**实验一: 操作系统初步实验报告**

**计科1601班 马静雯 16281046**

一、（系统调用实验）了解系统调用不同的封装形式。

要求：1、参考下列网址中的程序。阅读分别运行用API接口函数getpid()直接调用和汇编中断调用两种方式调用Linux操作系统的同一个系统调用getpid的程序

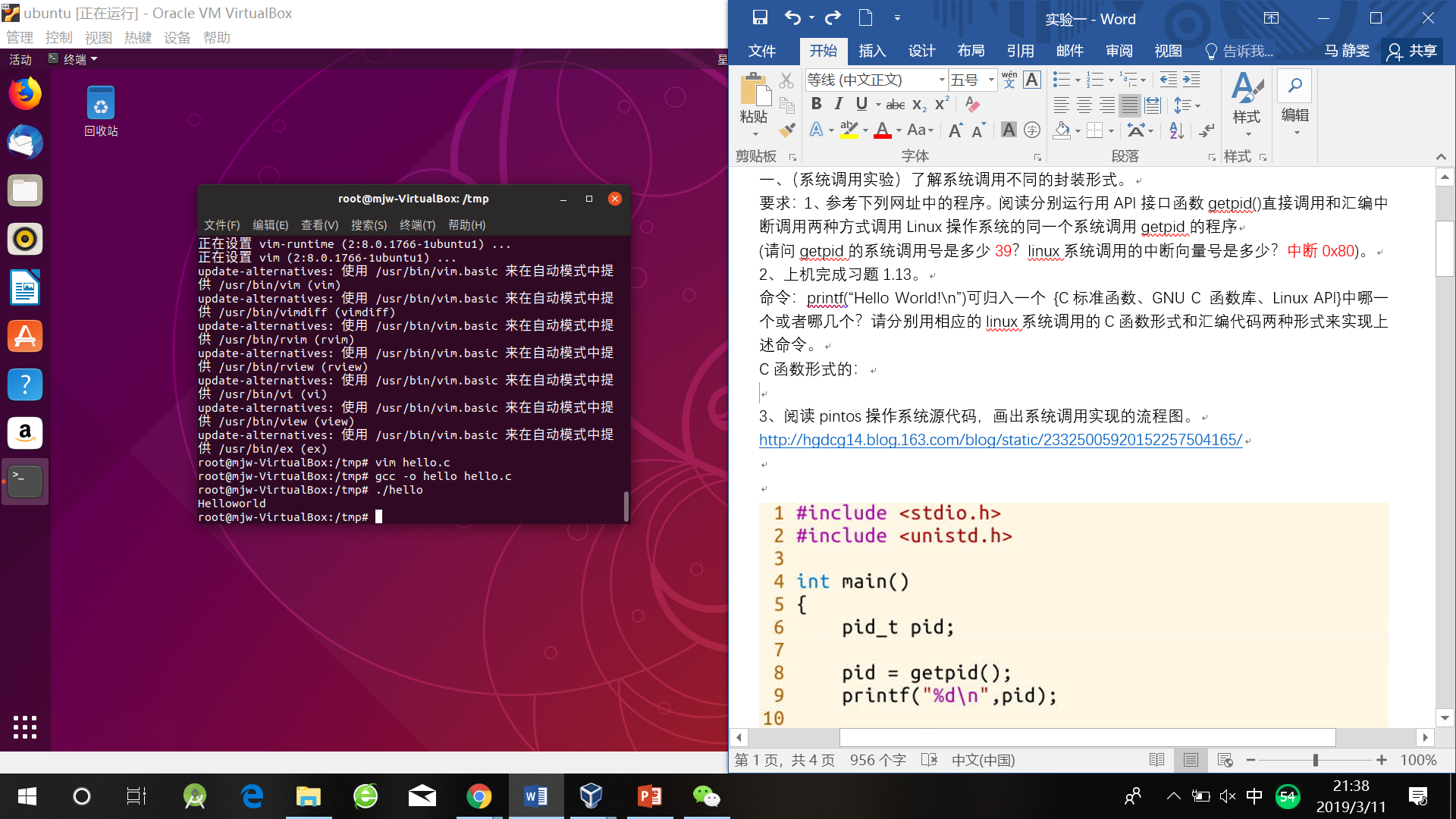
请问getpid的系统调用号是多少？39

linux系统调用的中断向量号是多少？中断0x80。

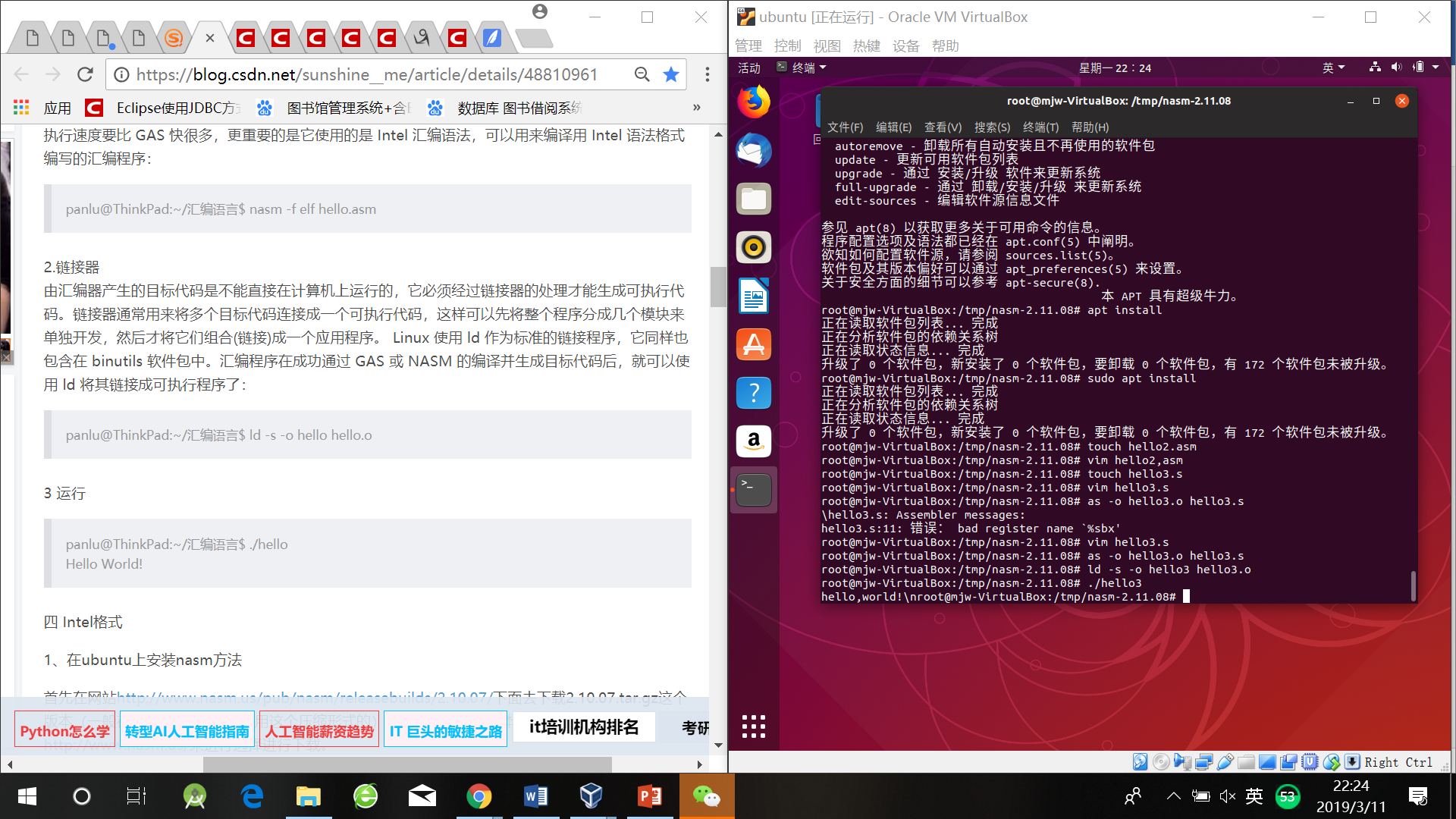
2、上机完成习题1.13。

命令：printf(“Hello World!\n”)可归入一个 {C标准函数、GNU C 函数库、Linux API}中哪一个或者哪几个？请分别用相应的linux系统调用的C函数形式和汇编代码两种形式来实现上述命令。

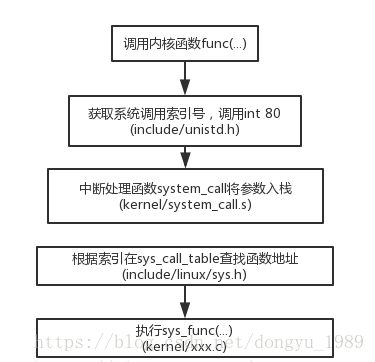
C函数形式：



汇编形式：



3、阅读pintos操作系统源代码，画出系统调用实现的流程图。



二、（并发实验）根据以下代码完成下面的实验。

要求：

1. 编译运行该程序（cpu.c），观察输出结果，说明程序功能。

(编译命令： gcc -o cpu cpu.c –Wall)（执行命令：./cpu）

1. 再次按下面的运行并观察结果：执行命令：./cpu A & ; ./cpu B & ; ./cpu C & ; ./cpu D &程序cpu运行了几次？他们运行的顺序有何特点和规律？请结合操作系统的特征进行解释。

事实上，它所做的是调用Spin()，一个反复检查时间并一旦运行一秒钟就返回的函数。 然后，它打印出用户在命令行上输入的字符串，并一直重复。将单个CPU（或一小组CPU）转换为看似无限数量的CPU，从而允许许多程序看起来一次运行，这就是我们所谓的虚拟化CPUCPU运行了4次，运行的顺序是随机的。



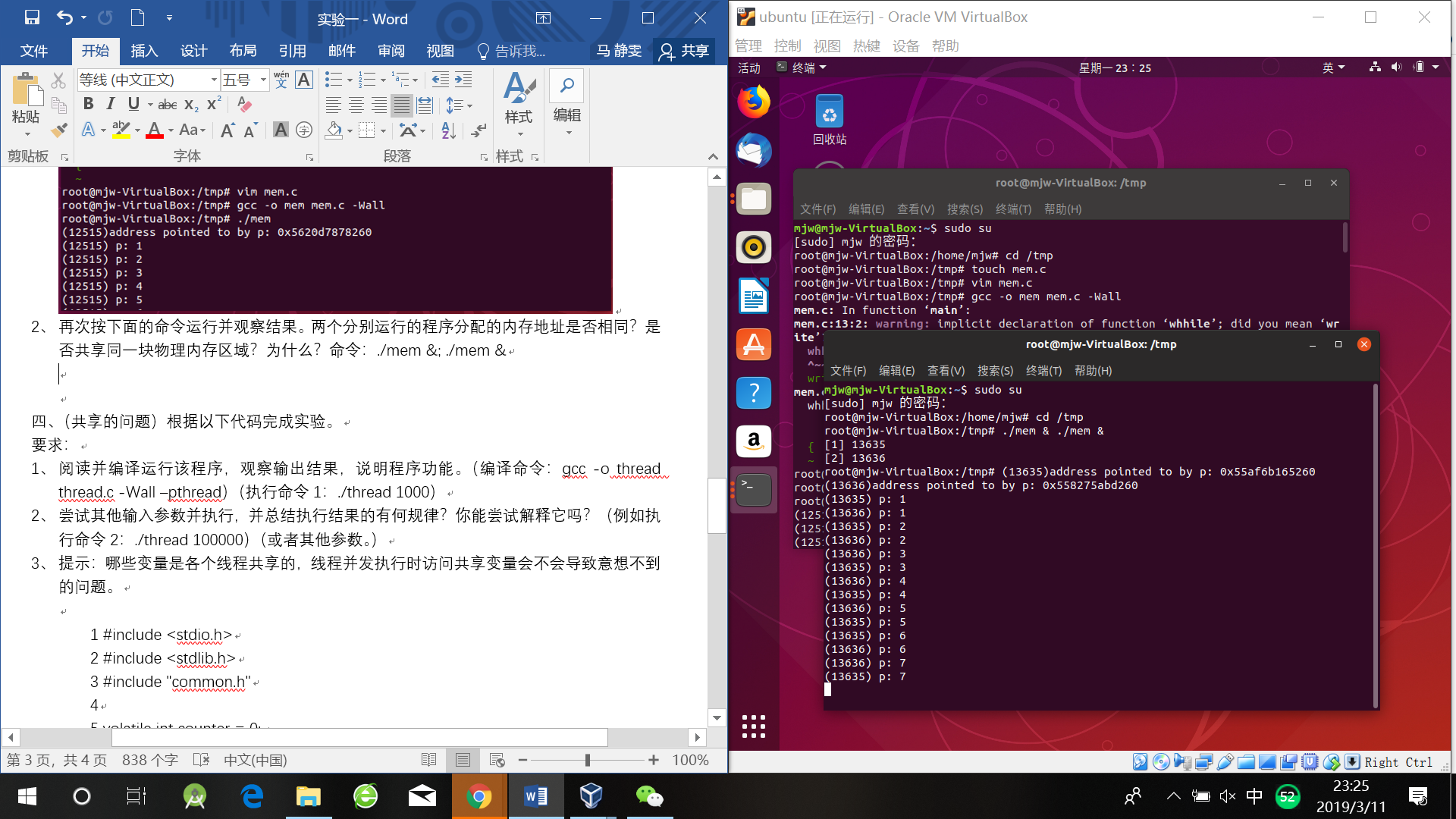
三、（内存分配实验）根据以下代码完成实验。

要求：

1. 阅读并编译运行该程序(mem.c)，观察输出结果，说明程序功能。(命令： gcc -o mem mem.c –Wall) 

程序功能：该程序做了几件事。 首先，它分配一些内存。 然后，它打印出内存地址（a2），然后将数字0放入新分配的内存的第一个位置。 最后，它循环：延迟一秒并递增存储在p中保存的地址的值。 对于每个print语句，它还会打印出正在运行的程序的进程标识符。 该PID在每个运行过程中都是唯一的。

1. 再次按下面的命令运行并观察结果。两个分别运行的程序分配的内存地址是否相同？是否共享同一块物理内存区域？为什么？命令：./mem &; ./mem &

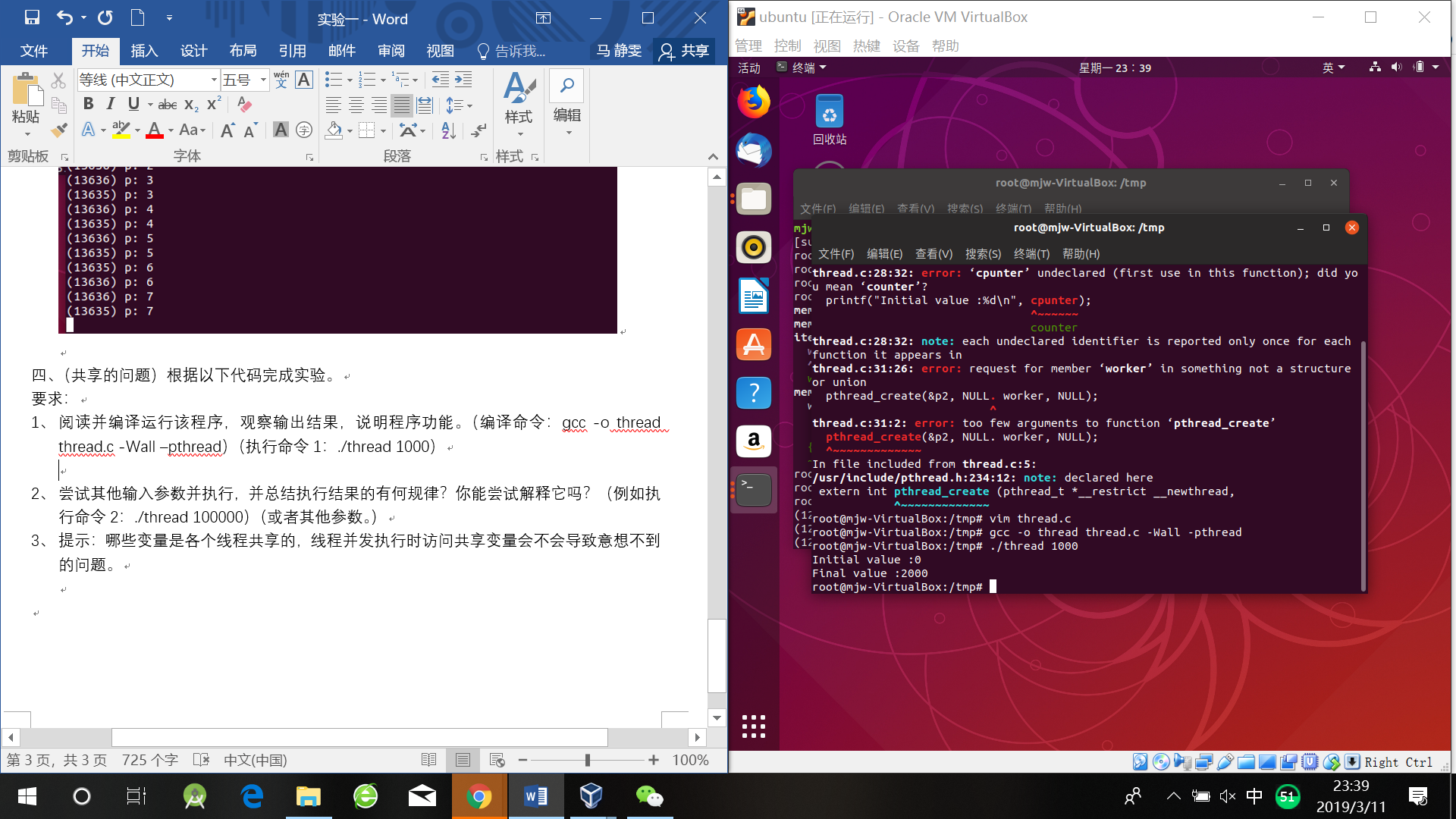


实际上，这正是这里发生的事情，因为操作系统虚拟化了内存。 每个进程访问自己的私有虚拟地址空间（有时只称为其地址空间(address space)），操作系统以某种方式映射到机器的物理内存。 一个正在运行的程序中的内存引用不会影响其他进程（或OS本身）的地址空间；就运行程序而言，它拥有所有的物理内存。 然而，现实是物理内存是由操作系统管理的共享资源。

四、（共享的问题）根据以下代码完成实验。

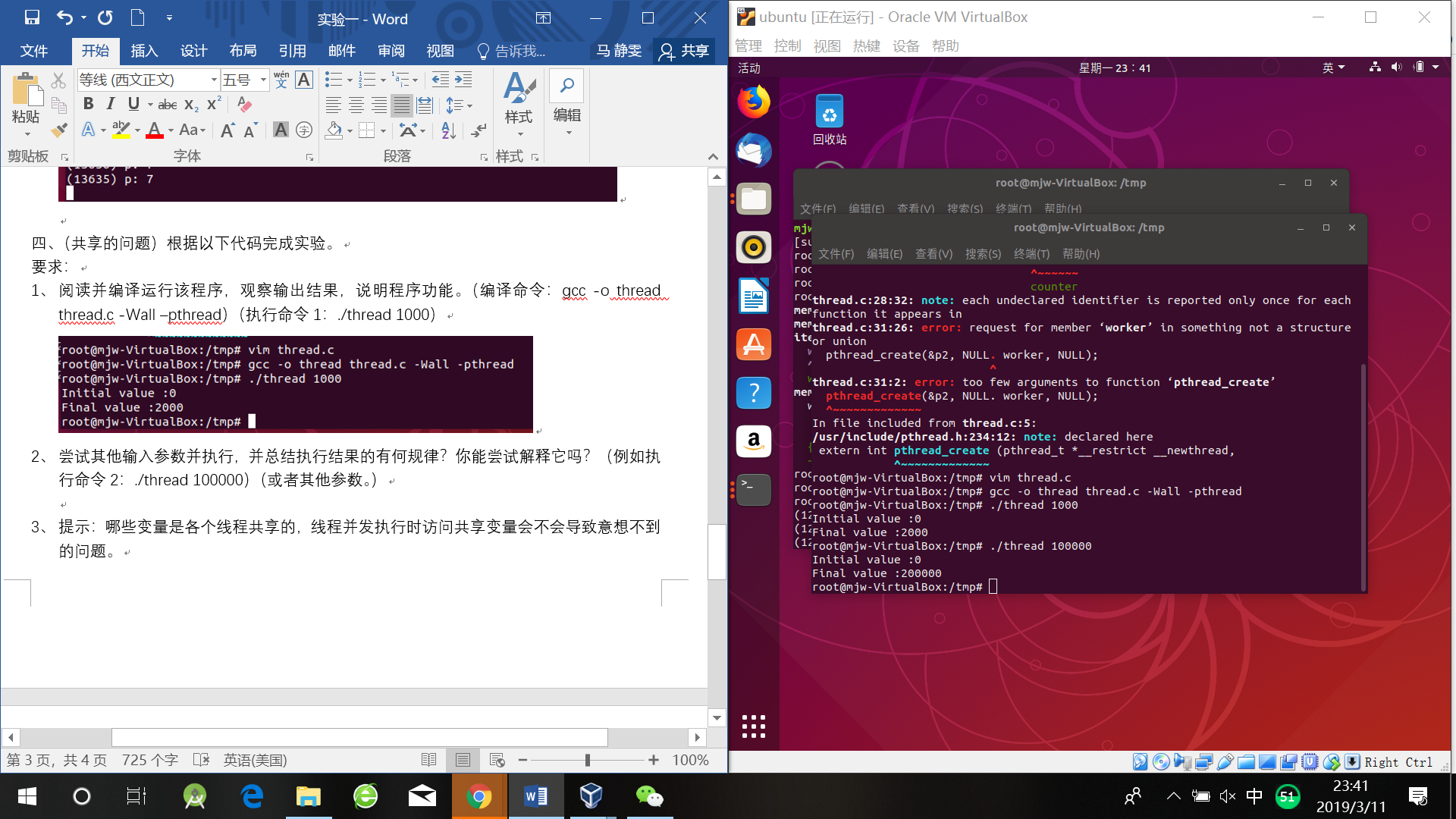
要求：

1. 阅读并编译运行该程序，观察输出结果，说明程序功能。（编译命令：gcc -o thread thread.c -Wall –pthread）（执行命令1：./thread 1000）



程序功能：主程序使用Pthread .create()创建两个线程。 您可以将线程视为在与其他函数相同的内存空间中运行的函数，其中一次激活多个函数。

1. 尝试其他输入参数并执行，并总结执行结果的有何规律？你能尝试解释它吗？（例如执行命令2：./thread 100000）（或者其他参数。）



当两个线程完成时，计数器的最终值为2000和200000，因为每个线程将计数器递增1000次和100000次。 实际上，当循环的输入值设置为N时，我们期望程序的最终输出为2N