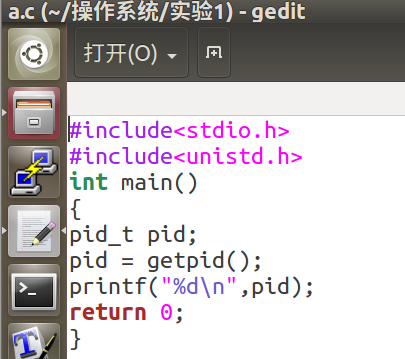
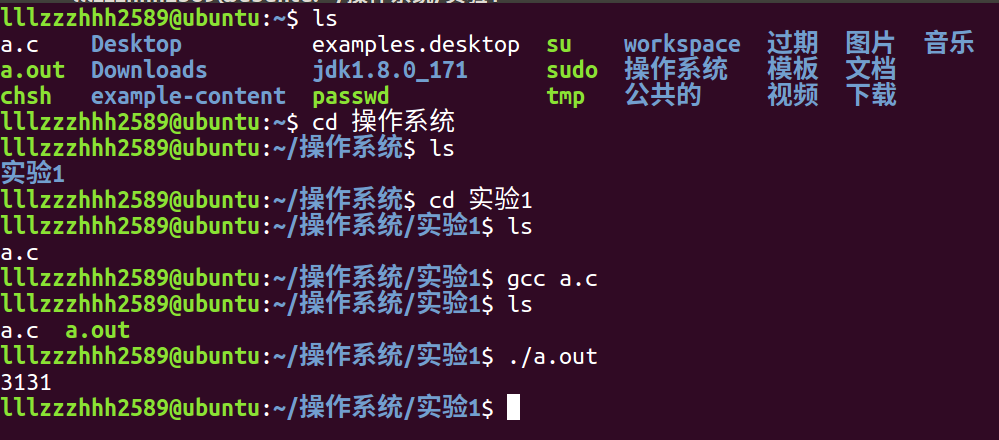
操作系统第一次实验

罗泽昊 16281294 安全1601

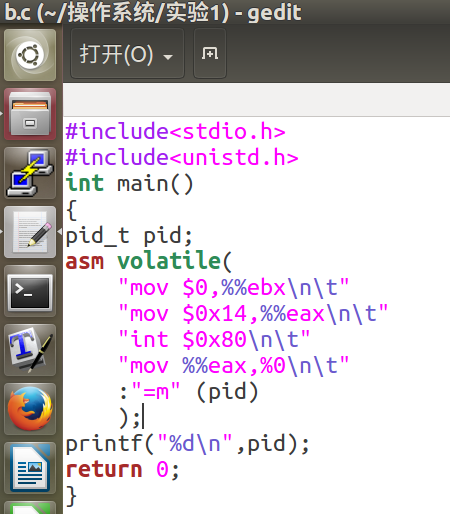
# 一、系统调用实验

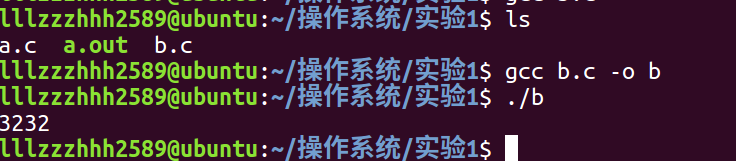
## 1.1运用不同的方式调用getpid程序





直接调用





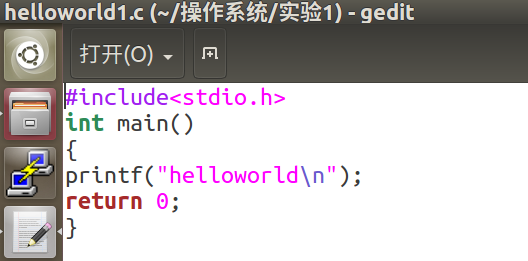
汇编间接调用

getpid的系统调用号是39

linux系统调用的中断向量号是0x80

## 1.2上机完成习题1.13

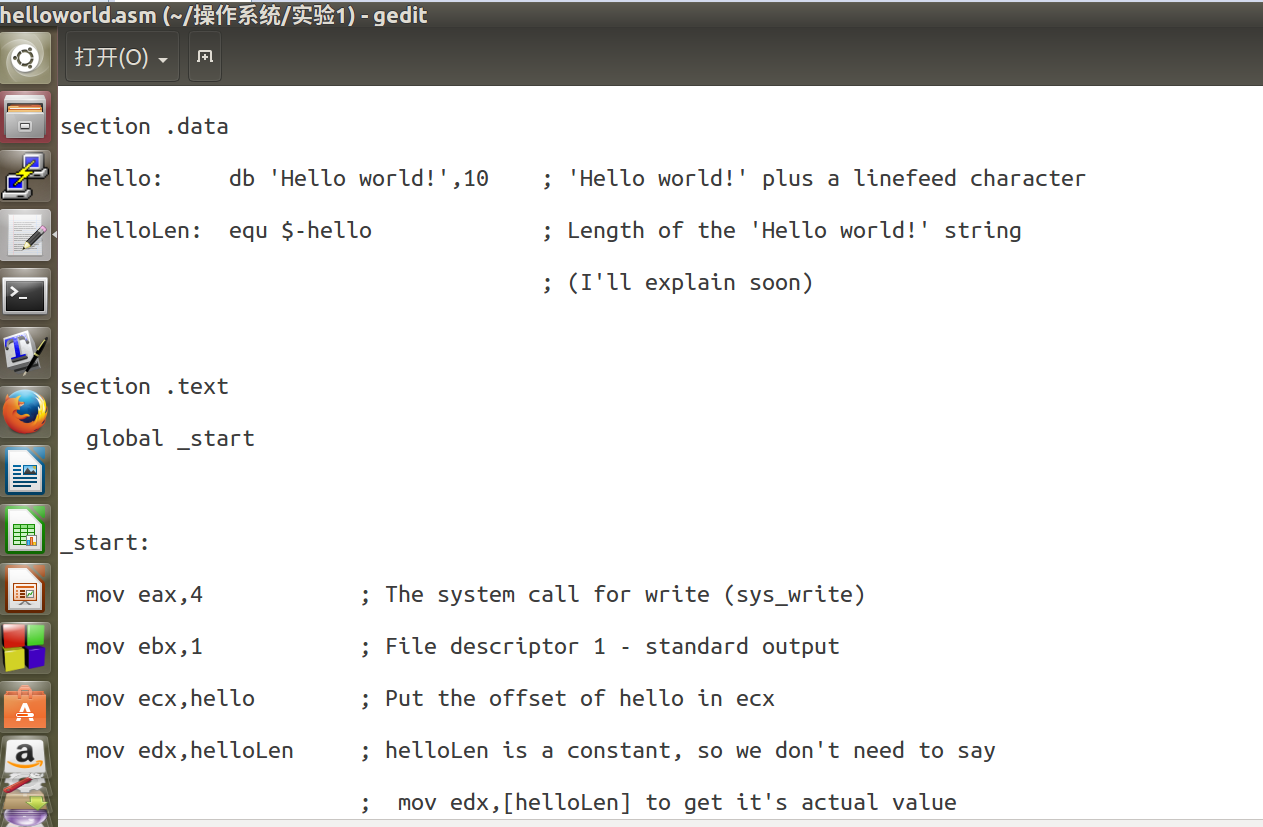
C语言方法

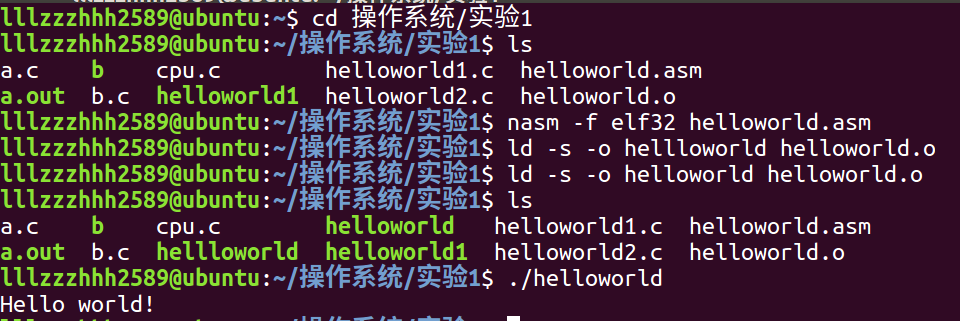




汇编方法

安装NASM（是一款基于80x86和x86-64平台的汇编语言编译程序），并用汇编方法实现helloworld





参考网址<https://blog.csdn.net/qq_41315938/article/details/83785381>

## 1.3系统调用实现流程图

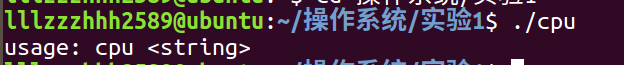


# 二、并发实验

## 2.1编译运行该程序

执行./cpu，没有传入参数，所以argc=1，所以输出如图所示结果。

从代码中我们可以看出，当输入一个参数时，程序执行while（1）后的内容，将会输出argv[1]里的内容，而argv数组存储的就是我们在shell中传入的参数，即argv[0]是程序名，argv[1]是参数名。



## 2.2执行命令：./cpu A & ; ./cpu B & ; ./cpu C & ; ./cpu D &

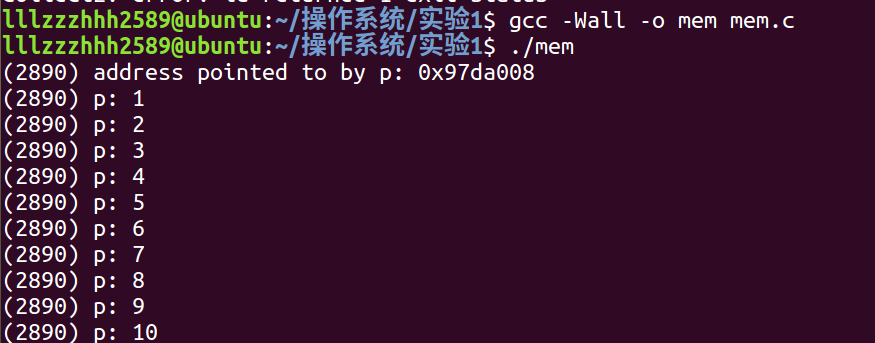


程序CPU运行了4次，程序运行的顺序是随机无序的。

原因是这四个程序ABCD是并发执行的，即在这一时间段内四个程序同时执行，但在某一时刻，却只有一个程序在实际执行，在这期间，程序之间就需要竞争资源，从而导致了程序是走走停停进行的，执行完成的顺序因此也是杂乱无序的。

# 三、内存分配实验

## 3.1阅读并编译运行该程序(mem.c)



该程序首先会输出p在内存中分配的地址，之后会持续输出p的进程号以及执行的次数

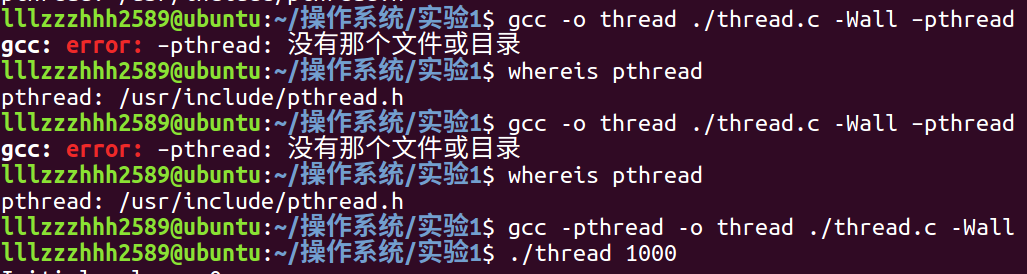
## 3.2再次按下面的命令运行并观察结果

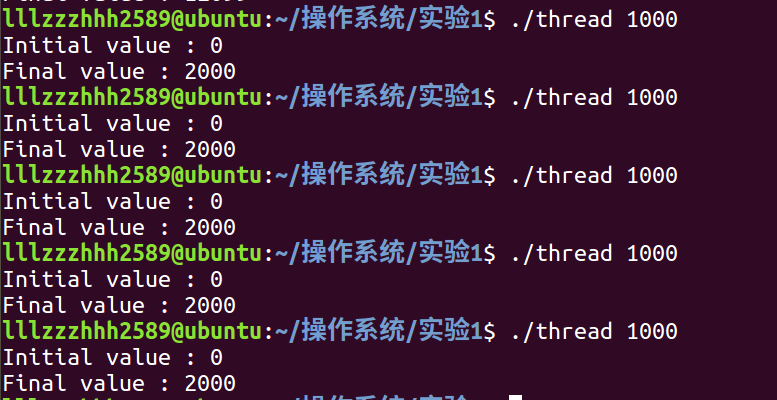


同时运行两个程序，这两个程序分配的内存地址不同，没有共享同一块物理内存区域。相当于两个不同的程序在分别执行，体现了内存分配。

# 4共享的问题

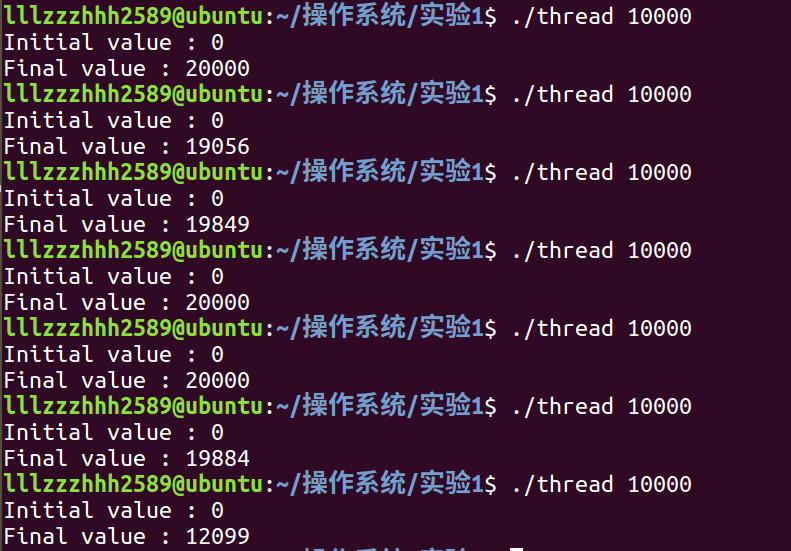
## 4.1阅读并编译运行该程序，观察输出结果，说明程序功能。





这个程序创建了两个线程，每个线程均进行了函数worker的运算，即各进行了一个loops大小的递加运算，loops即为输入的参数大小。最后输出运算的初始值以及最终结果（输入的二倍）。

## 4.2尝试其他输入参数并执行



多次实验，输出的最终结果有时不是二倍，二是小了一些。

### 4.3哪些变量是各个线程共享的，线程并发执行时访问共享变量会不会导致意想不到的问题

Counter是两个线程共享的，会遇到问题，即执行自增运算但并没有改变counter的大小，或者说两个线程共享一个counter资源时出现了错误。

原因：线程共享进程的地址空间，所以线程对资源的改变会反映到进程中

# 五、遇到的问题和解决方法总结

## 5.1汇编的实现

通过查询，了解到了nasm这个汇编语言编译程序，并对汇编进行了一小部分的自学

## 5.2程序代码的修改

老师给的代码中有一些错误，除了同学给出的删去"common.h"，改Spin(1)为sleep(1)

还有一些头文件要补，如<assert.h>,<pthread.h>, <unistd.h>等，还有一些大小写的问题

## 5.3编译命令的修改

通过查询网络，询问同学和老师，修改了执行gcc的命令格式，将gcc命令链接库文件的命令移到前面解决了问题成功编译。如gcc -o thread thread.c -Wall –pthread改为gcc –pthread -o thread thread.c -Wall 等。

## 5.4总结

有较大收获，对操作系统、Linux命令都有了更多的了解。