

# 《操作系统》课程作业

|  |  |  |
| --- | --- | --- |
| 学 号： | 16281024 | |
| 姓 名： | 张宗灏 | |
| 专 业： | 计算机科学与技术 | |
| 学 院： | 计算机与信息技术学院 | |
|  |  |  |
|  |  |  |
| 提交日期： | 2019年03月12日 | |

# 实验一：操作系统初步

## 实验题目

一、 （系统调用实验）了解系统调用不同的封装形式。

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？linux系统调用的中断向量号是多少？)。

2、上机完成习题1.13。

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。

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）

2、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。

三、 （内存分配实验）根据以下代码完成实验。

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)

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是否相同？是否共享同一块物理内存区域？为什么？命令：./mem &; ./mem &

四、 （共享的问题）根据以下代码完成实验。

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）（执行命令1：./thread 1000）

2、 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。）

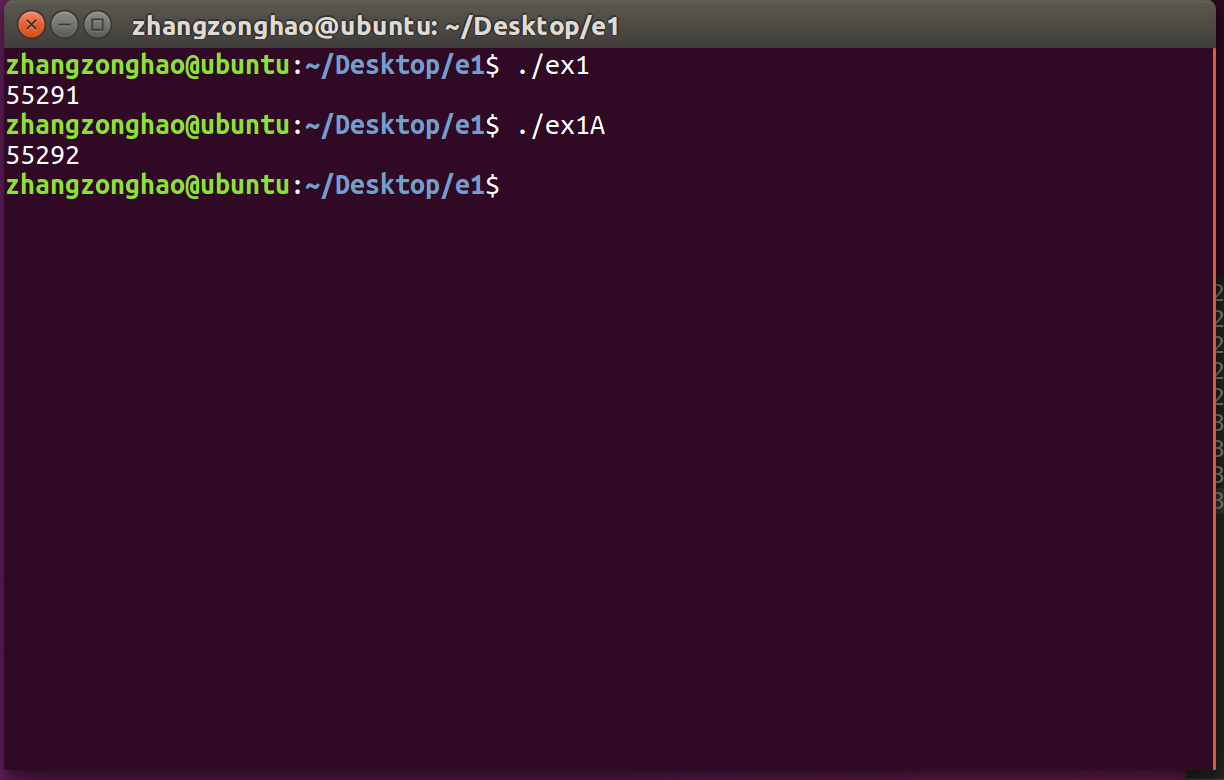
3、 提示：哪些变量是各个线程共享的，线程并发执行时访问共享变量会不会导致意想不到的问题。

## 实验解答

一、

1、 getpid的系统调用号位0x14，Linux系统调用的中断向量号是0x80。

所给代码的程序运行结果如下



2、Linux系统调用实现

使用C语言程序代码如下

#include <stdio.h>

#include <stdlib.h>

#include <unistd.h>

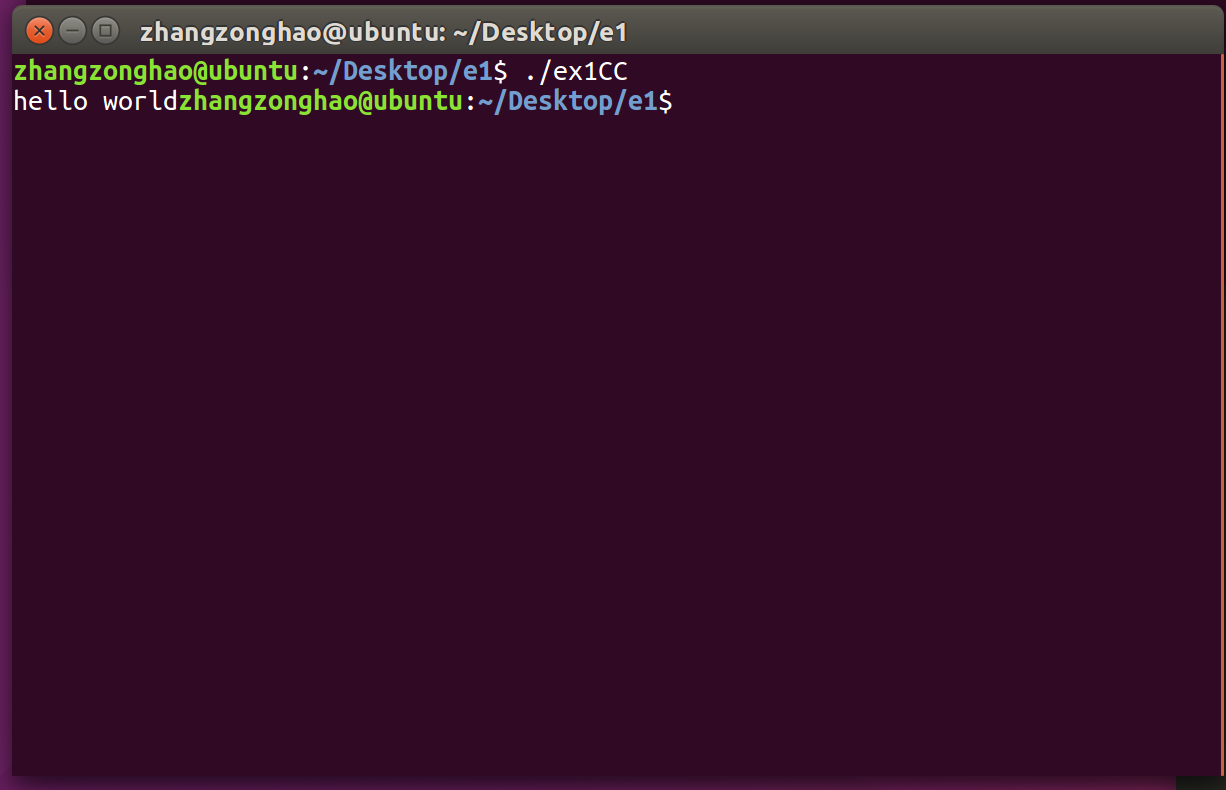
int main(int argc, char \*argv[]){

write(1,"hello world\n",11);

return 0;

}

程序运行结果如下



使用汇编语言程序代码如下

#include <stdio.h>

#include <unistd.h>

void main (){

char const \*MASSAGE = “hello world\n”;

int len = 11;

int result = 0;

asm volatile(

“mov %2, %%edx:\n\r”

“mov %1, %%ecx;\n\r”

“mov $1, %%ebx;\n\r”

“mov $4, %%eax;\n\r”

“int $0x80”

:”=m”(result)

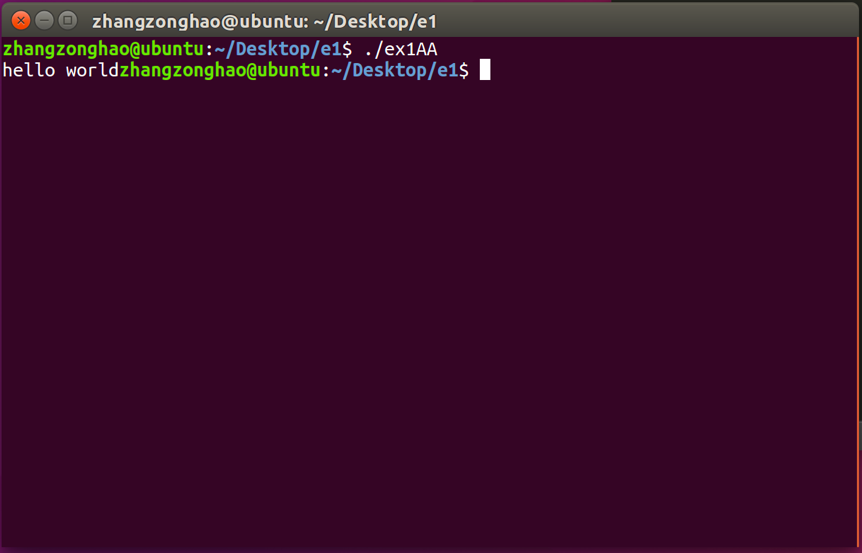
:”m”(msg),”r”(len)

:”%%eax”

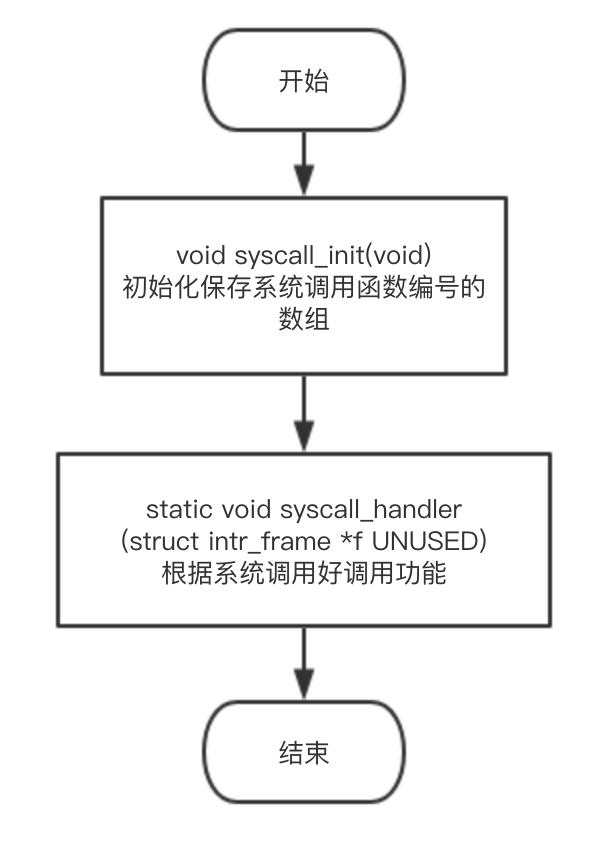
);

}

程序运行结果如下

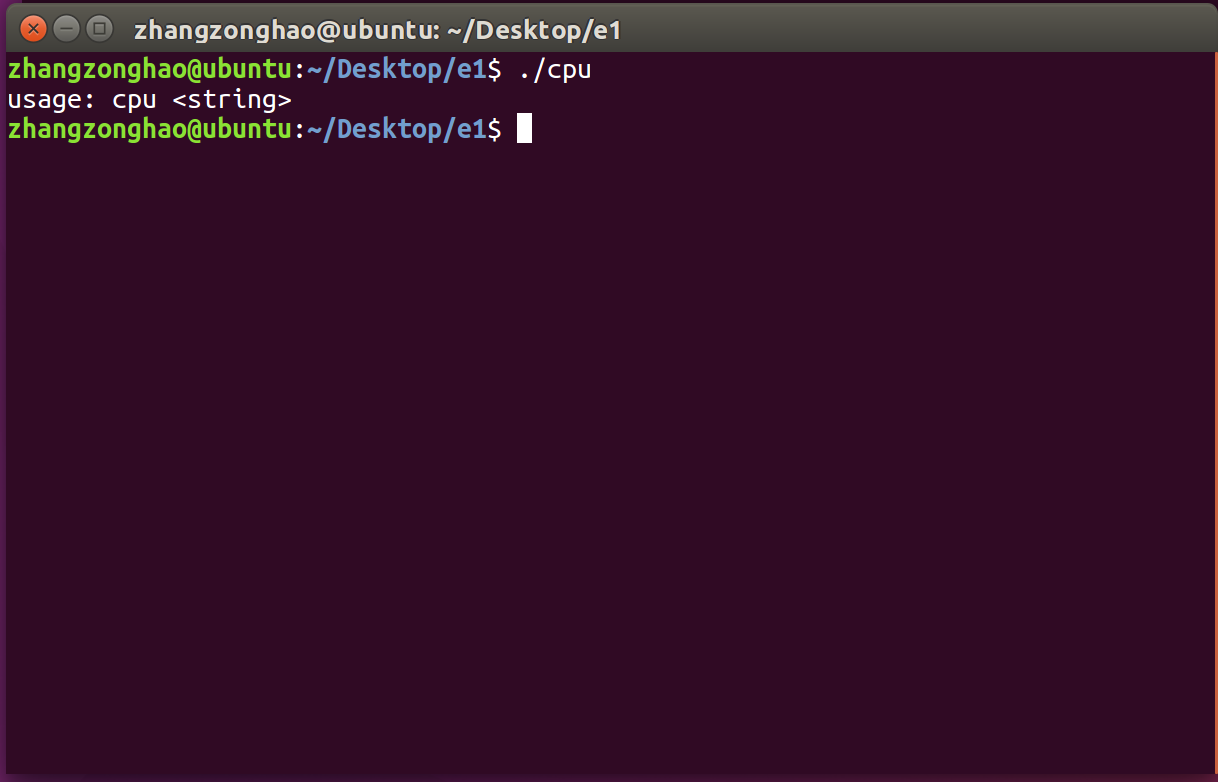


3、通过源代码可知，pintos操作系统的系统调用实现流程图如下



二、

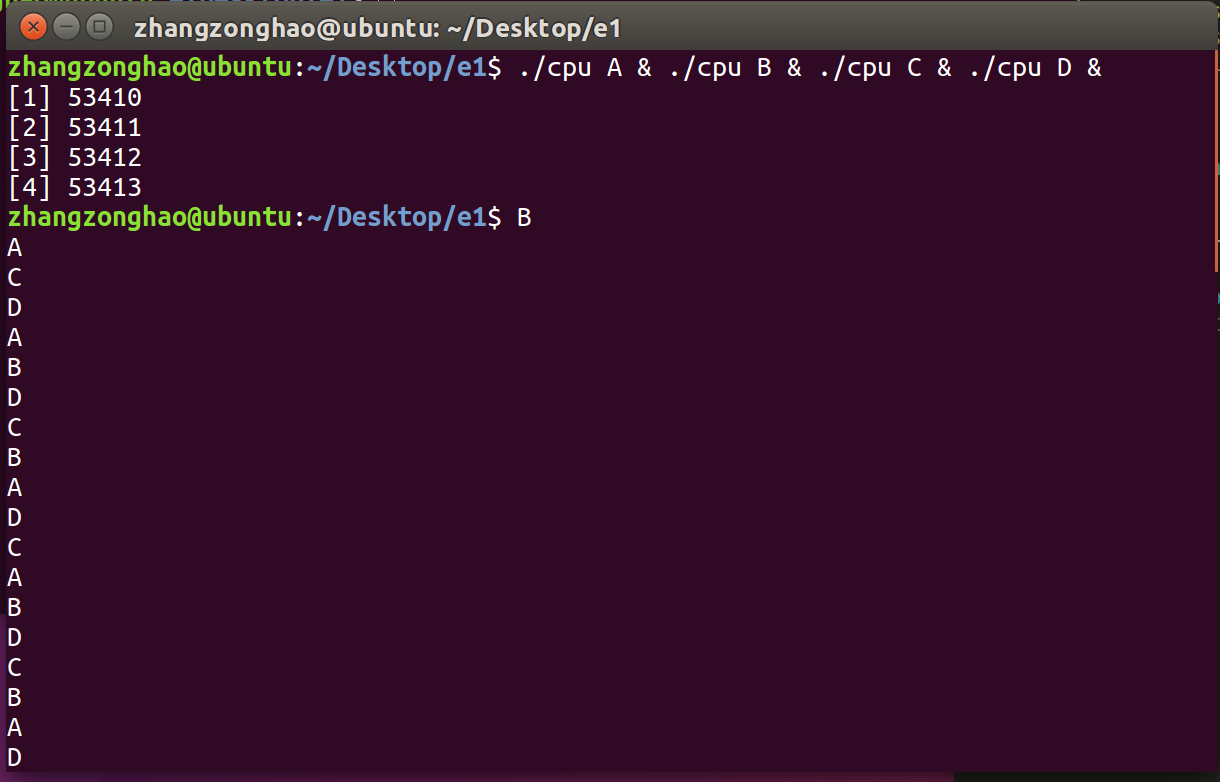
1、使用题中给的编译命令和执行命令，运行结果如下。



根据运行结果并结合代码可以看出，该程序的功能是输出命令行参数的内容。并且如果没有输入参数，那么会输出提示信息——如上图所示。

对代码中问题的解决：common.h头文件改为unistd.h。并且吧spin()函数改为speel()函数。

2、使用题中给的执行命令，运行结果如下。



可以看到有四个程序在同时运行，四个程序的进程号是相邻的。并且输出顺序和参数的输入顺序（ABCD）是不一致的，并且输出顺序是在不断变化的。并且输出是在不断进行的，每次会输出四个字母。

究其原因，四个程序在宏观上是同时运行的，微观上是在CPU中交替运行的。因为四个程序的优先级相同，先进行的程序不一定是先结束，所以这四个程序同时运行时，结束的顺序是不确定的，进而导致输出顺序是不确定的。

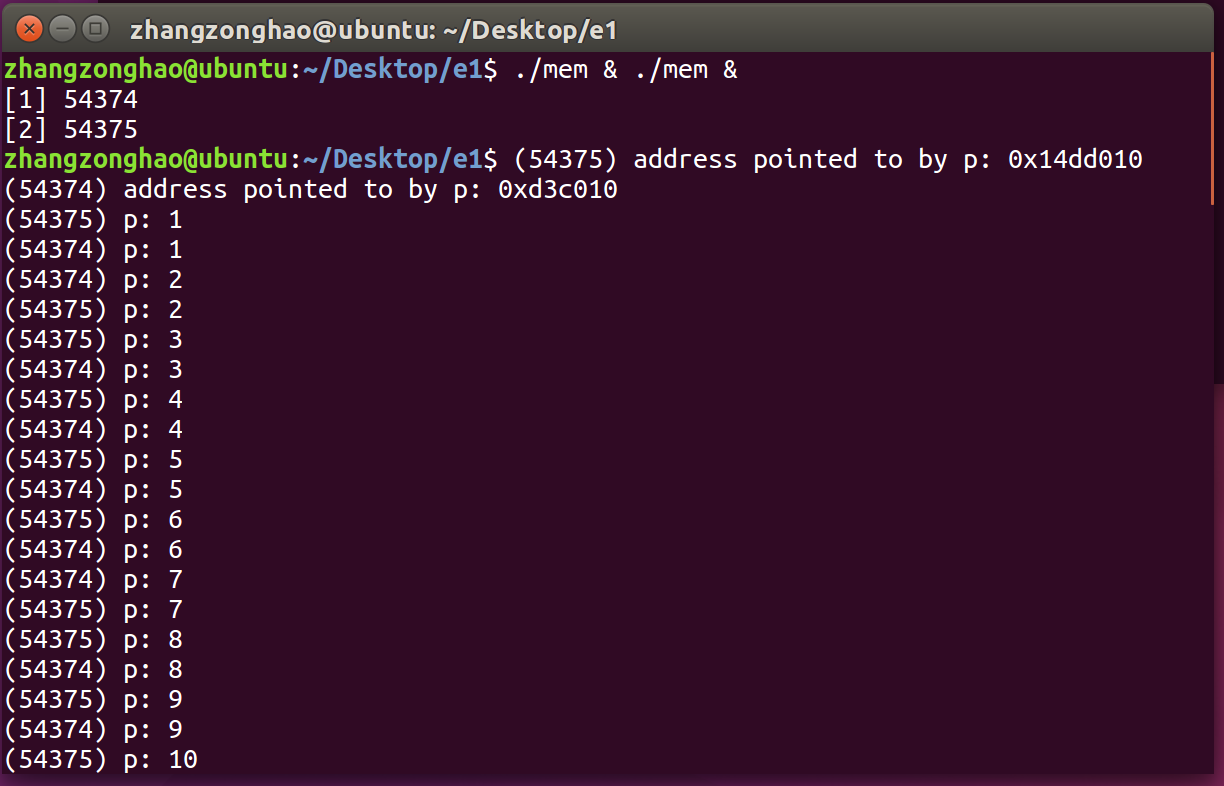
三、

1、 使用所给的编译命令和执行命令，运行结果如下。



首先，打印程序进程号并且输出指针p被分配的内存地址。然后给p所指的内存单元赋值为0。然后在一个每次延时一秒的循环中，对p指向的内存单元中的值进行递增1的操作。并且输出进程号和当前p指向的内存单元中的值。

2、使用所给的编译命令和执行命令，运行结果如下。

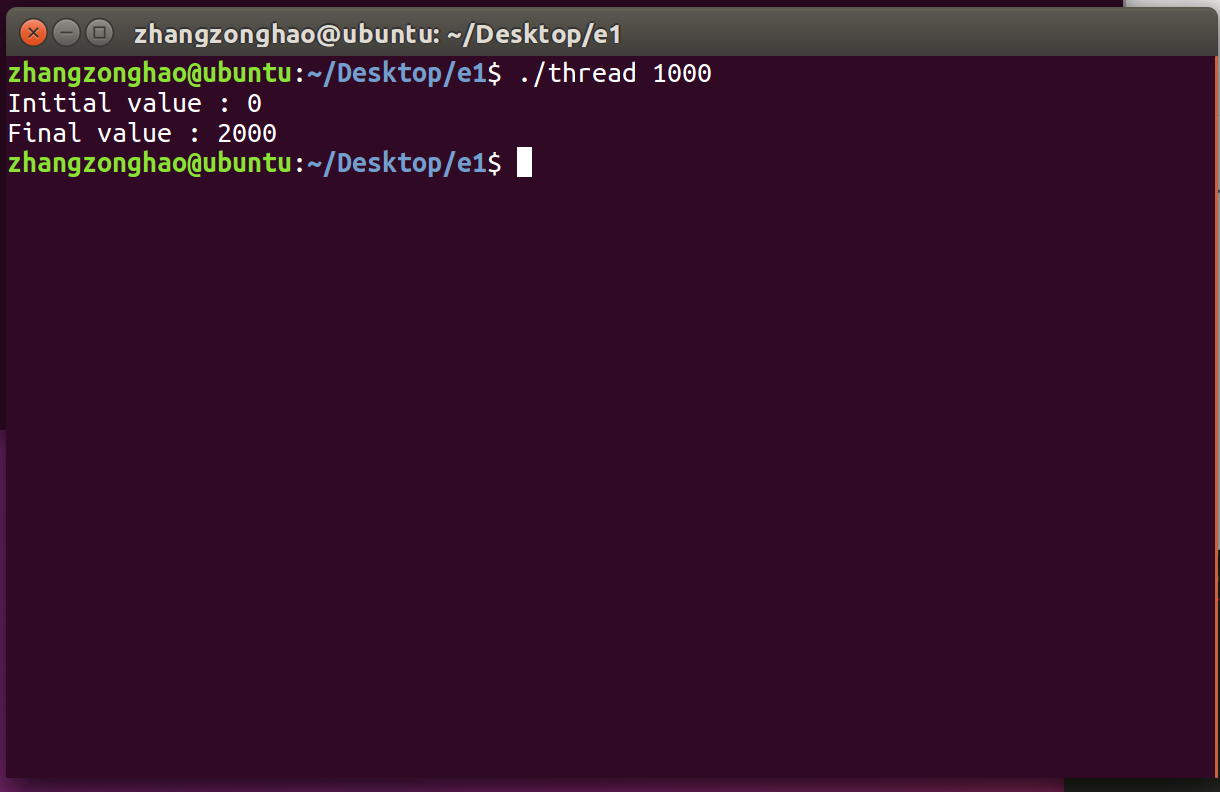


从运行结果来看，两个程序中的指针p指向的内存地址不同并且相距较远。进而可知，两个独立运行的程序分配的内存地址不同，并且不共享一块物理地址。

这是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，然后操作系统通过某种方式把虚拟地址空间映射到机器的物理内存。这样一个程序的内存引用不会影响其他进程（包括OS本身）的地址空间。

四、

1、使用所给的编译命令和执行命令，运行结果如下。



输出的结果中，起始值为0，而最终值2000为输入参数1000的两倍。

程序使用pthread.create()函数创建了两个线程。每个线程中运行一个worker()函数，该函数的作用是循环递增，循环次数上限为输入的参数。这里输入的参数为1000，两个线程中的worker()函数每个把counter的值“加1”1000次，所以counter的最终值为2000。

程序的功能是输入参数为N时，程序最终输出值为2N。

2、不断增大输入的参数的值，会发现得到输出不是2N。

计数器递增，需要三个指令：计数器的值从存储器加载到寄存器中、递增、存储回内存。这三条指令不是同时执行的，所以这些操作之间的执行顺序是不固定的。而counter、 loops这两个全局变量是被这两个线程共享的，当两个的线程一起被调用时，共享全局变量counter。 多线程程序中，同一个变量可能被多个线程修改。所以当参数变得很大时，可能会因为指令调用顺序和参数共享的缘故导致输出结果的错误。