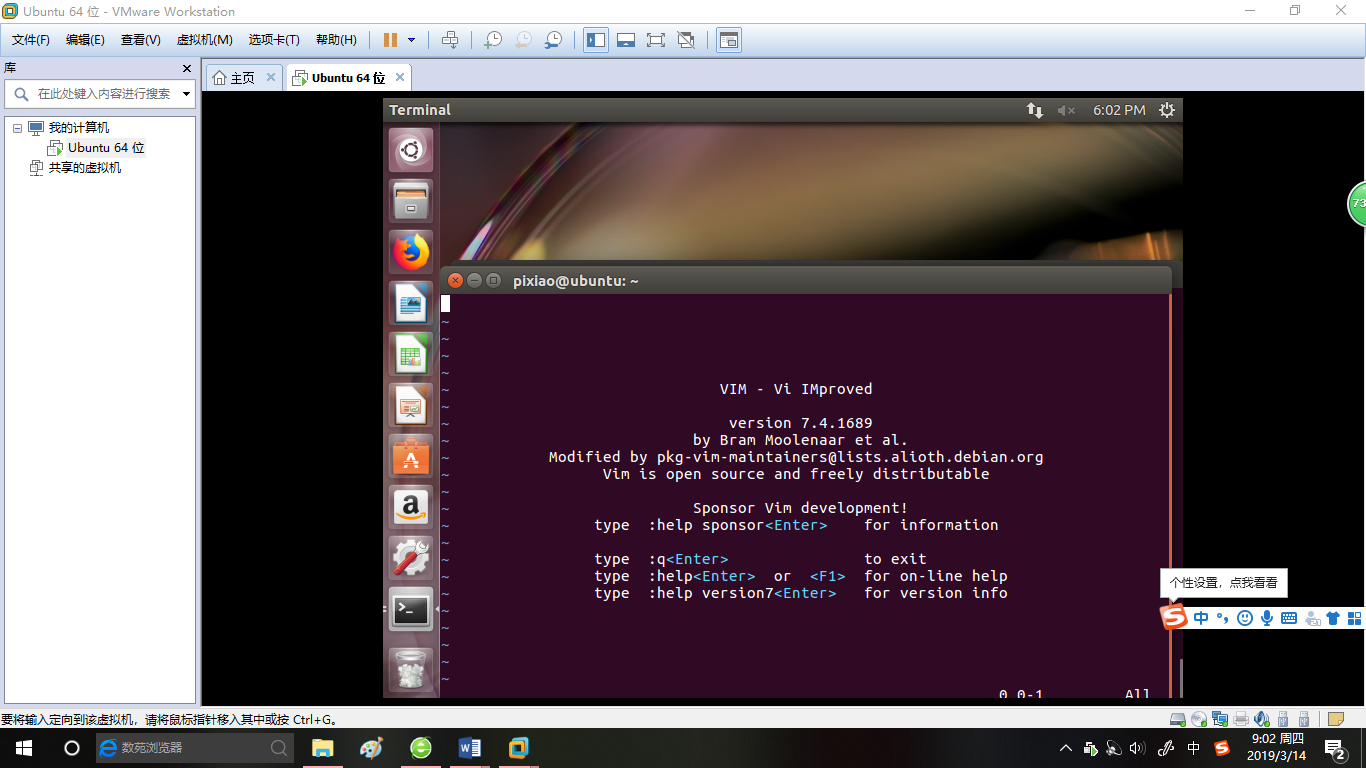
实验二 进程控制

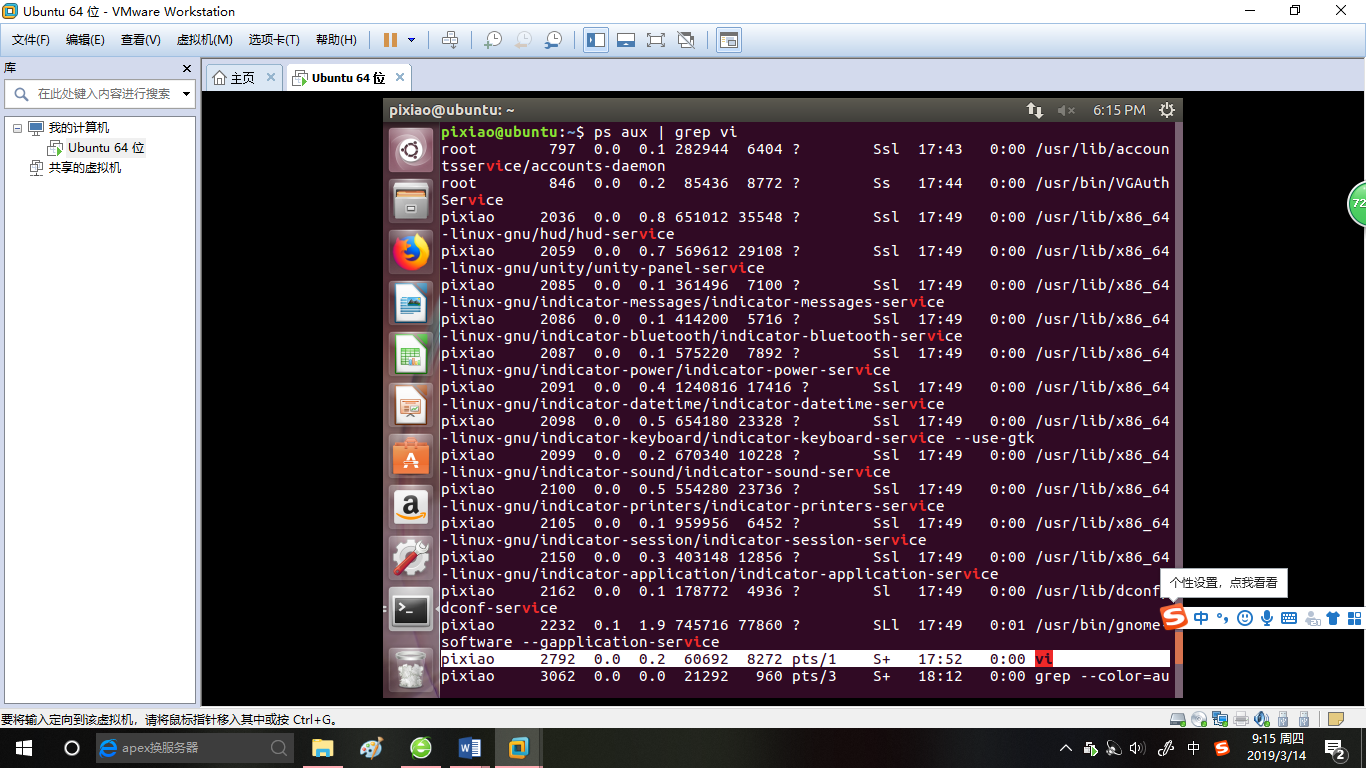
15282028 赵旭宏 安全1601

## 实验题目及过程：

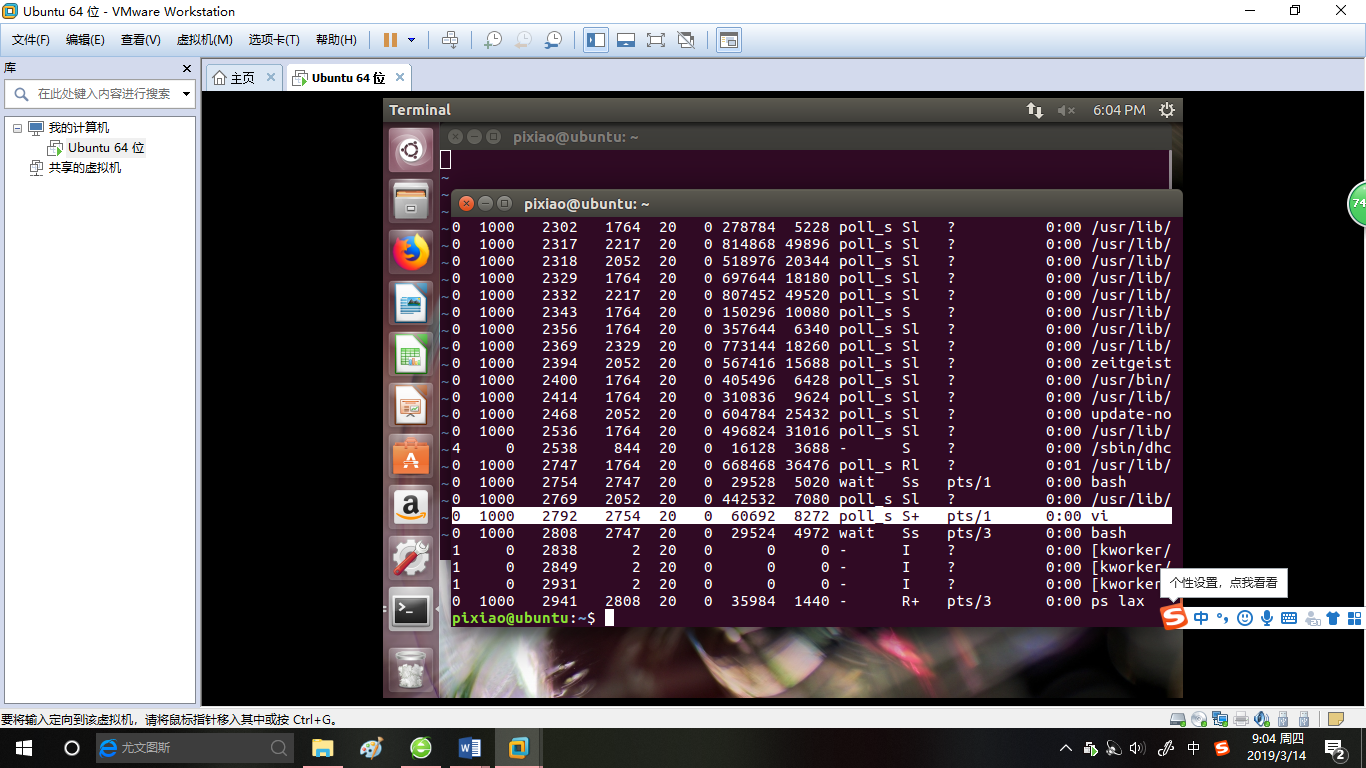
* 1. 打开一个vi进程。记录过程中所有进程的ID和父进程ID。将得到的进程树和由pstree命令的得到的进程树进行比较。



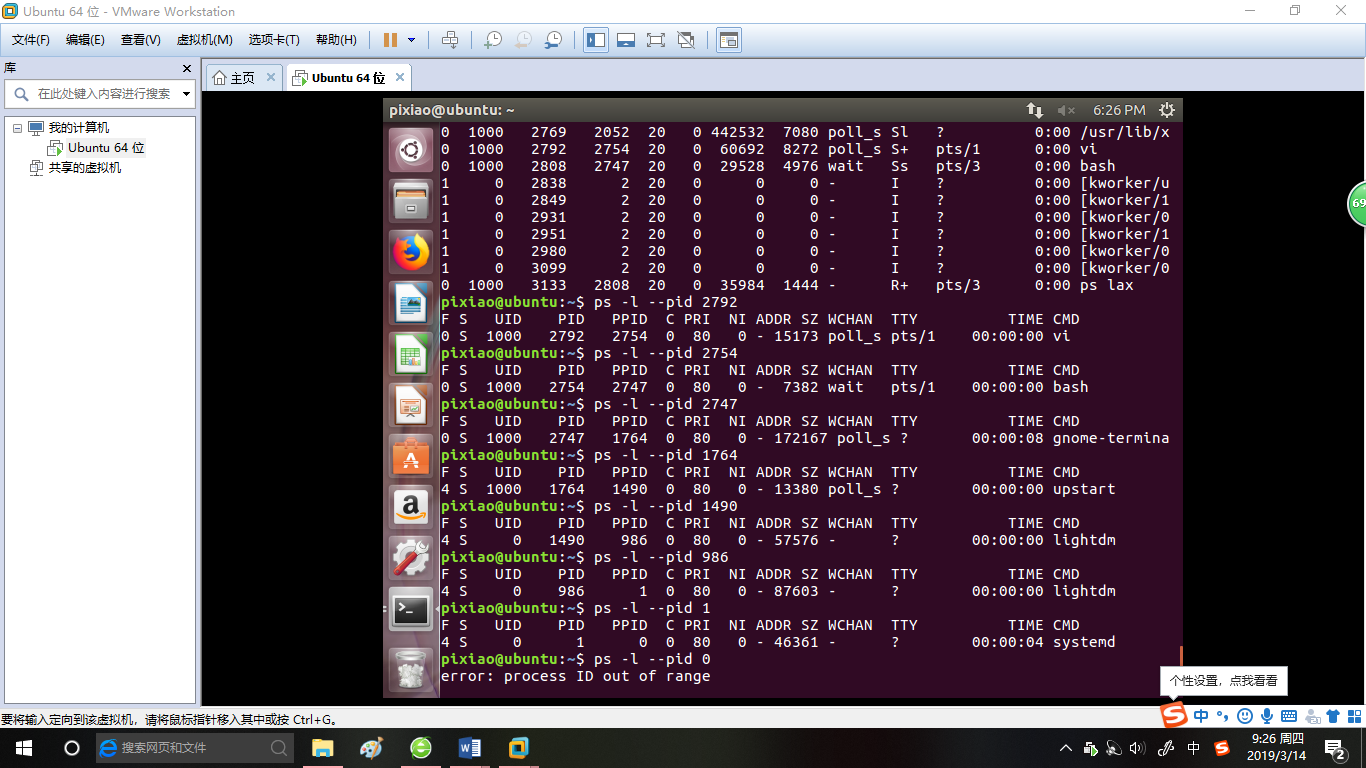
通过ps命令以及选择合适的参数，只显示名字为vi的进程：



输入ps -A命令，查看所有进程的ID，如下图可以看到，第一步打开的vi编辑器的id是2792：



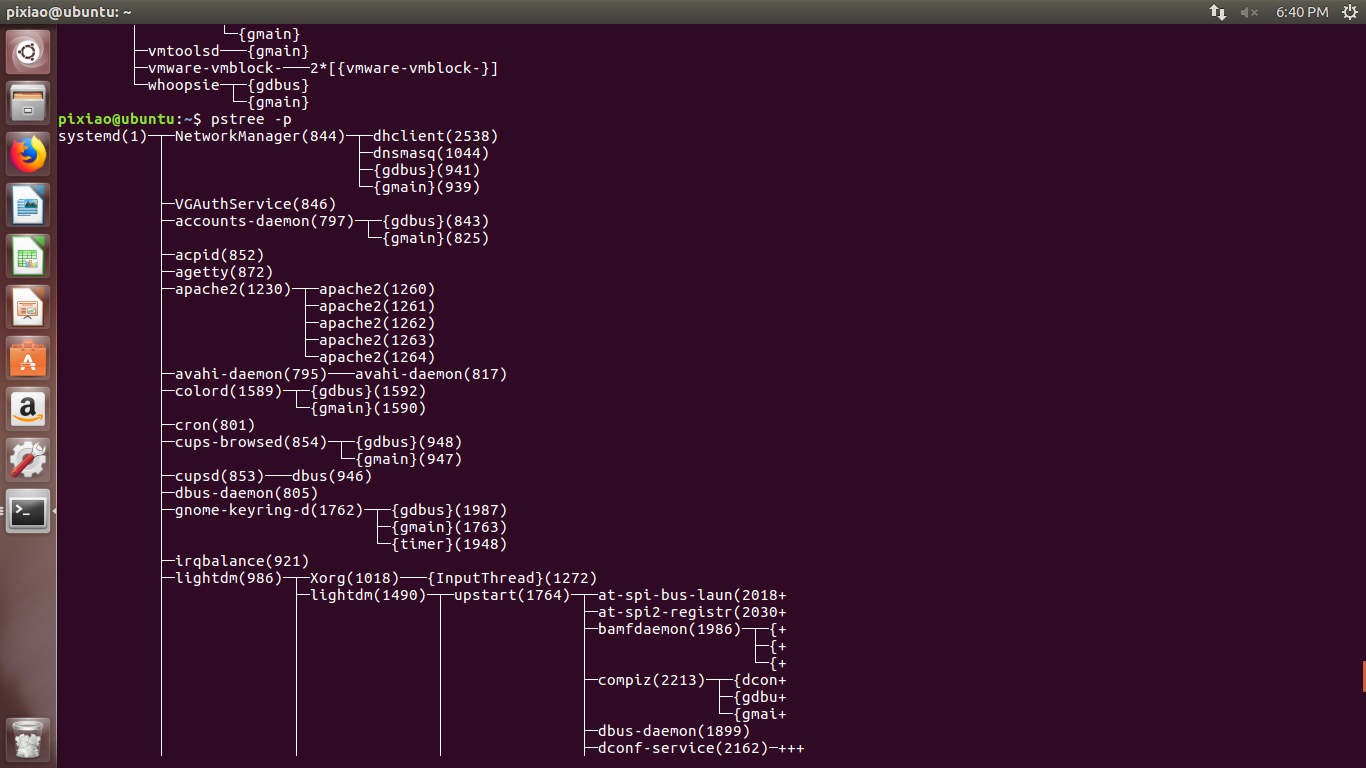
寻找vi进程的父进程，直到init进程为止：

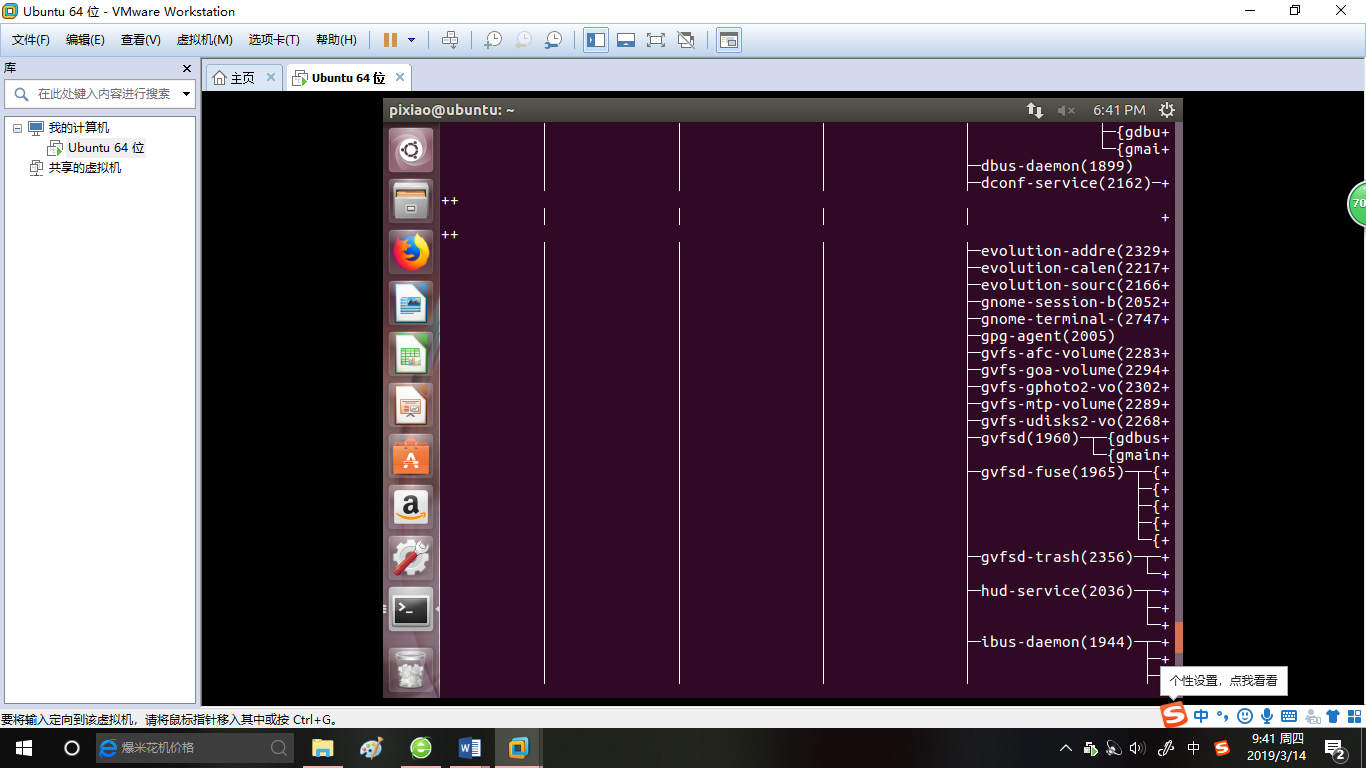


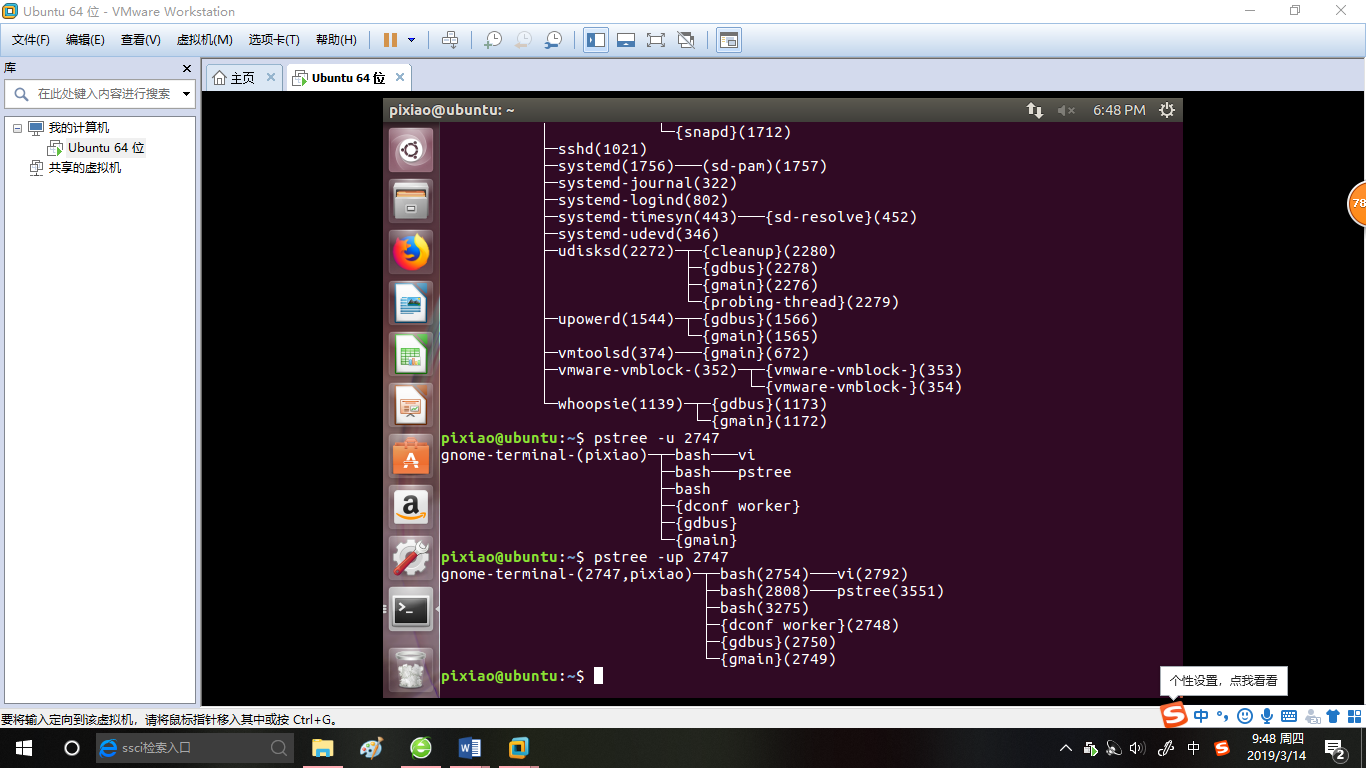
记录过程中所有进程的ID和父进程ID：

|  |  |
| --- | --- |
| 子进程ID | 父进程ID |
| 2792 （vi） | 2754 |
| 2754 | 2747 |
| 2747 | 1764 |
| 1764 | 1490 |
| 1490 | 986 |
| 986 | 1 |

由pstree命令的得到的进程树如下：

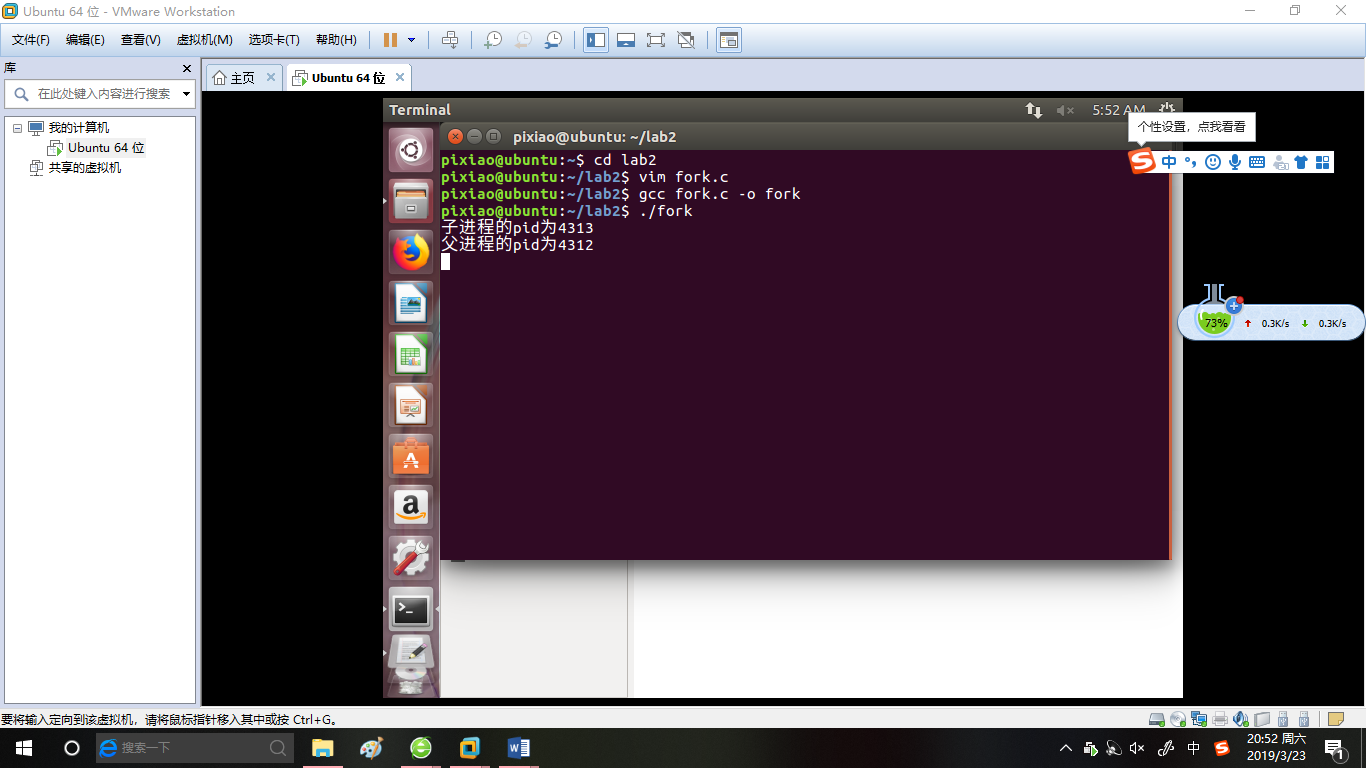




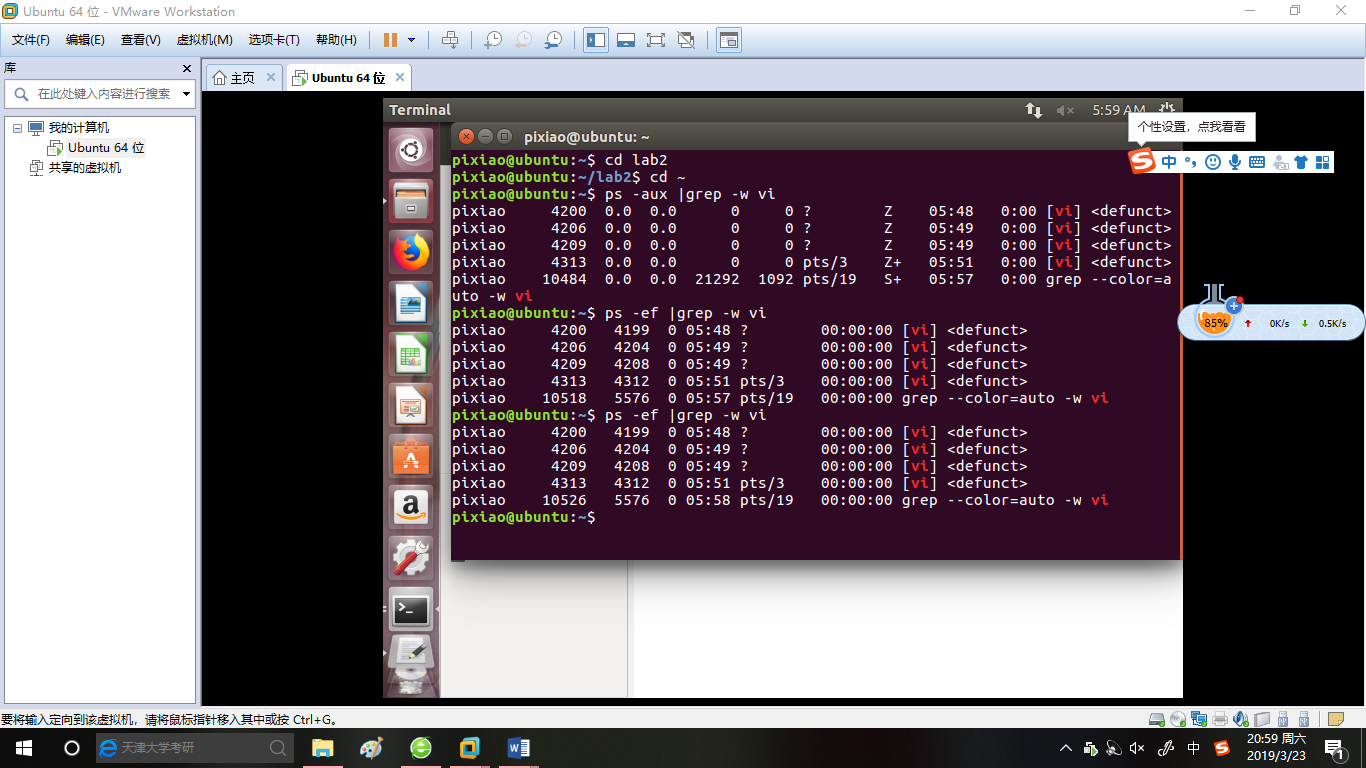


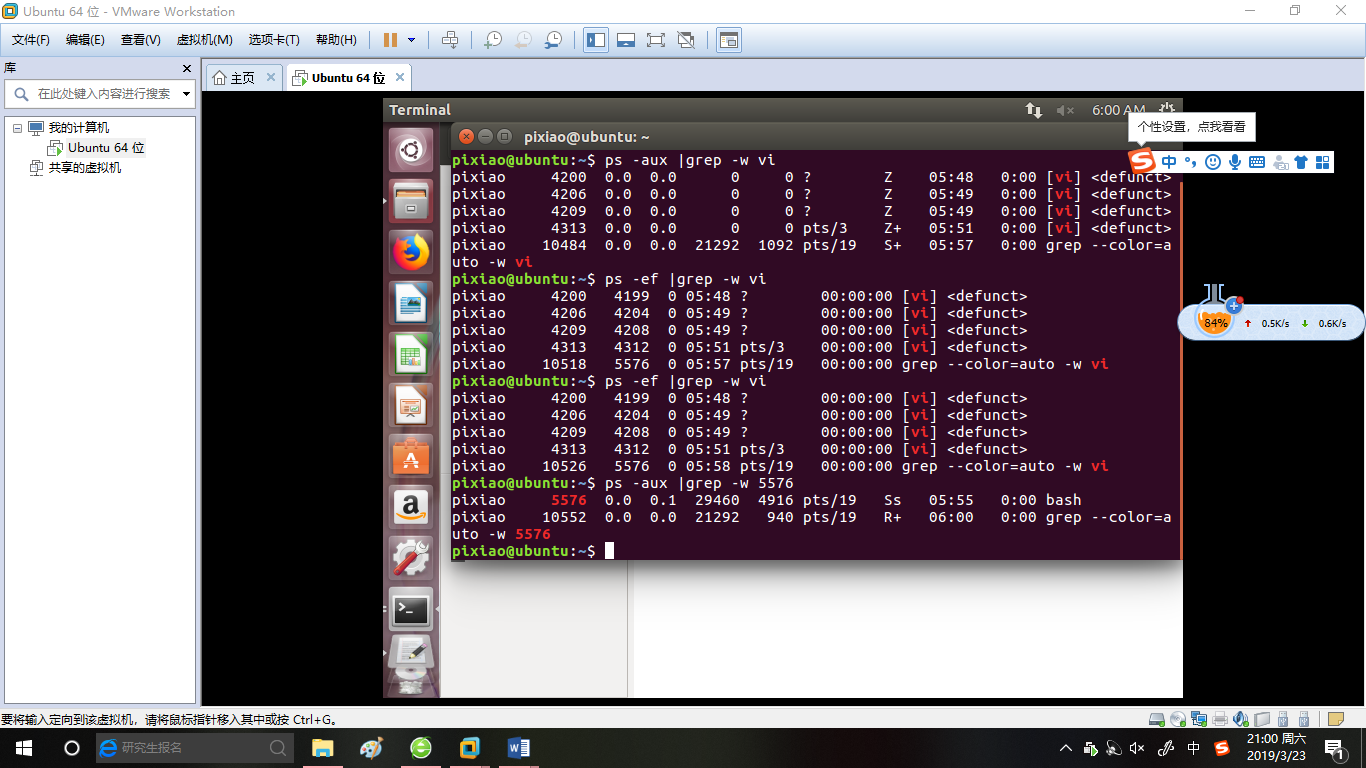
在Linux下，每个进程有唯一的PID标识进程。PID是一个从1到32768的正整数，其中1一般是特殊进程init，其它进程从2开始依次编号。当用完32768后，从2重新开始。进程在Linux中呈树状结构，init为根节点。将得到的进程树和由pstree命令的得到的进程树进行比较，发现是一致的。

* 2、编写程序，首先使用fork系统调用，创建子进程。在父进程中继续执行空循环操作；在子进程中调用exec打开vi编辑器。



在另外一个终端中，通过ps aux等命令，查看vi进程及其父进程的运行状态。





**了解linux 下的ps命令及参数含义:**

%CPU 进程的cpu占用率

%MEM 进程的内存占用率

VSZ 进程所使用的虚存的大小

RSS 进程使用的驻留集大小或者是实际内存的大小

TTY 与进程关联的终端（tty）

STAT 检查的状态：进程状态使用字符表示的，如R（running正在运行或准备运行）、S（sleeping睡眠）、I（idle空闲）、Z (僵死)、D（不可中断的睡眠，通常是I/O）、P（等待交换页）、W（换出,表示当前页面不在内存）、N（低优先级任务）T(terminate终止)、W has no resident pages

START （进程启动时间和日期）

TIME ;（进程使用的总cpu时间）

COMMAND （正在执行的命令行命令）

NI (nice)优先级

PRI 进程优先级编号

PPID 父进程的进程ID（parent process id）

SID 会话ID（session id）

WCHAN 进程正在睡眠的内核函数名称；该函数的名称是从/root/system.map文件中获得的。

FLAGS 与进程相关的数字标识

常用参数

-A 显示所有进程（等价于-e）(utility)

-a 显示一个终端的所有进程，除了会话引线

-N 忽略选择。

-d 显示所有进程，但省略所有的会话引线(utility)

-x 显示没有控制终端的进程，同时显示各个命令的具体路径。dx不可合用。（utility）

-p pid 进程使用cpu的时间

-u uid or username 选择有效的用户id或者是用户名

-g gid or groupname 显示组的所有进程。

U username 显示该用户下的所有进程，且显示各个命令的详细路径。如:ps U zhang;(utility)

-f 全部列出，通常和其他选项联用。如：ps -fa or ps -fx and so on.

-l 长格式（有F,wchan,C 等字段）

-j 作业格式

-o 用户自定义格式。

v 以虚拟存储器格式显示

s 以信号格式显示

-m 显示所有的线程

-H 显示进程的层次(和其它的命令合用，如：ps -Ha)（utility）

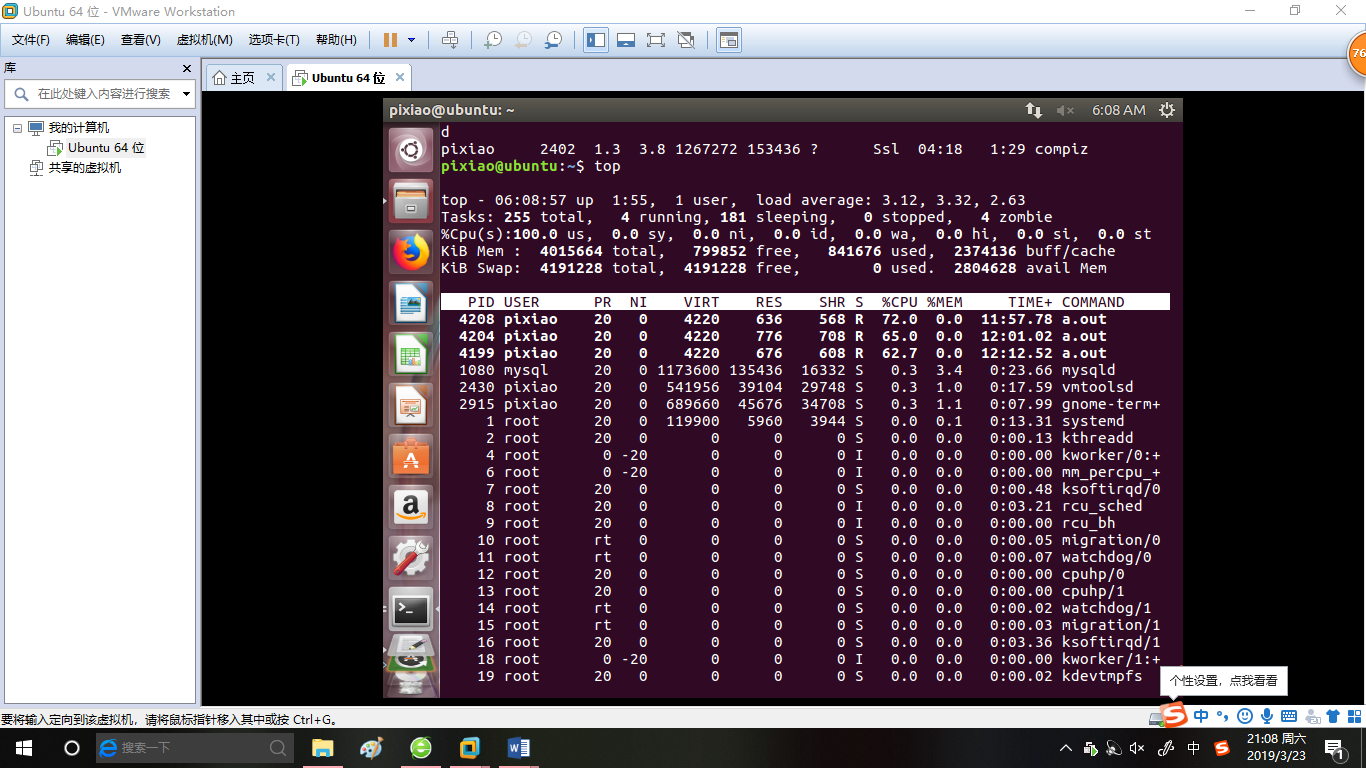
e 命令之后显示环境（如：ps -d e; ps -a e）(utility)

h 不显示第一行

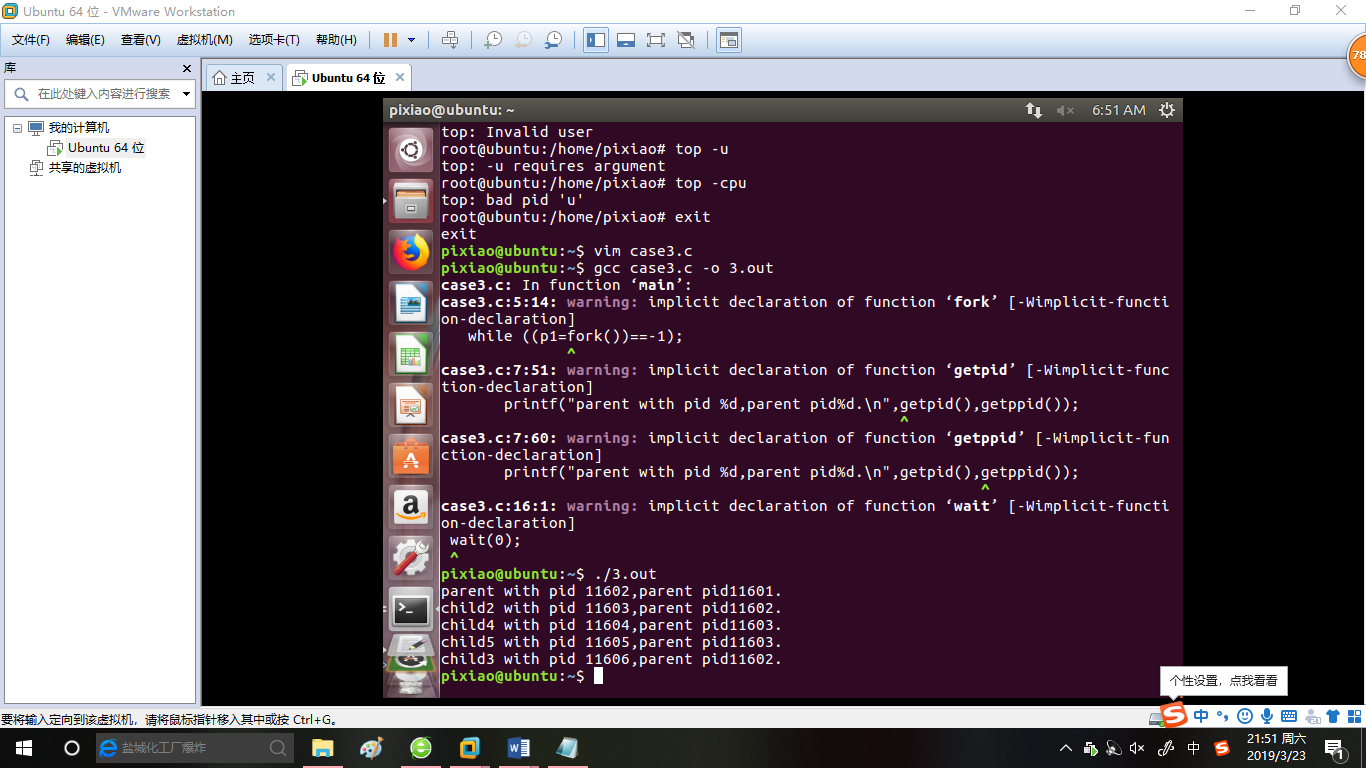
**使用ps -aux的命令对所有进程占用cpu率进行排序：**



**执行top命令，查看cpu占用率：**



* 3、使用fork系统调用，创建如下进程树，并使每个进程输出自己的ID和父进程的ID。观察进程的执行顺序和运行状态的变化。



流程图：

开始

p1=fork()

p1==0或

p1>0

P1==0

P1>0

P2=fork()

fork()

P2==0或

P2>0

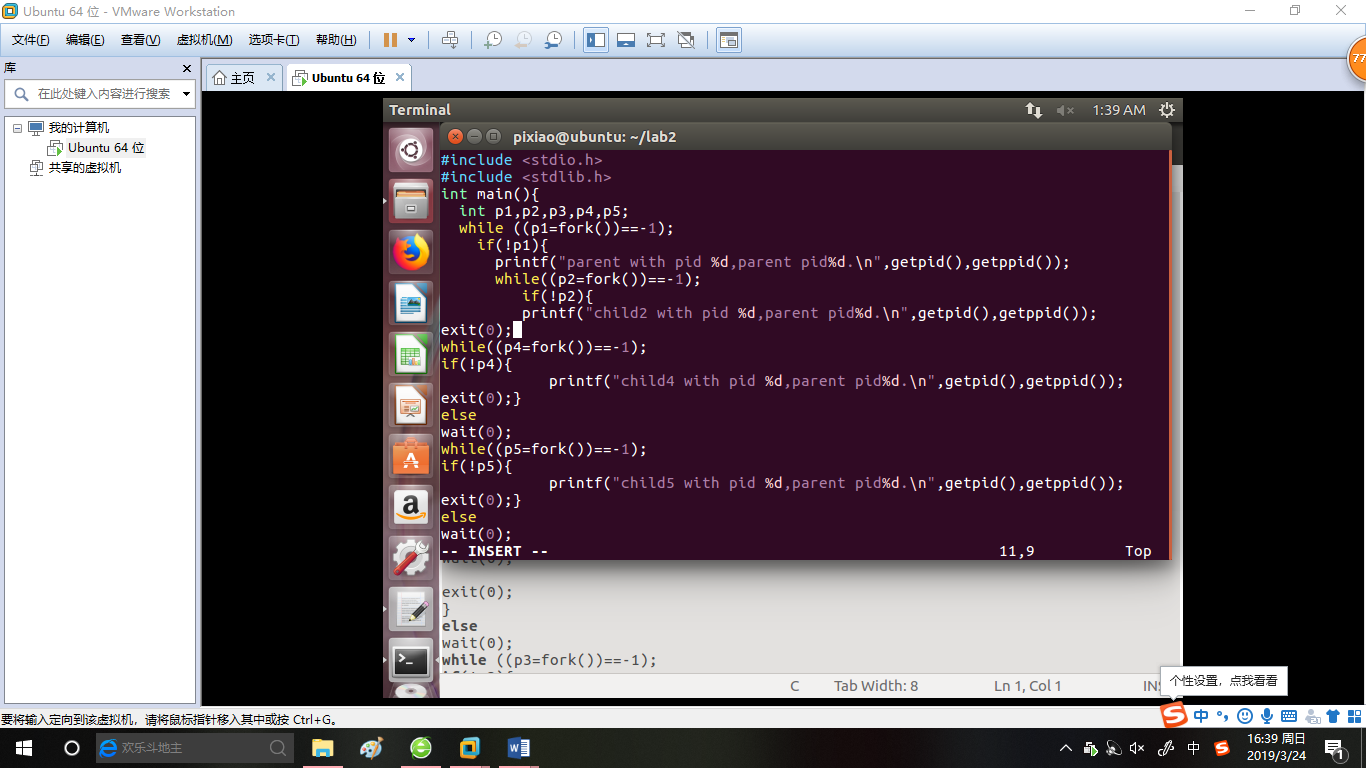
输出进程pid和parent的pid

P2>0

p2==0

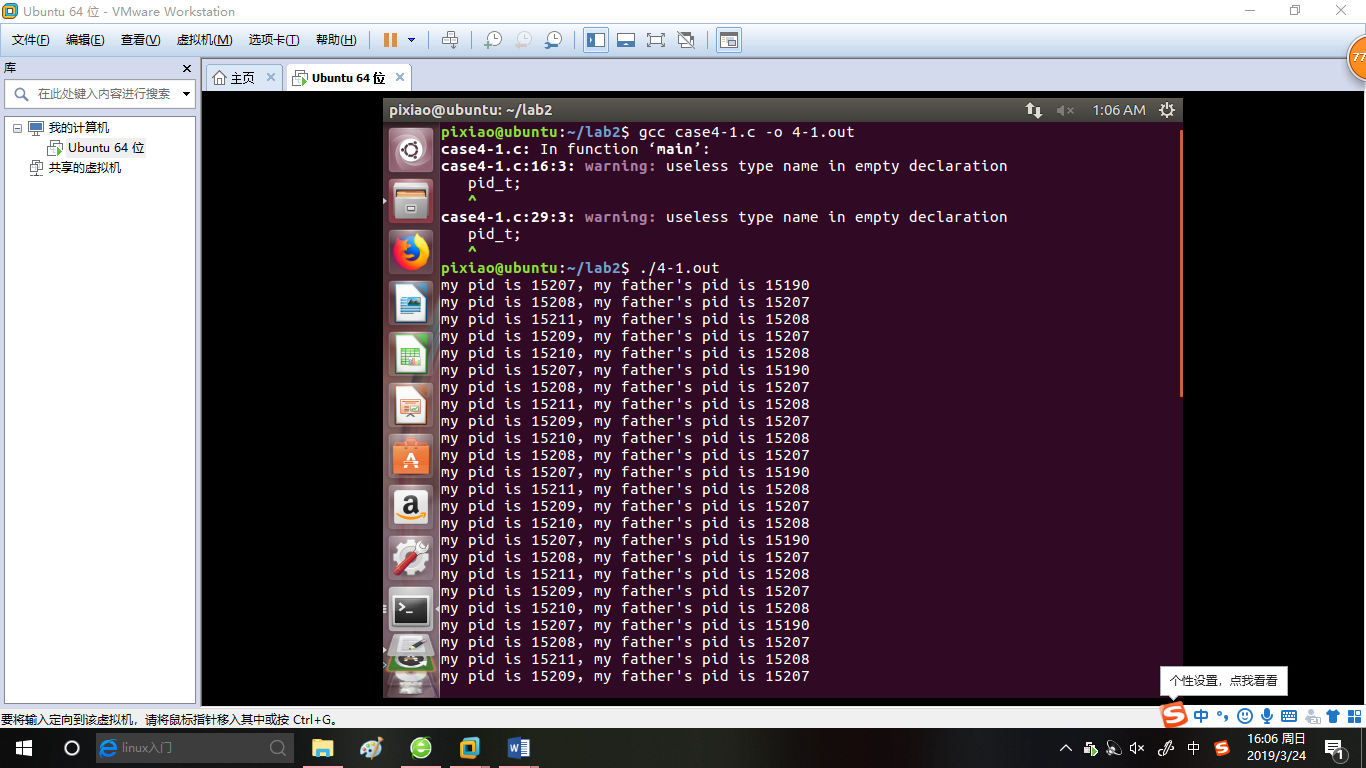
输出进程pid和parent的pid

 4、修改上述进程树中的进程，使得所有进程都循环输出自己的ID和父进程的ID。然后终止p2进程，观察发现，终止p2进程后，p4，p5进程fork失败。

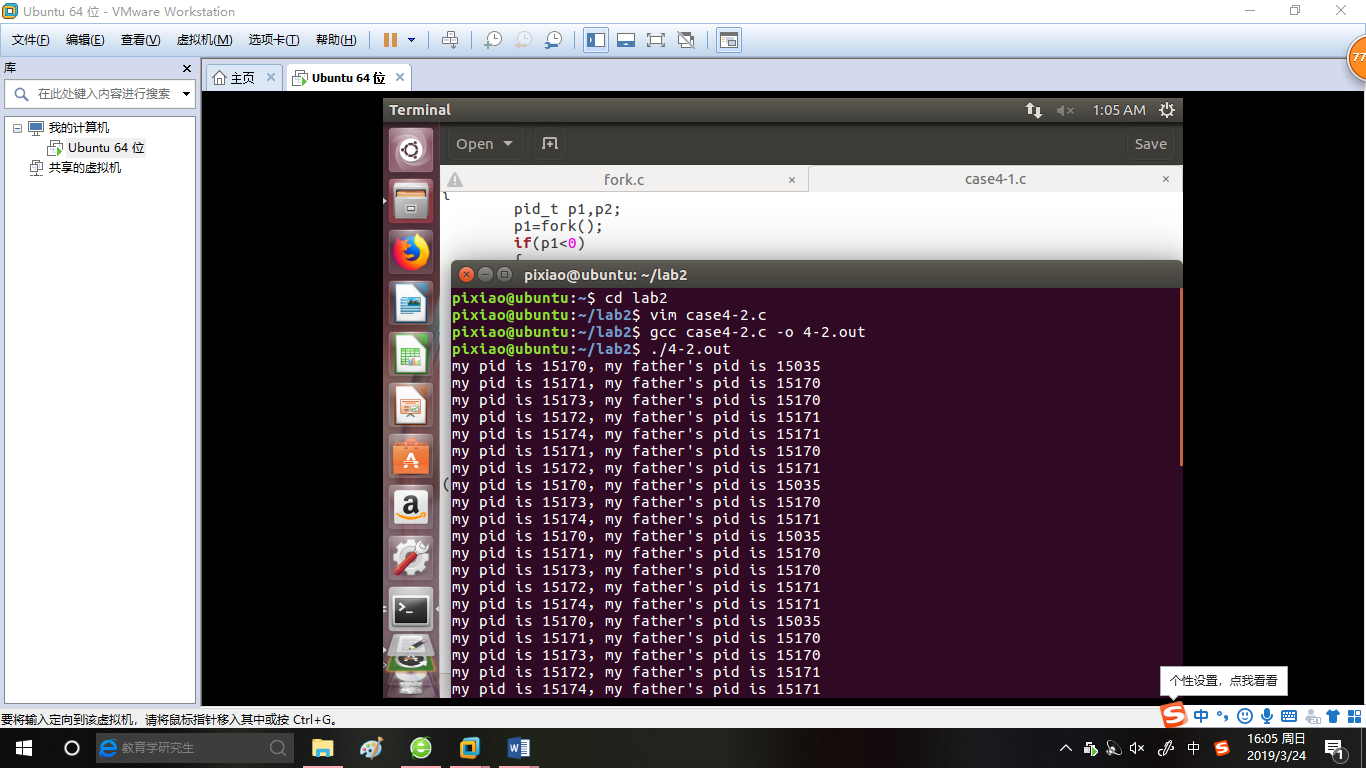




采用kill -9：



自己正常退出exit()：p2在执行十次以后输出就结束。



段错误退出：在进程 p2 的进程段里面定义一个野指针，野指针没有初始化会产生段错误导致进程退出。后面15433就消失了。

