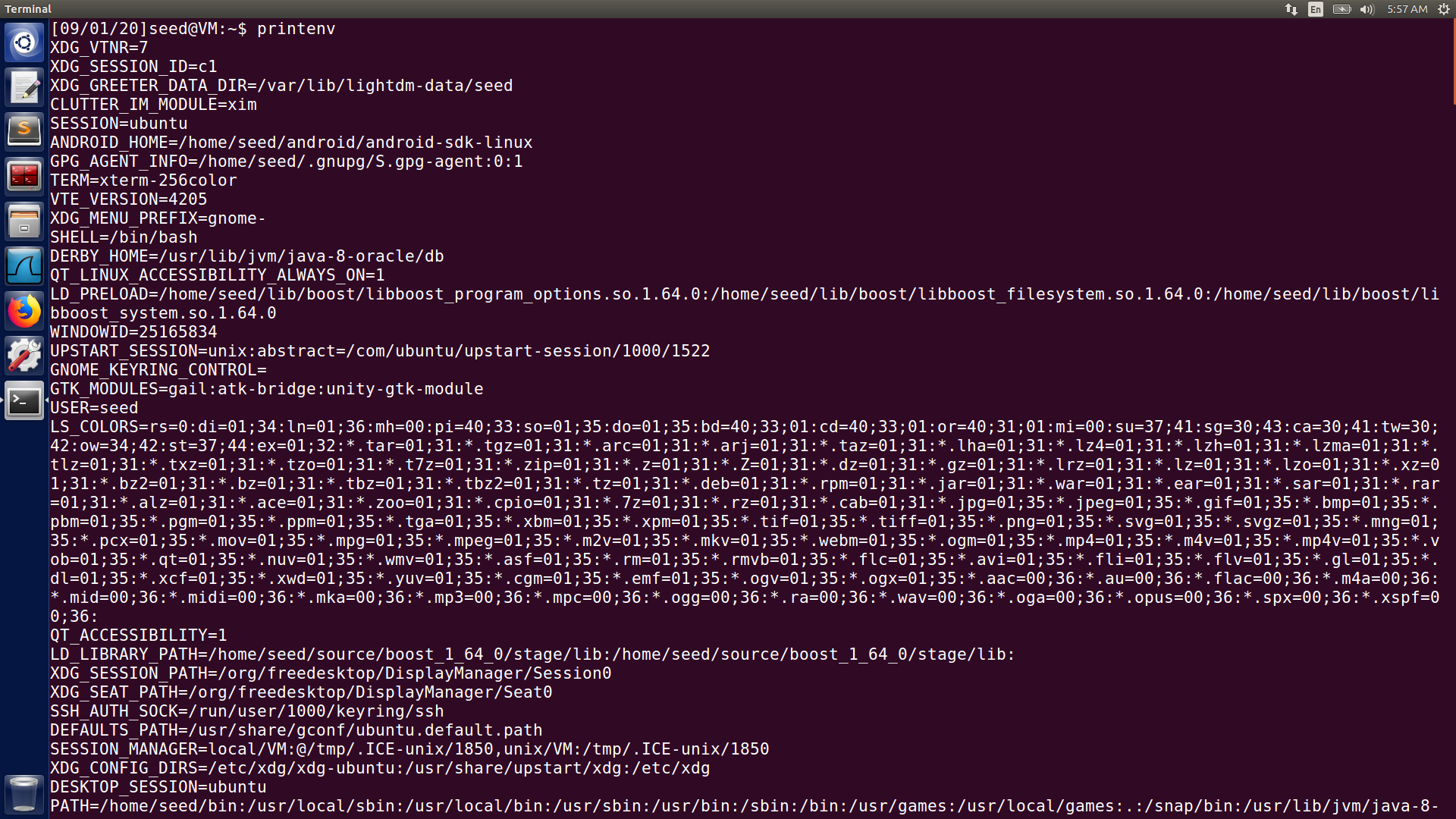
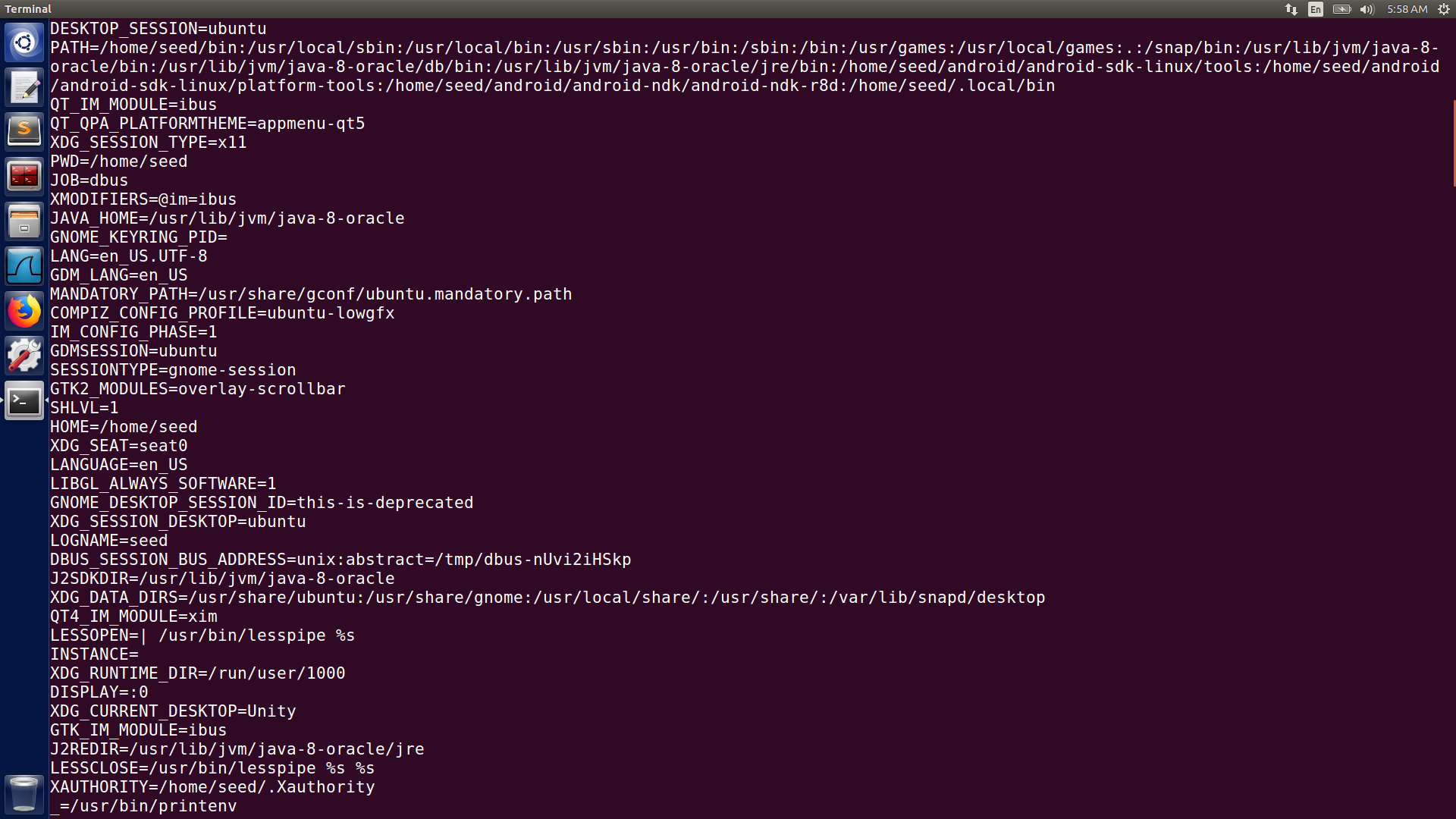
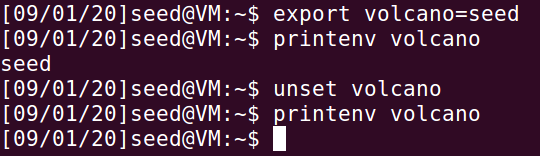
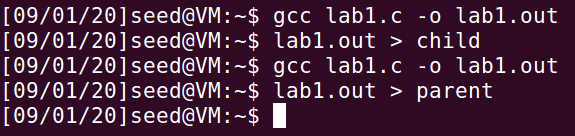
**Task 1**

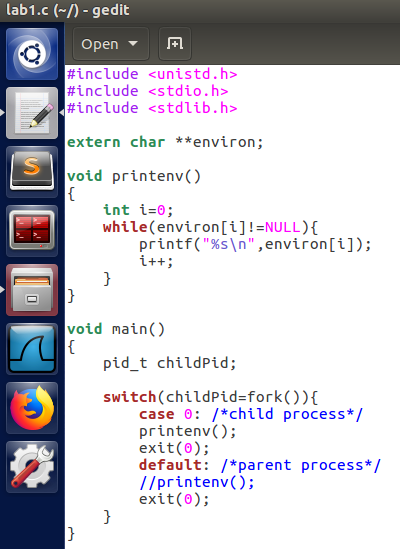


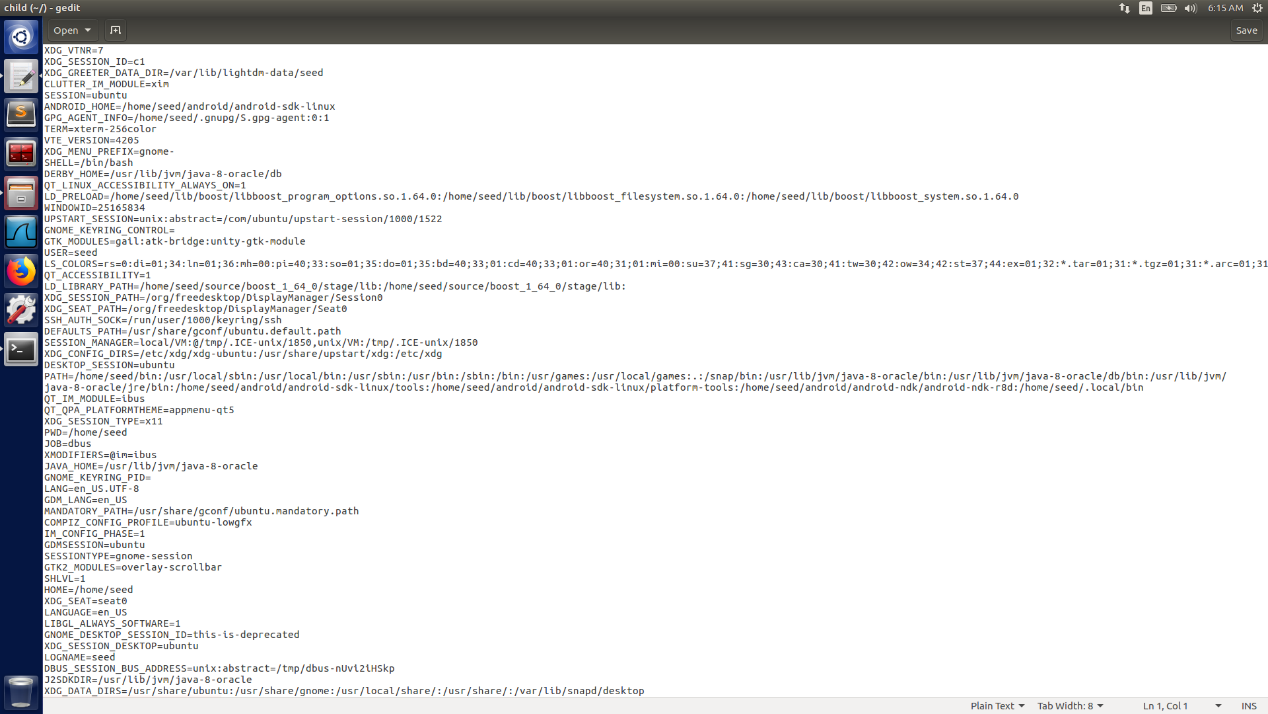


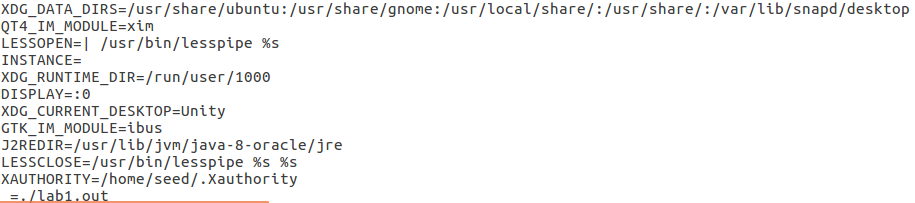


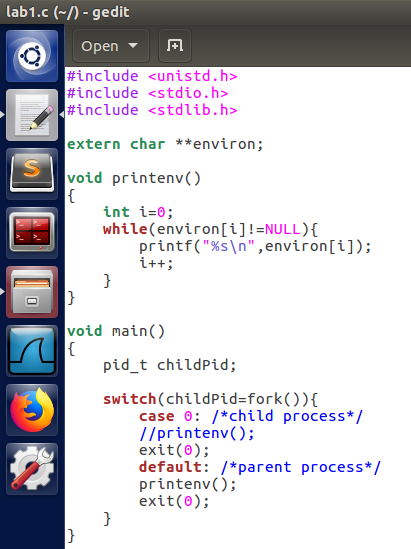
**Task 2**

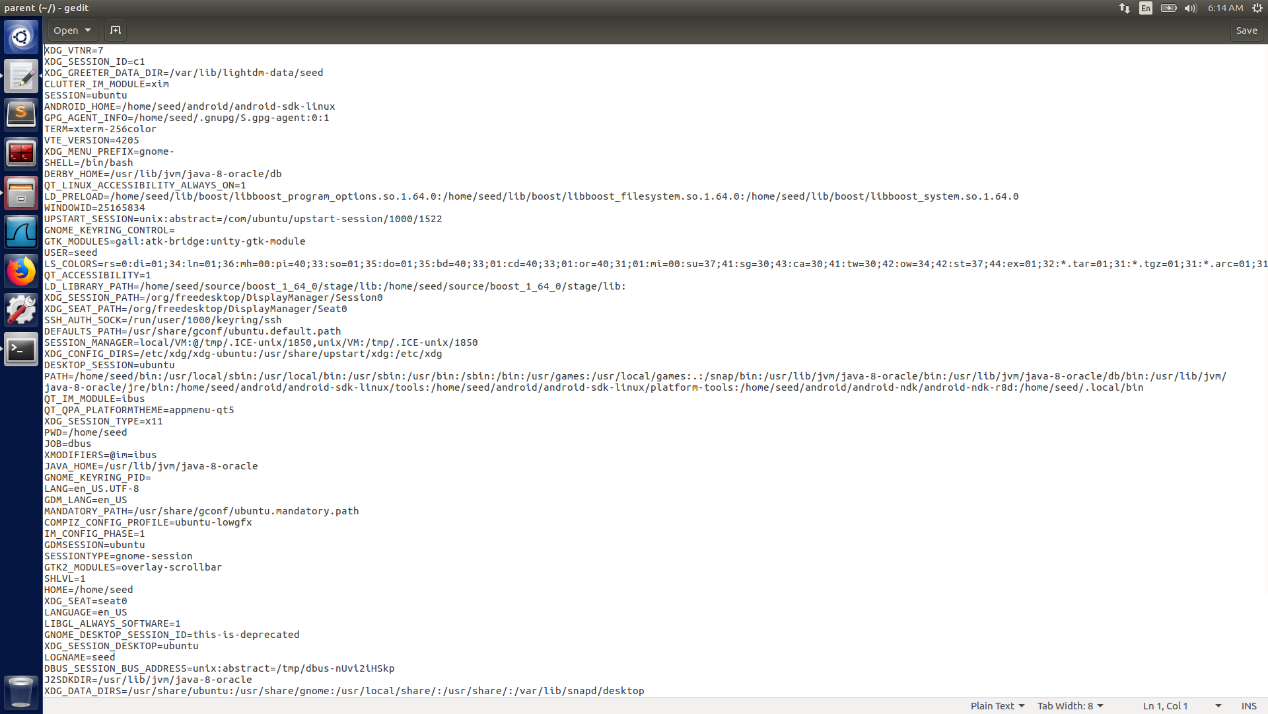


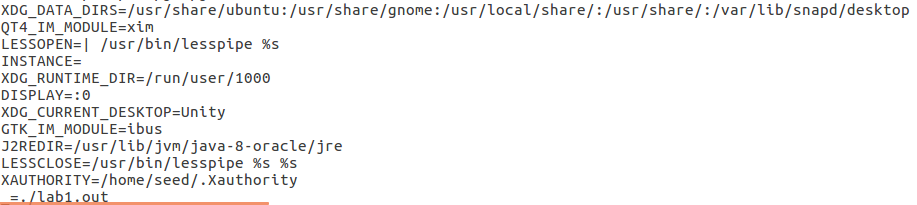








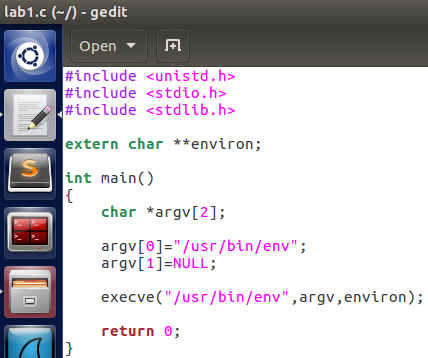
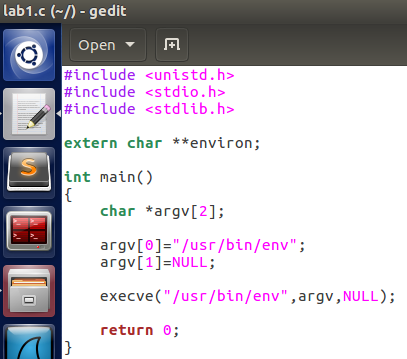


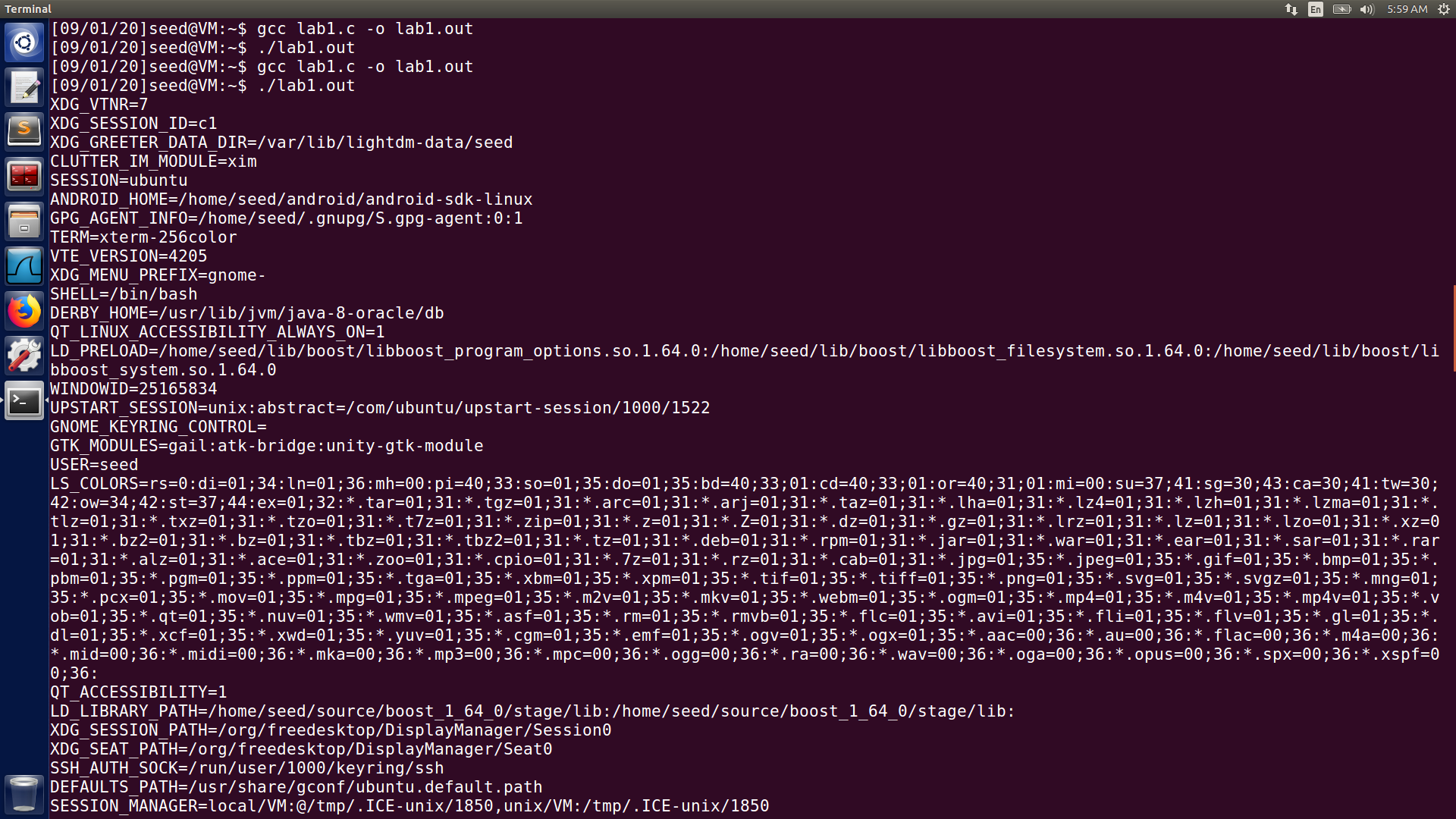


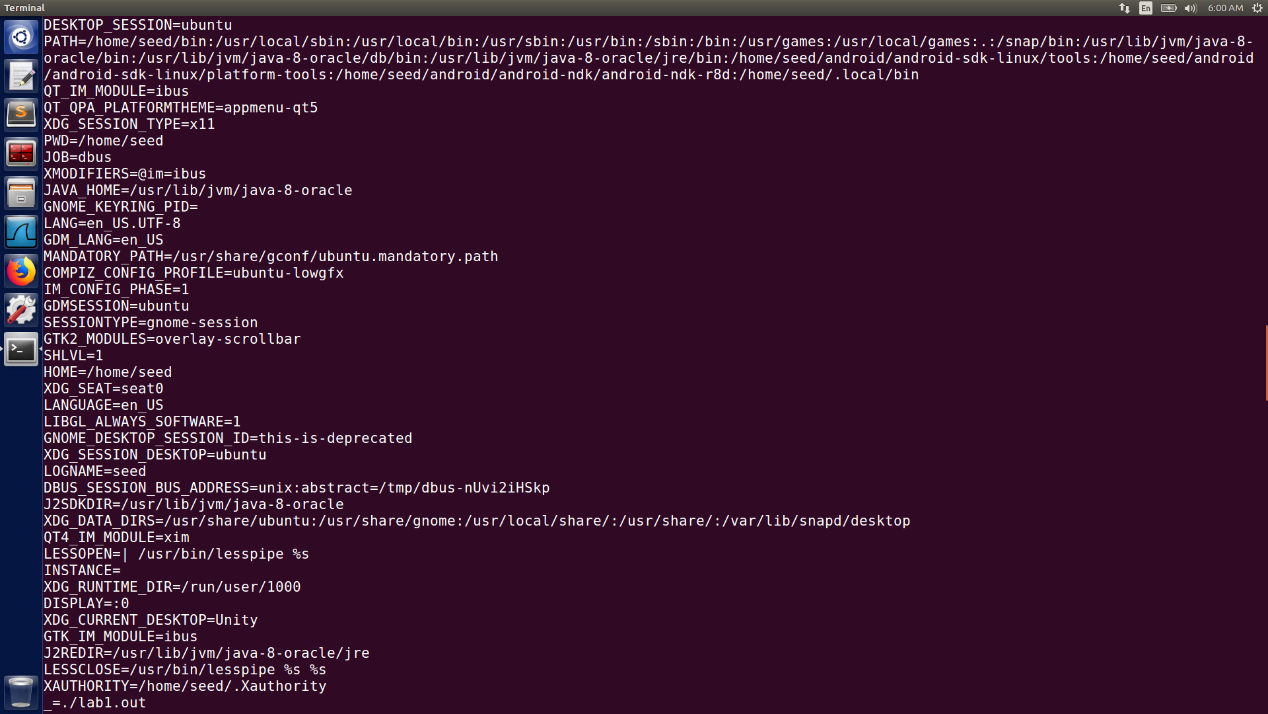


根据实验结果可知，子进程和父进程的运行结果是相同的，因此父进程的环境变量是会被子进程所继承的。

**Task 3**

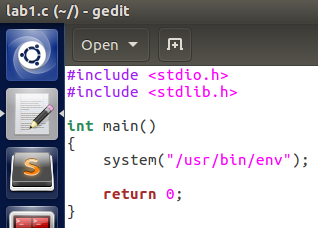


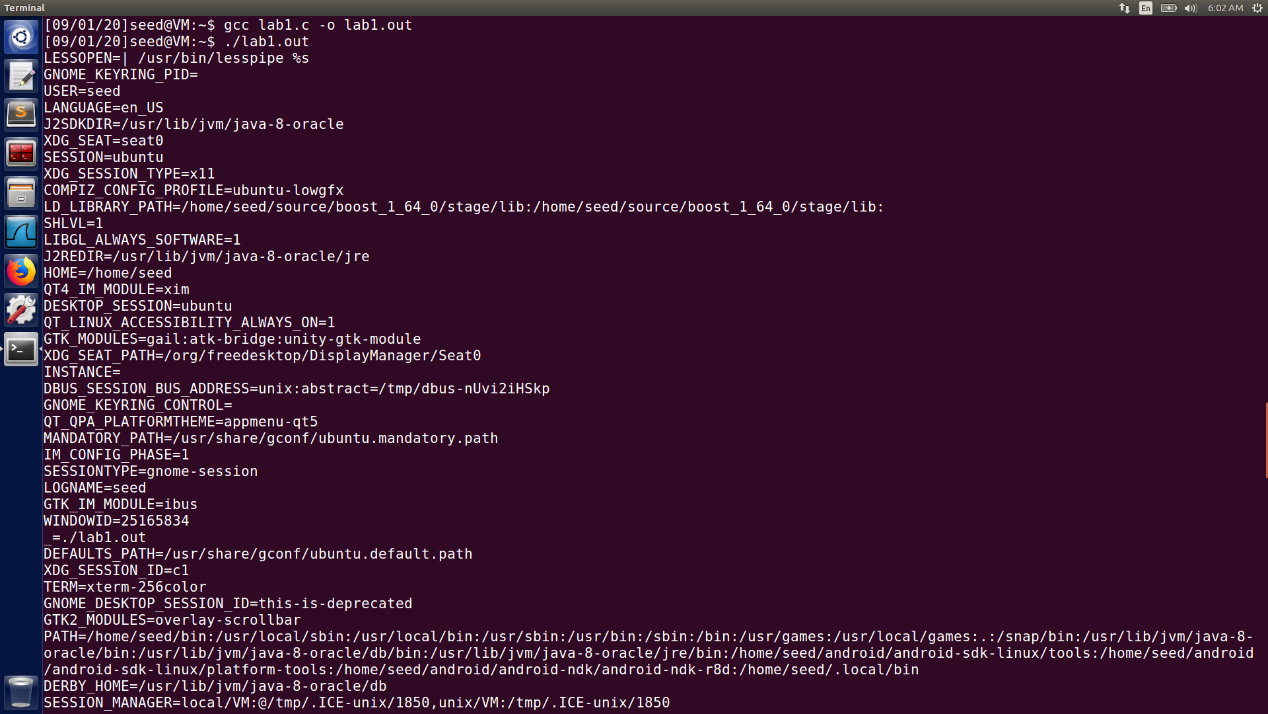


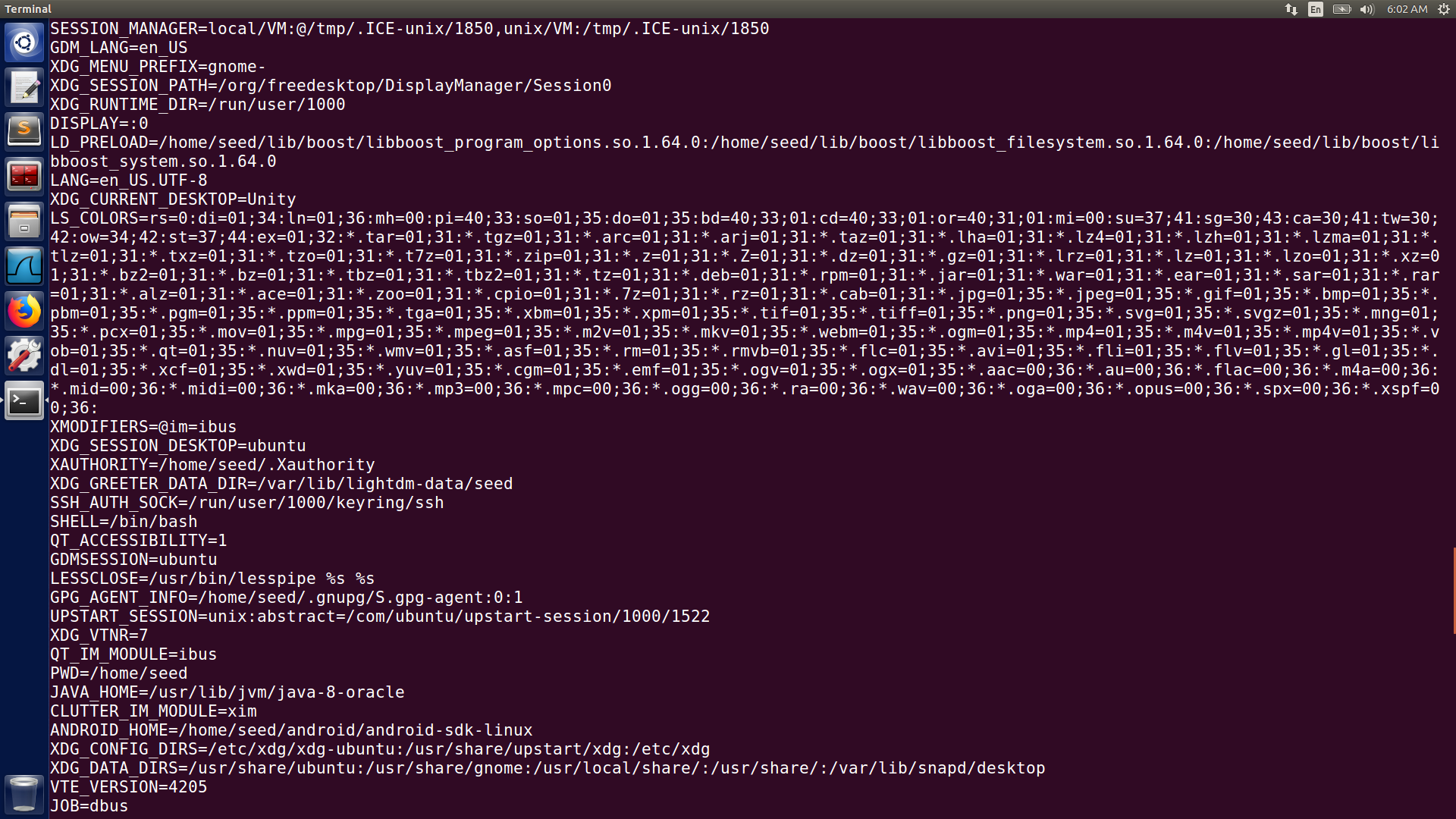


在Step 1中，输出的结果为空；而在Step 2中，输出的结果如上图。根据实验结果可知，新程序获取环境变量是通过execve()调用/usr/bin/env程序，再利用环境变量数组将其传给新程序。

**Task 4**

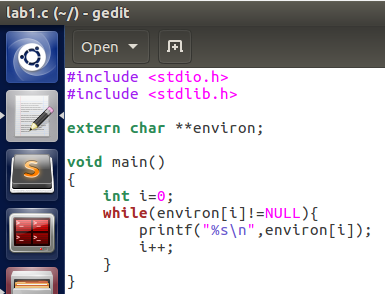


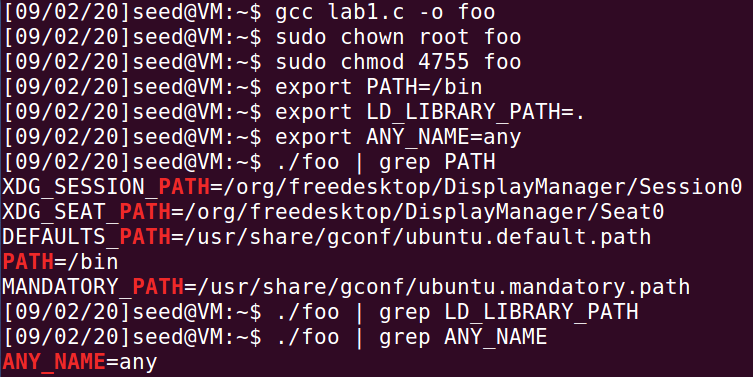




根据实验结果可知，system()可以传递所调用函数的环境变量。

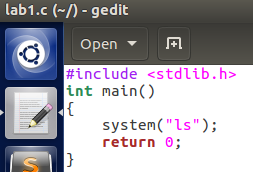
**Task 5**

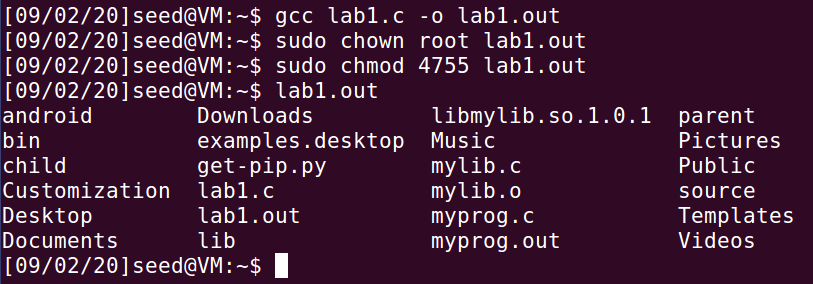


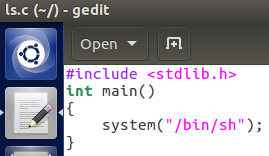


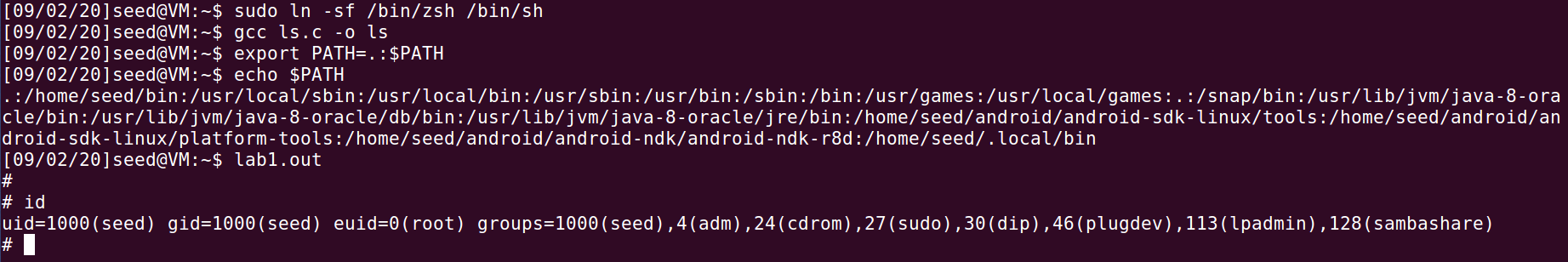
根据实验结果可知，在shell进程中设置的环境变量都会在Set-UID子进程中，即Set-UID进程从用户的进程继承了环境变量。

**Task 6**



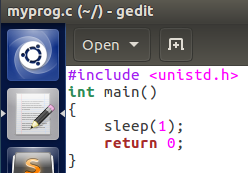
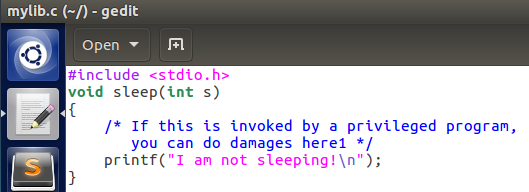


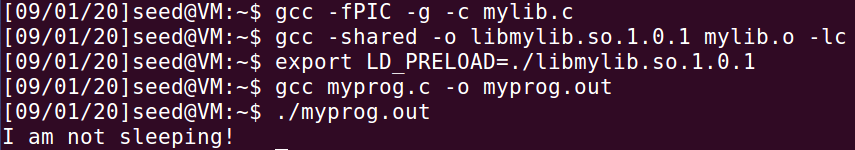


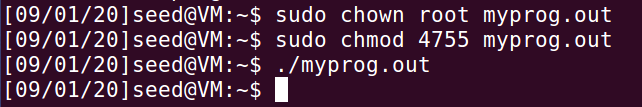


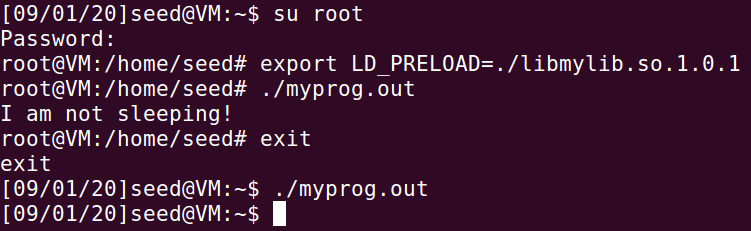
根据实验结果可知，此Set-UID程序可以运行其他代码而不是/bin/ls，并且是以root权限运行。我们可以通过编译创建ls同名的可执行文件，然后将此ls所在目录添加到PATH，再运行Set-UID程序会优先执行此ls文件即可实现运行其他代码，当ls同名的可执行文件中有/bin/sh指令时，则能获得root权限。

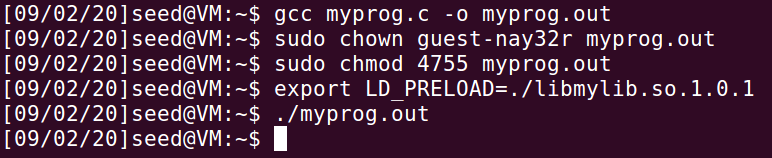
**Task 7**



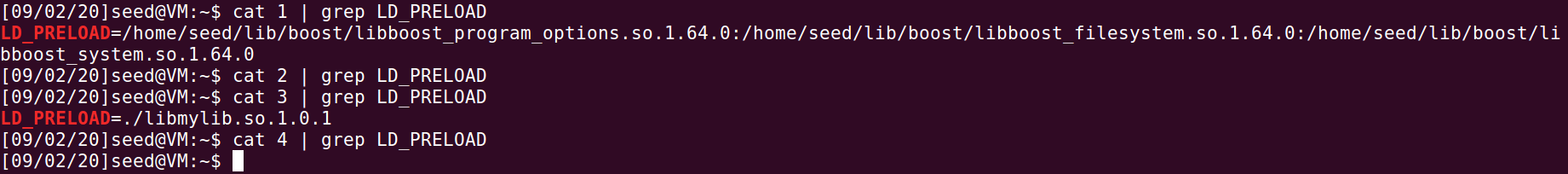
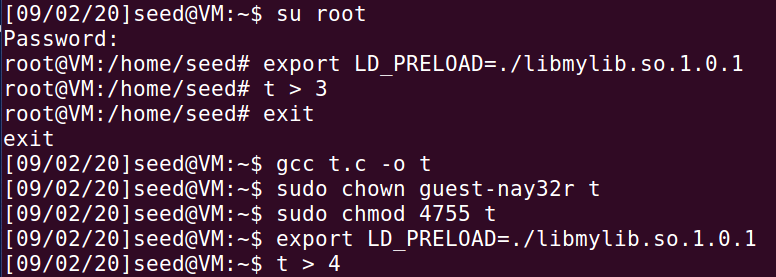
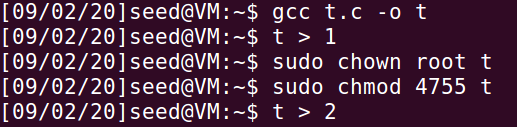
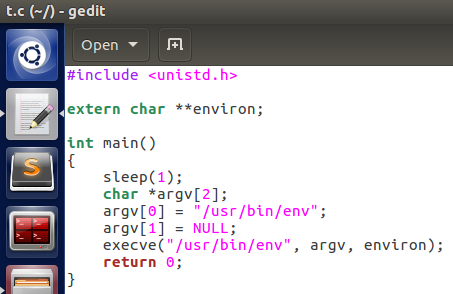






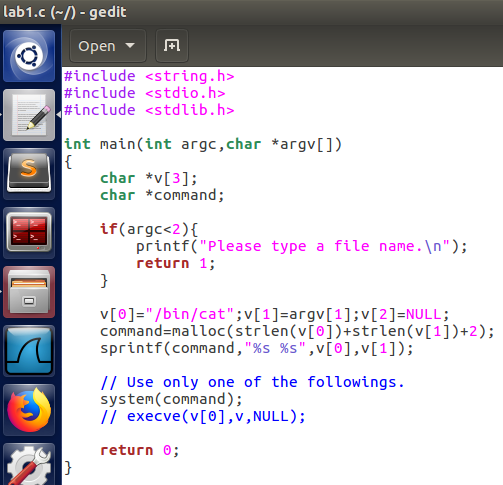


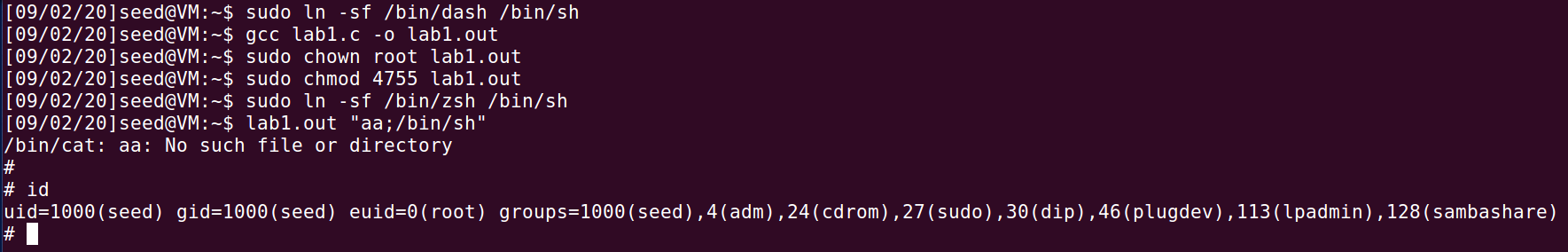
在Step 2中，myprog作为常规程序被普通用户运行时，会使用重载的sleep函数；作为Set-UID的root程序被普通用户运行时，不会使用重载的sleep函数；作为Set-UID的root程序被root用户再次导入LD\_PRELOAD环境变量和运行时，会使用重载的sleep函数；作为Set-UID的用户程序被其他非root用户再次导入LD\_PRELOAD环境变量和运行时，不会使用重载的sleep函数。

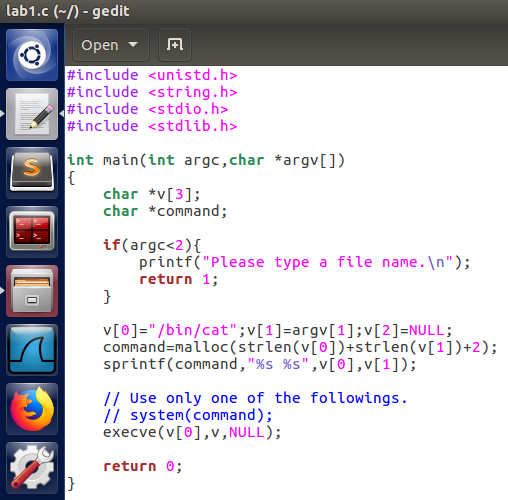


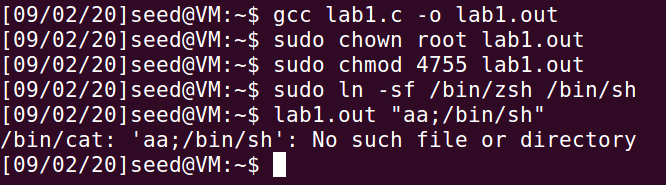
在执行文件中增加显示环境变量的函数并输出到文件中，再利用指令查看环境变量是否有LD\_PRELOAD。根据实验结果可知，原因是只有在运行自己拥有的程序的时候才会使用LD\_PRELOAD环境变量和重载的sleep函数，否则会忽略，所以导致了Step 2中不同的实验结果。

**Task 8**





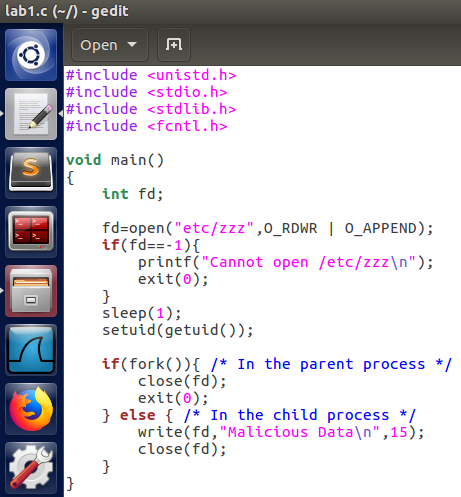


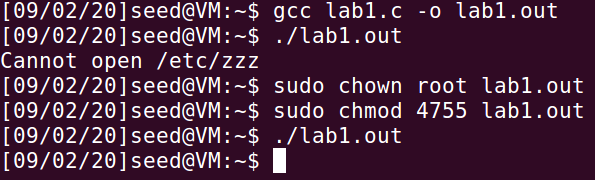


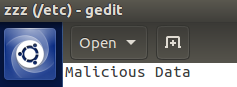
根据实验结果可知，在Step 1中使用system()时，Bob可以损坏系统的完整性，操作方法是在输入的数据中加入指令名，system()在运行时会直接执行该指令，输入的数据被变成指令进而获得权限；而在Step 2中使用execve()则不会有这种风险，指令和数据被划分清楚，输入的数据不会变成指令，使用无法通过该方法获得权限。

**Task 9**









根据实验结果可知，/etc/zzz文件会被修改，原因是在Set-UID程序中产生的文件句柄没有被删除，仍然存在；这导致句柄的权限也会被后续的语句继续使用，该句柄具有root权限，所以可以修改/etc/zzz文件。