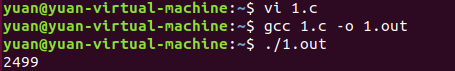
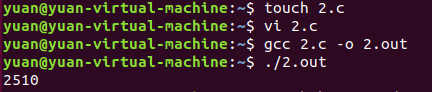
# 一系统调用实验

## 1 运用不同的方式调用getpid程序

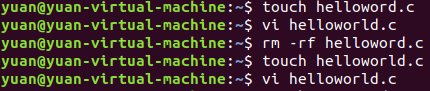
getpid的系统调用号39，中断向量号int 80h

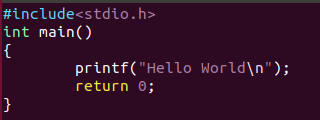




## 2 习题1.13

### 2.1 C语言方式实现







### 2.2 汇编方式实现





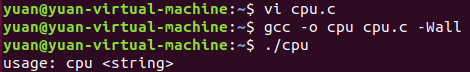


## 3 系统调用实现流程图



# 二并发实验

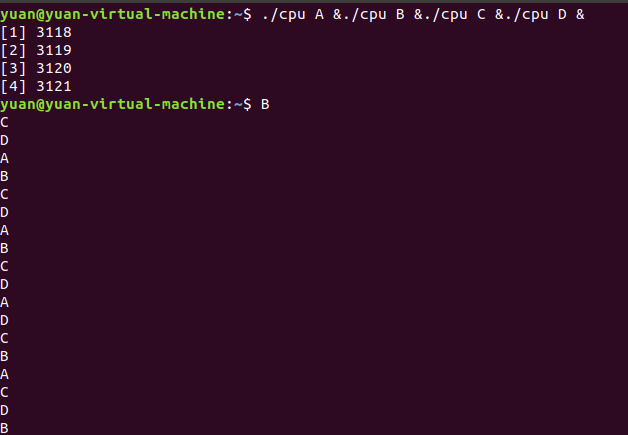
## 1 编译运行该程序



因为该命令的argc=1，所以输出如图所示结果。

从代码中我们可以看出，程序执行后将会输出argv[1]里的内容，而argv数组存储的就是我们在shell中输入的内容，即argv[0]是程序名，argv[1]是参数名，也就是说，这段代码执行后将会输出我们输入的参数。由于后续我们是同时运行了4个参数不同的程序，所以我们可以通过输出的内容判断程序并发执行时，进行的先后顺序。

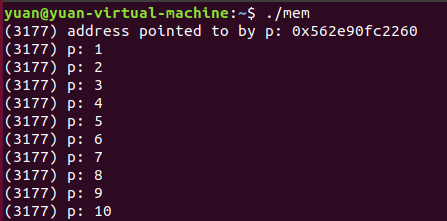
## 2 按下面命令运行并观察结果



CPU运行了4次，程序运行的顺序是随机无序的。这是因为这四个程序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

## 1 观察输出结果，说明程序功能



从这段代码中我们可以看出，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。

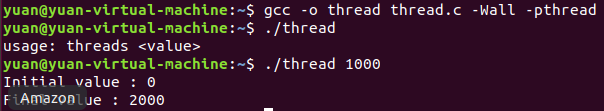
## 2 再次按下面的命令运行并观察结果



两个分别运行的程序分配的内存地址不同，没有共享同一块物理内存区域。因为这相当于两个不同的程序在分别执行。

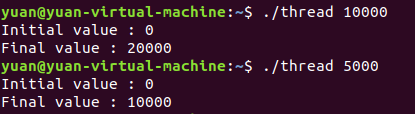
# 四共享的问题

## 1 观察输出结果，说明程序功能



这段代码创建了两个线程，每个线程均进行了函数worker的运算，即进行了一个loops大小的递加运算，最后输出运算的初始值以及最终结果。

## 2总结规律并解释



规律：Final value的值始终为输入参数的2倍

解释：在程序中，我们输入的参数即为loop的值，因为在程序中我们创建了两个线程p1,p2，每个线程都进行了一个loop的递增循环，因为进行了两次，所以Final Value为输入参数的2倍

## 3分析是否会出现问题

程序中counter变量时共享的，并发执行时并不会带来问题，因为一个线程在访问counter时，另一个线程处于等待状态，不能对counter进行修改