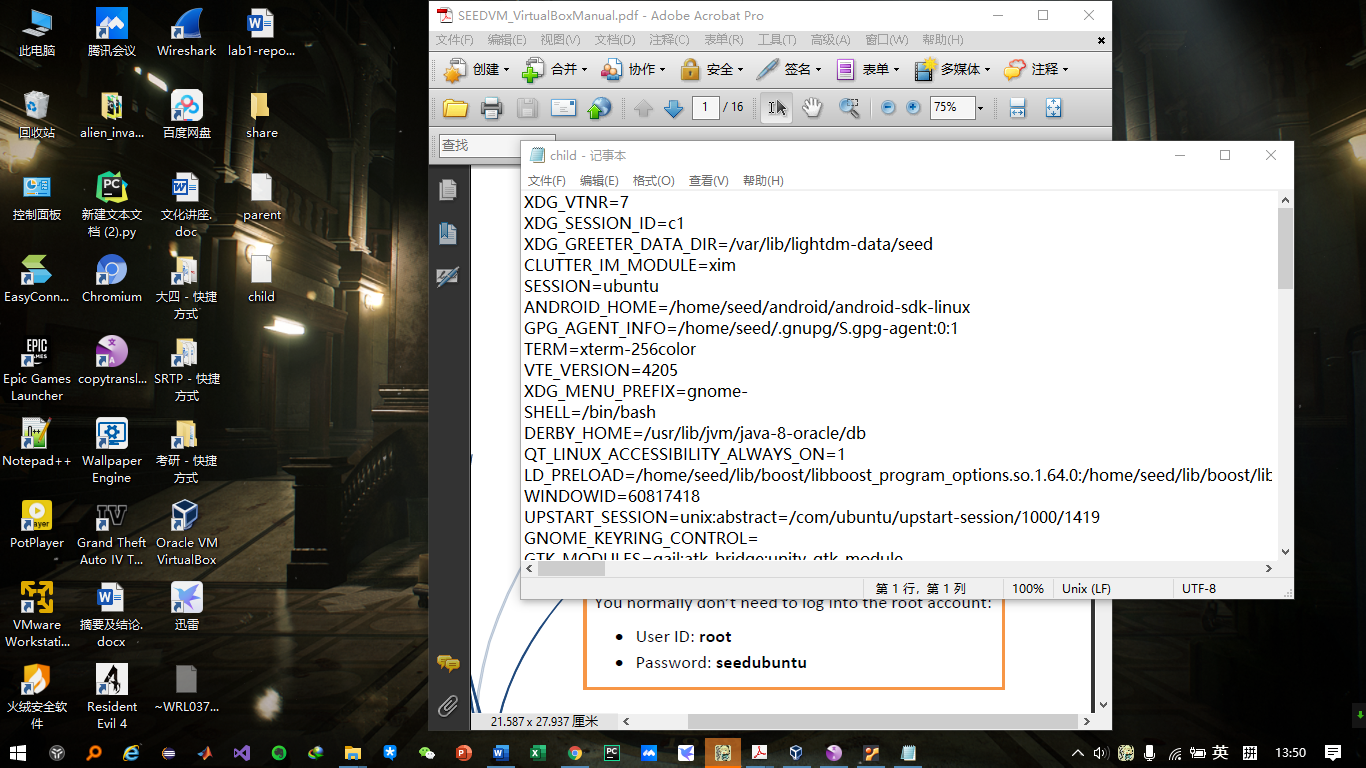
输入“printenv PWD”打印出PWD的环境变量

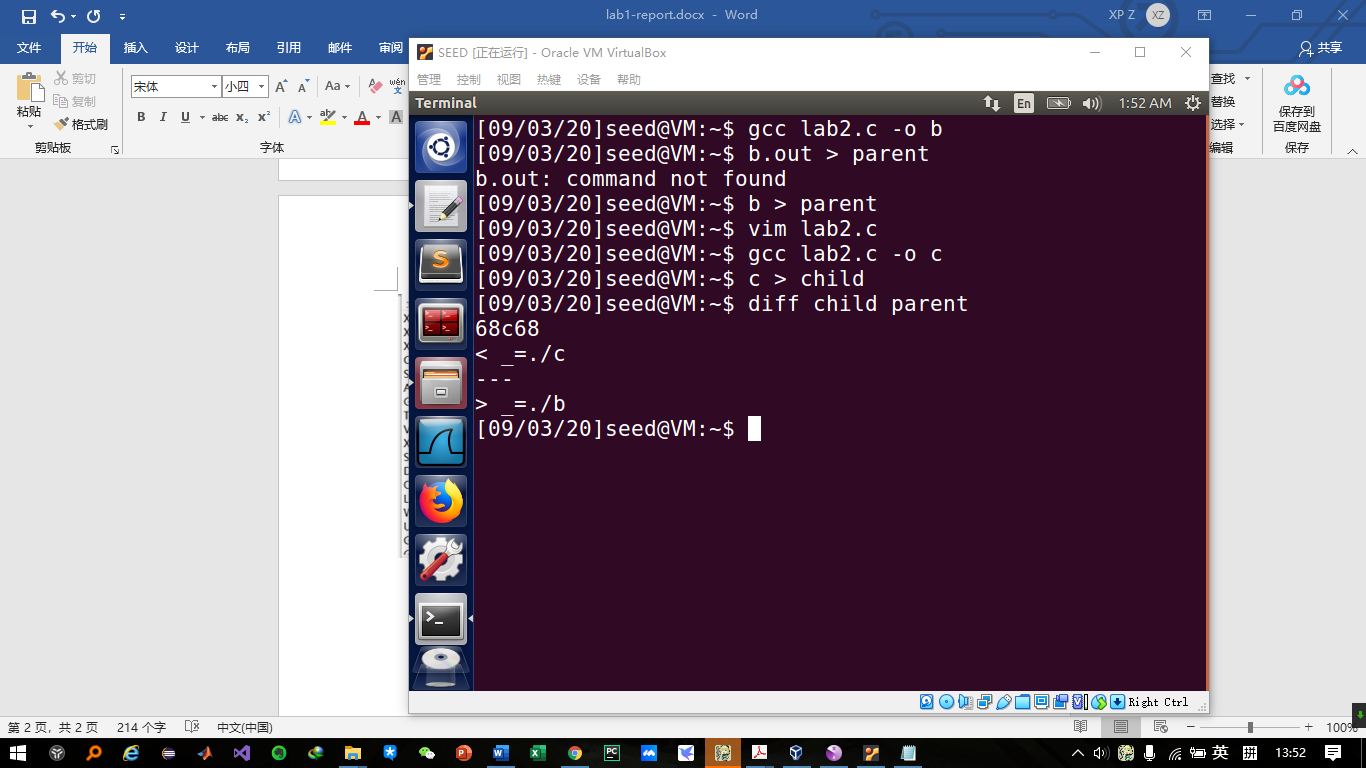


输入“export”列出系统环境变量，unset命令用来删除命令，这里就不加以尝试。本实验是个用来让我们获取相关环境变量并熟悉他们的过程性实验。

TASK2: Passing Environment Variables from Parent Process to Child Process

实验过程：

针对示例的程序，分别输出子进程和父进程的环境变量，得到如下文件

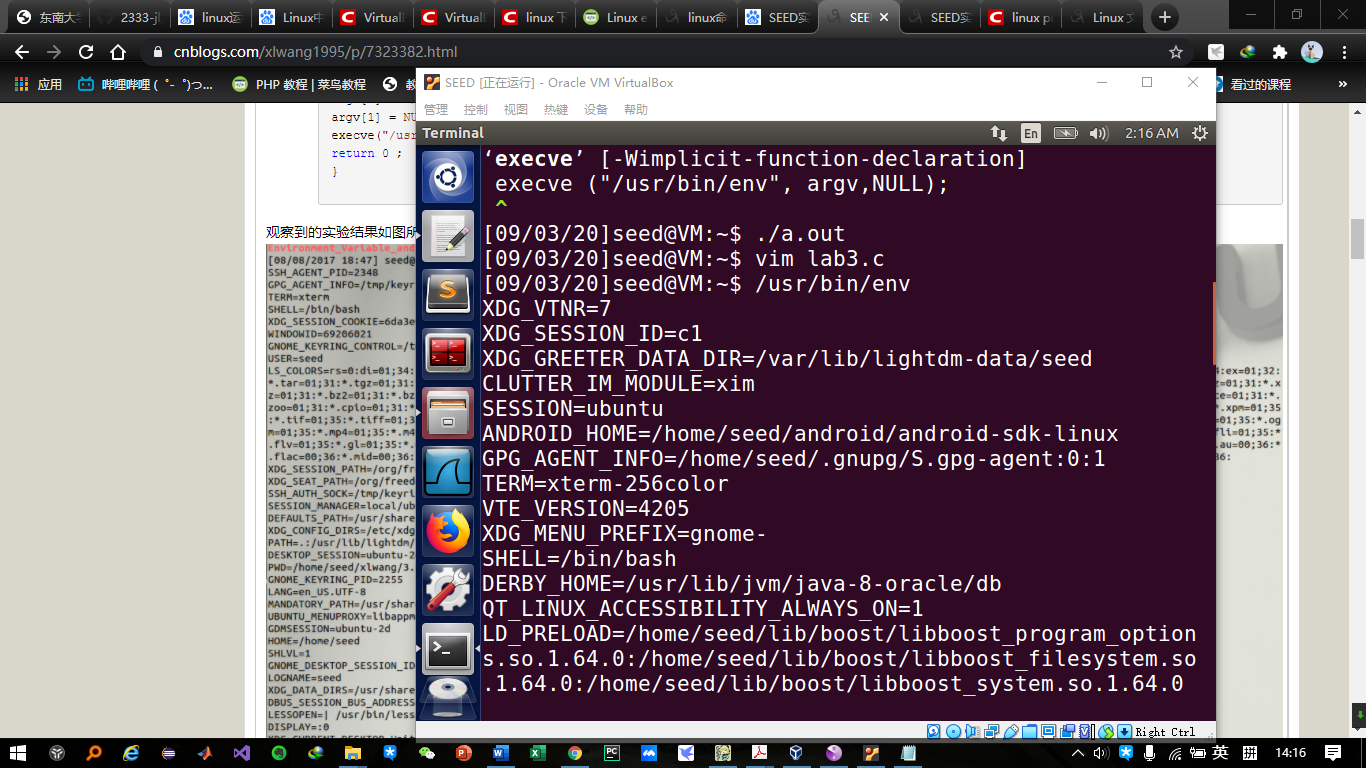
分别针对子进程和父进程生成的两组文件用diff命令比对，发现没有差别

这两个文件输出的环境变量完全相同。说明原环境变量被子进程完全继承。通过man fork，对fork函数做了进一步了解。fork函数通过系统调用创建一个与原来进程几乎完全相同的进程，子进程自父进程继承了进程的资格，环境，堆栈与内存根目录等；但是子进程没有继承父进程的某些特性，比如父进程号，文件描述符，资源使用等。

TASK3: Environment Variables and execve()

实验过程：

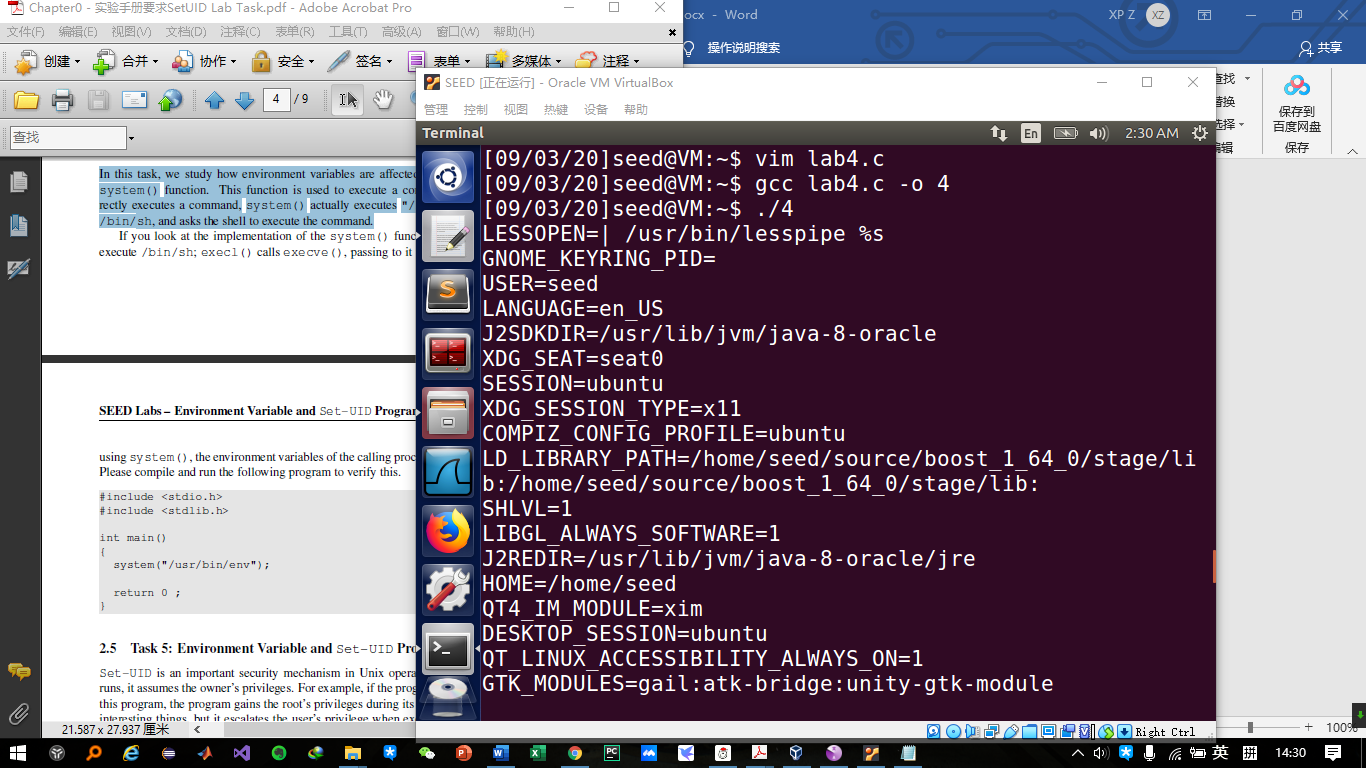
该程序简单地调用了/usr/bin/env,该系统调用能够打印出当前进程的环境变量。执行结果如下9



然后改变execve()函数的参数，execve()用来执行路径/usr/bin/env的环境，第二个参数是利用数组指针来传递给执行文件，并且需要以空指针(NULL)结束，最后一个参数则为传递给执行文件的新环境变量数组。

TASK4:

编译程序后，结果如下：



由于system函数定义为int system（const char \* string），该函数调用/bin/sh来执行参数指定的命令,/bin/sh一般是一个软连接，指向某个具体的shell，比如bash,-c 选项是告诉shell从字符串command中读取命令；在该command执行期间，SIGCHLD信号会被暂时搁置，SIGINT和SIGQUIT则会被忽略，意思是进程收到这两个信号后没有任何动作。system（）函数的函数返回值有些复杂。为了更好地理解system（）函数的返回值，需要了解其执行过程，实际上system（）函数执行了三步操作：

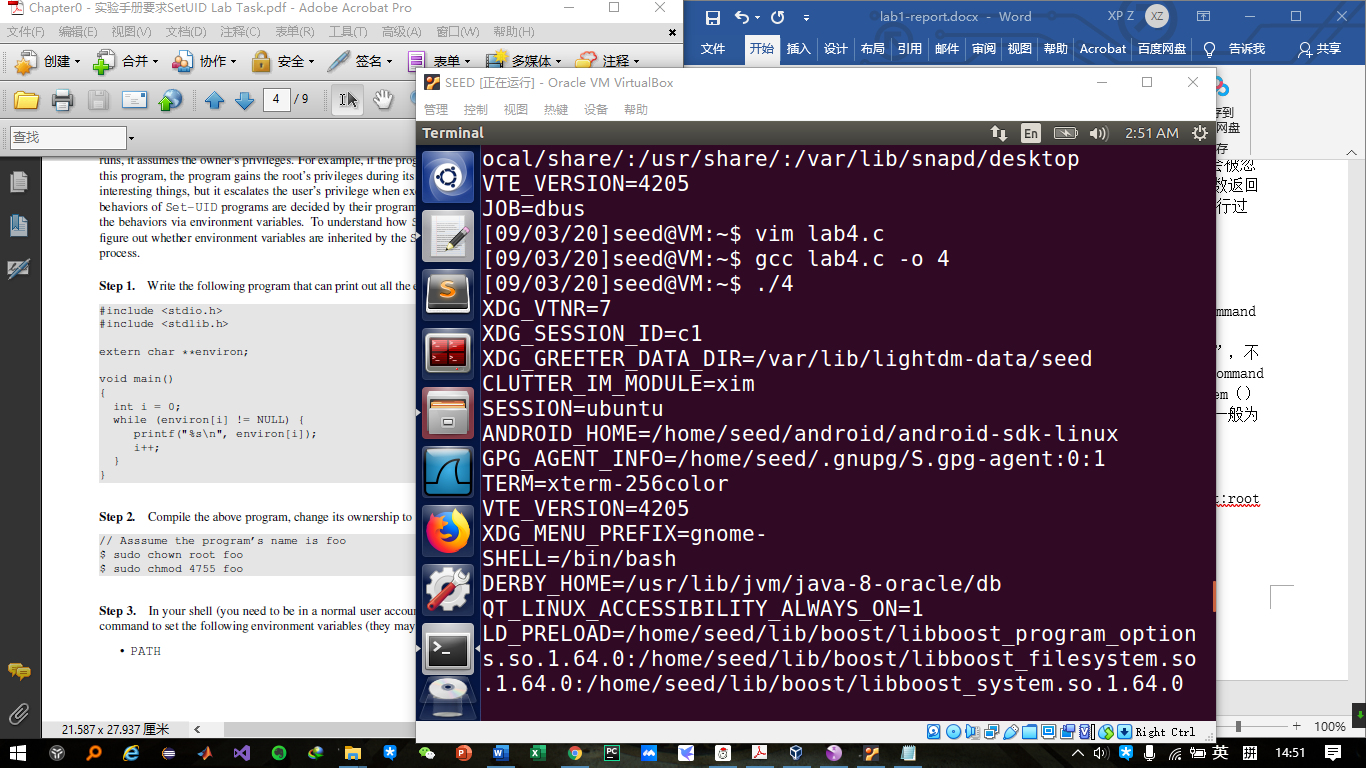
1 fork一个子进程；

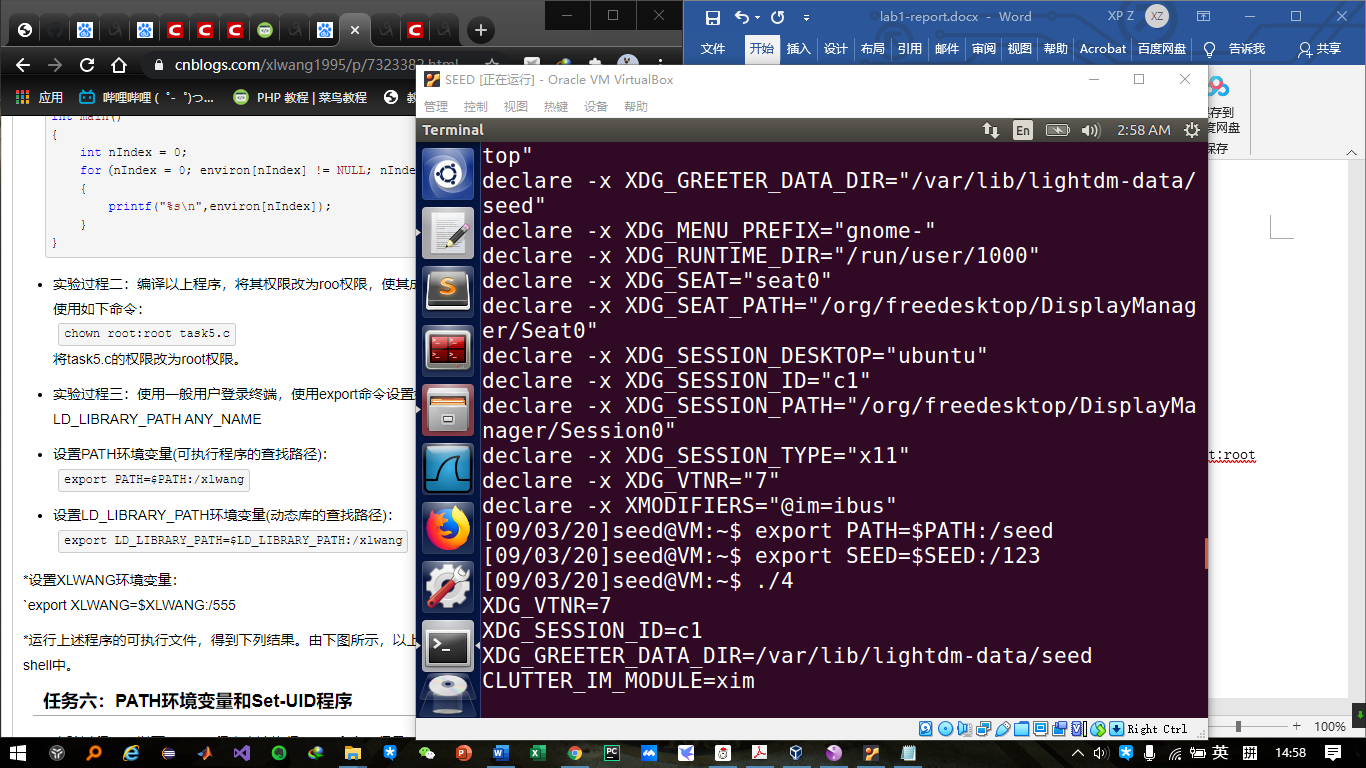
2 在子进程中调用exec函数去执行command；

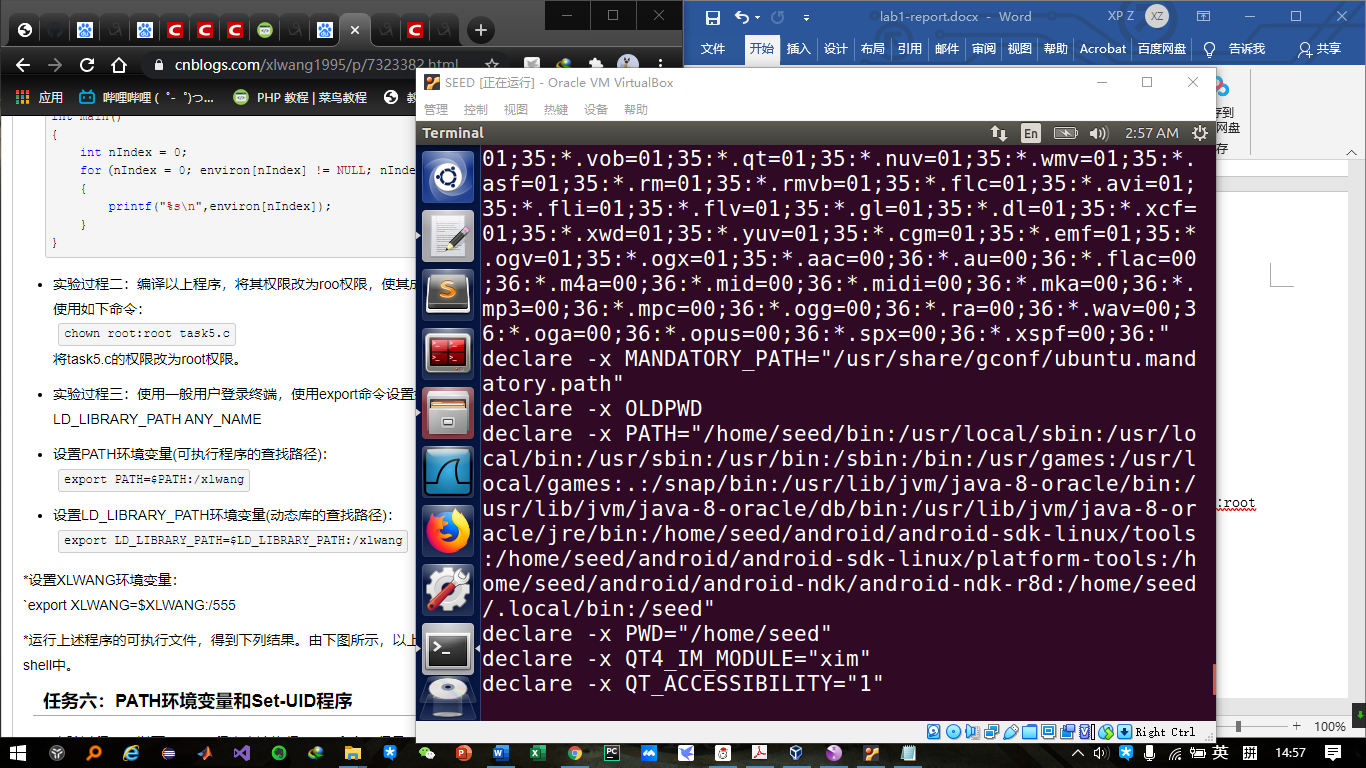
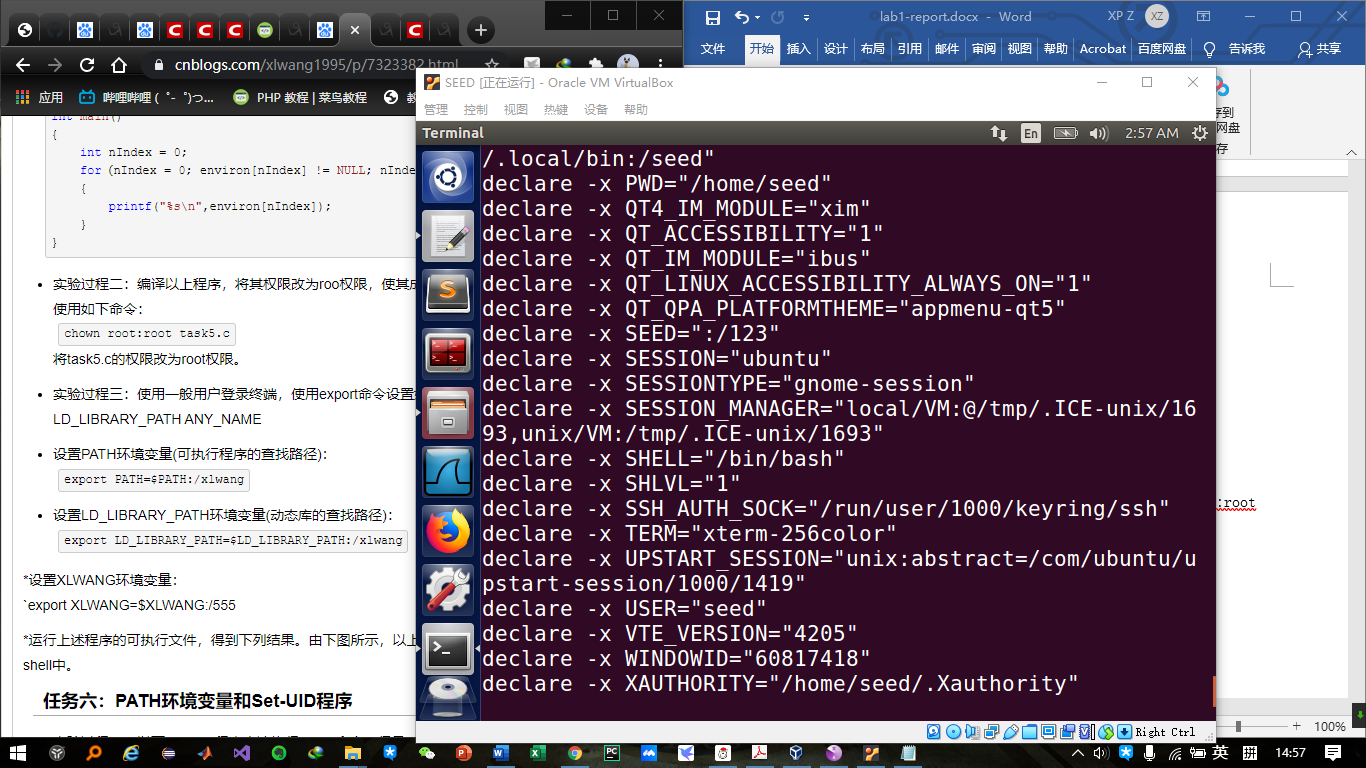
3 在父进程中调用wait去等待子进程结束。

若fork失败，system（）函数返回-1。如果exec执行成功，也即command顺利执行完毕，则返回command通过exit或return返回的值。（注意，command顺利执行不代表执行成功，例如command：“rm debuglog.txt”，不管文件存不存在该command都顺利执行了）如果exec执行失败，也即command没有顺利执行，比如信号被中断，或者command命令根本不存在，system（）函数返回127，如果command为NULL，则system（）函数返回值非0，一般为1。

TASK5:

编译可以输出目前进程所有环境变量的程序,并利用命令“chown root:root lab5.c”将其修改为root权限。

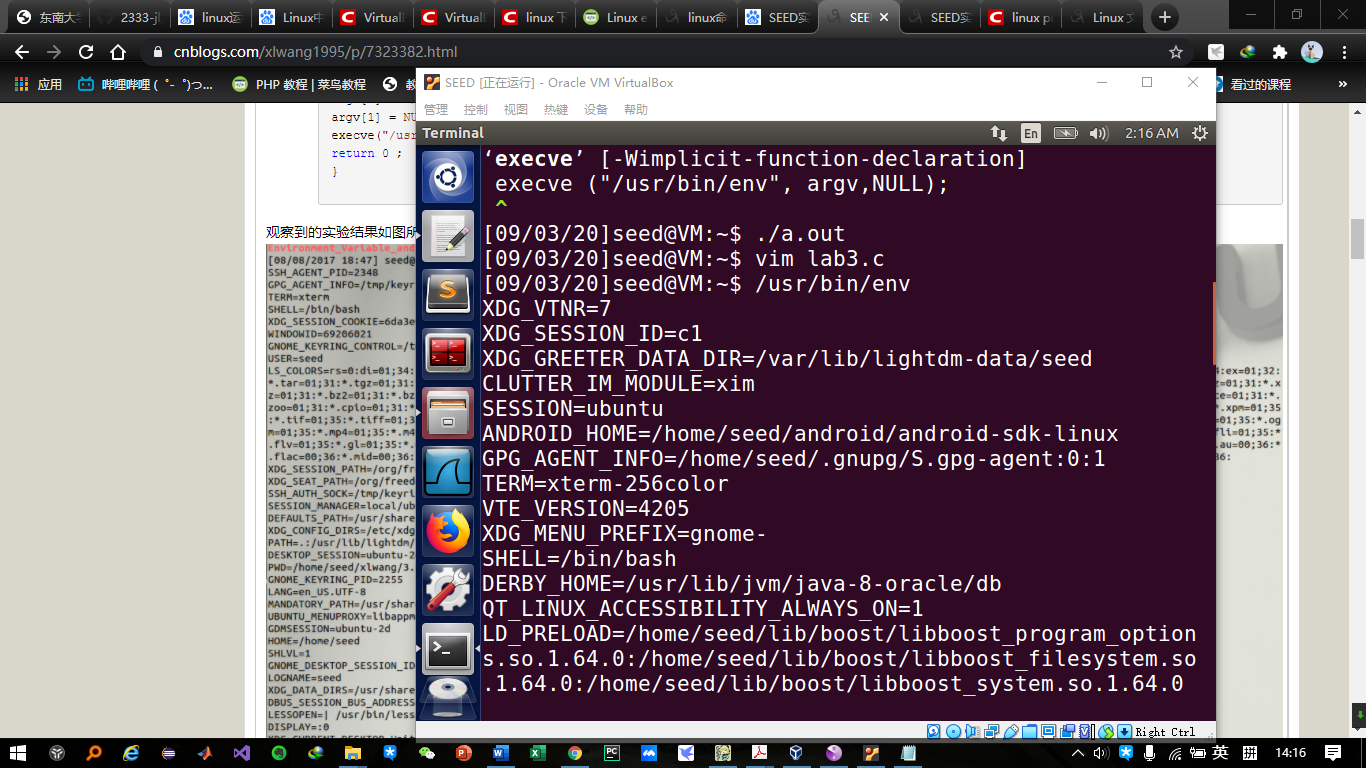
然后export插入环境变量

运行程序，得到如下

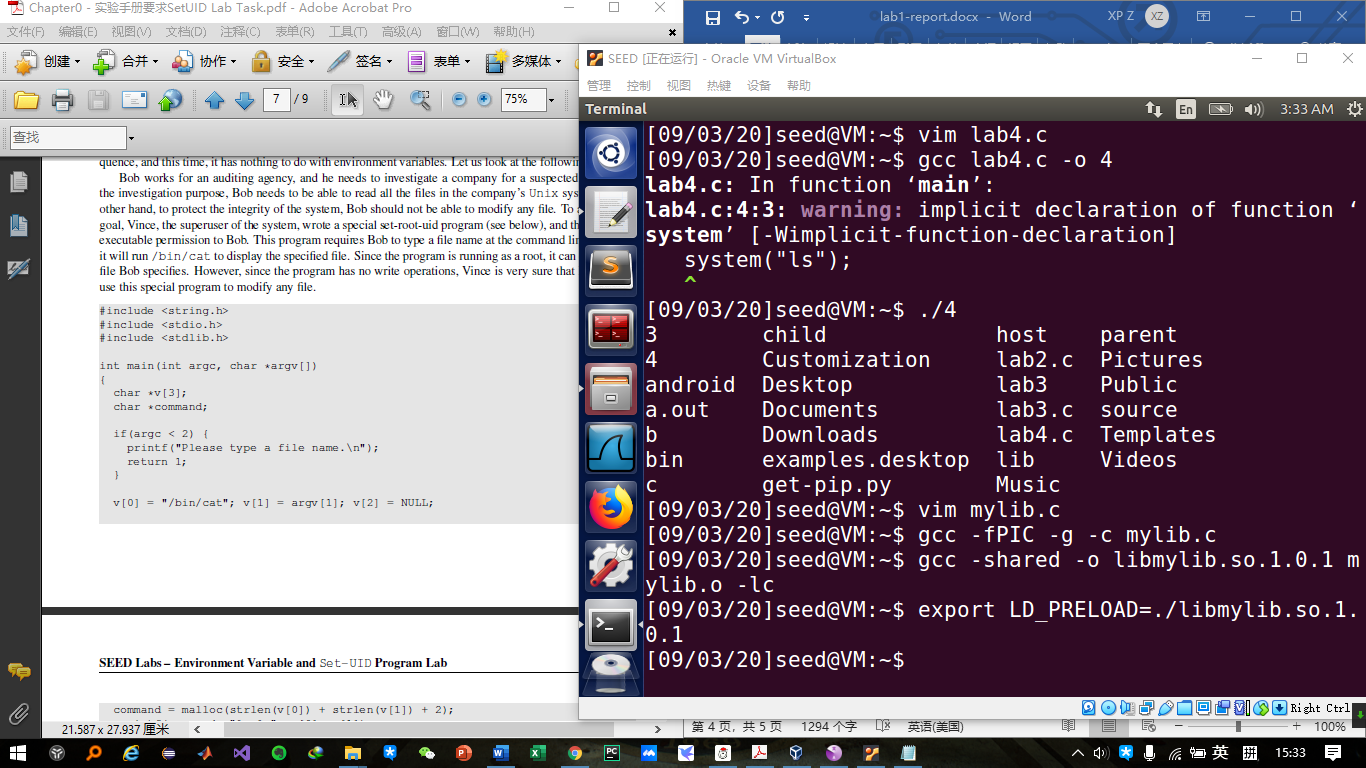
由图可见，几个新定义的环境变量均在shell中。

TESK6:

执行ls命令，system()函数是调用了shell环境变量并且将自己的可执行文件夹所在的目录加在了shell环境变量的头部。且发现SHELL=/bin/bash。运行export SHELL=/seed/:$SHELL

得到成功输出环境变量。

TESK7:

编辑程序如下：

当以普通用户的身份运行myprog程序时，输出：I am not sleeping!

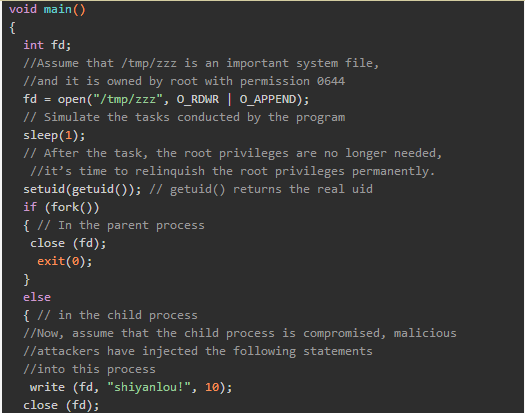
以普通用户身份运行拥有SET-UID root权限的myprog程序时，无输出。

非root用户设置LD\_PRELOAD环境变量并运行myprog程序时，输出：I am not sleeping! 导致他们不同的原因就在于LD\_PRELOAD环境变量。LD\_PRELOAD环境变量是Unix动态链接库的世界中的一个环境变量，它可以影响程序的运行时的链接，它允许你定义在程序运行前优先加载的动态链接库。这个功能主要是用来有选择性的载入不同动态链接库中的相同函数。在该实验中，mylib.c通过sleep函数，生成了一个libmylib.so.1.0.1链接库。然后将该链接库添加到LD\_PRELOAD环境变量上。比较这三次实验，第一次和第三次实验myprog程序均具有seed用户权限，而在seed用户的LD\_PRELOAD环境变量中也添加了该链接库。

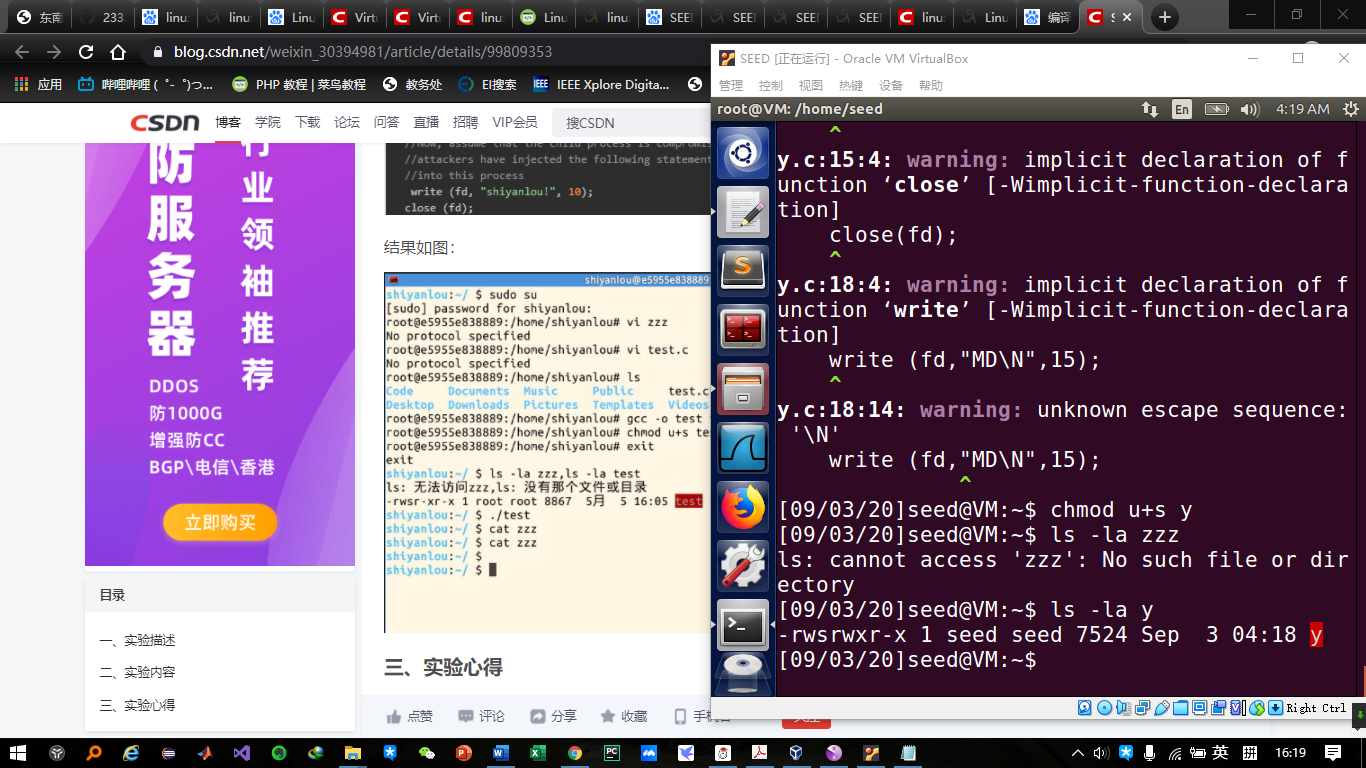
TEST8:

编译程序，赋予其root权限，将其变为SET-UID程序，该程序将会使用system()来调用命令。如果将示例程序中的v[0]=‘/bin/cat’改为v[0]=’rm’就会删除该命令。执行可执行程序，就会发现lab8.c也就是该文件已经被删除。也就是该lab8.c文件变成SET-UID文件时会被删除。因此SET-UID程序的不稳定性会给linux系统带来很多麻烦。

Test9：

编译以下程序，将其所有者更改为root，并将其设置为Set-UID程序。以普通用户身份运行程序，为了更加安全，Set-UID程序通常会调用setuid()系统调用函数永久的清除它们的root权限。然而有些时候，这样做是远远不够的。在root用户下，在/tmp目录新建一个空文件zzz。在root用户下将下面代码命名为test.c，放在/tmp目录下，编译这个程序，给这个程序设置root权限。

结果如下：

zzz发生了更改。