**操 作 系 统**

**实 验 报 告**

**实验名称：实验三　多线程程序实验**

**姓名： 　　　　陈亚楠**

**学号： 　　　　16340041**

**实验名称：多线程程序实验**

**一、实验目的：**

理解与掌握线程的相关概念。

**二、实验要求：**

**1.** 用线程生成Fibonacci数列：

用pthread线程库，按照第四章习题4.11的要求生成并输出Fibonacci数列。

1. 多线程矩阵乘法。

**三、实验过程：**

**实验1.用线程生成Fibonacci数列：**

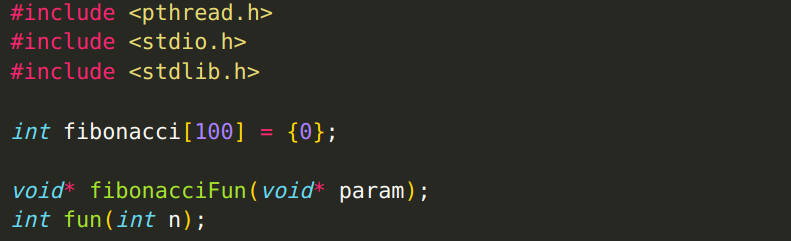
**1.** 实验设计：

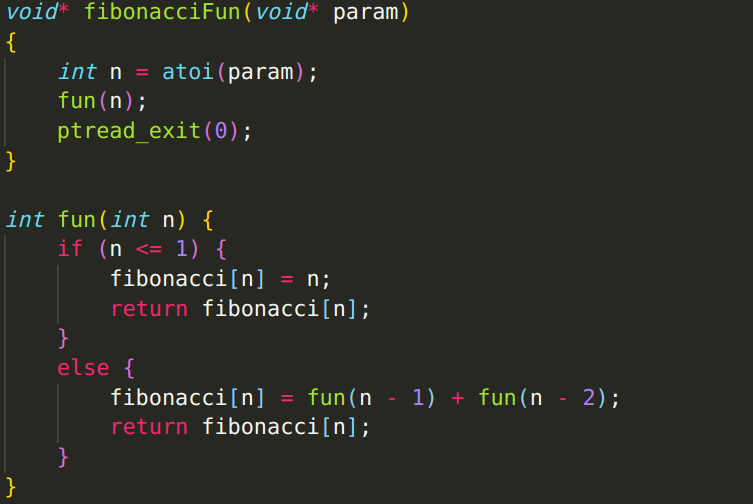
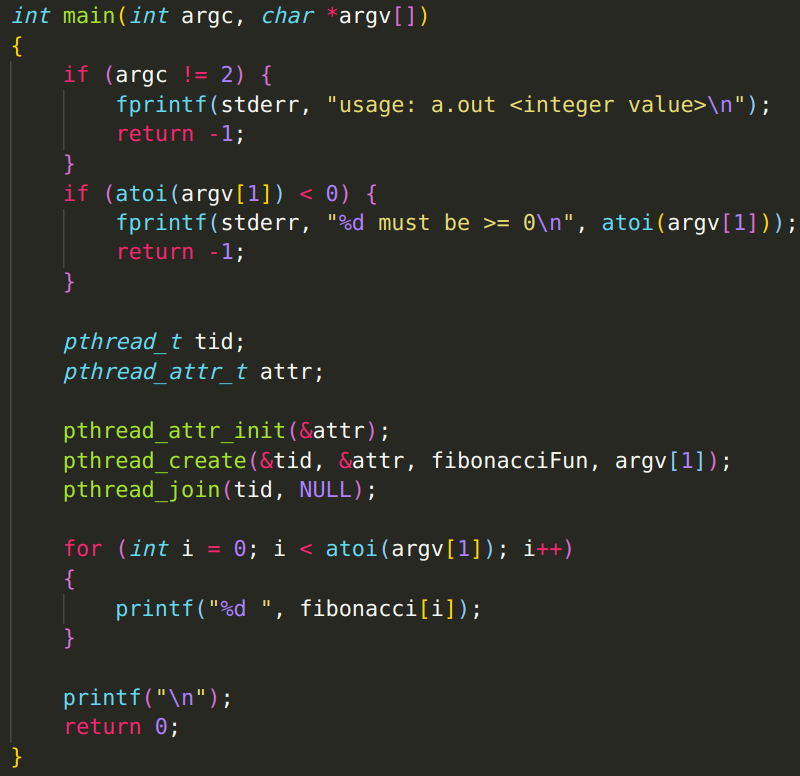
该程序通过一个独立线程计算Fibonacci数列。由于对于Pthread程序，独立线程是通过特定函数执行的，在本实验中，这个特定函数是fibonacciFun() 函数。当程序开始时，单个控制线程在main() 中开始。在初始化之后, main() 创建了第二个线程并在fibonacciFun() 中开始控制。两个线程共享全局数据fibonacci数组。

现在对该程序做一个更为详细的描述。所有Pthread程序都需要包括pthread.h头文件。语句pthread \_t tid声明了所创建线程的标识符。每个线程都有一组属性，包括栈大小和调度信息。pthread\_attr\_t attr表示线程的属性，通过函数调用pthread \_attr\_init(attr) 来设置这些属性。由于该程序不需要明确地设置任何属性，故使用提供的默认属性。通过函数调用pthread\_create() 创建一个独立线程。除了传递线程标识符和线程属性外，还要传递函数名称fibonacciFun以便新线程可以开始执行。最后传递由命令行参数argv[1]所提供的整数参数。

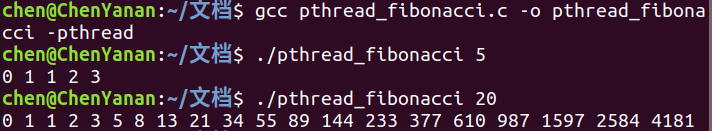
程序此时有两个线程: main()的初始(父)线程和通过fibonacciFun()函数执行计算Fibonacci数列线程。在创建了计算Fibonacci数列线程之后，父线程通过调用 pthread\_join()函数，以等待计算Fibonacci数列线程的完成。计算Fibonacci数列线程在调用了函数 pthread\_exit()之后就完成了。一旦计算Fibonacci数列线程返回，父线程将输出累加和的值。

**2.** 程序代码：





**3.** 实验结果：



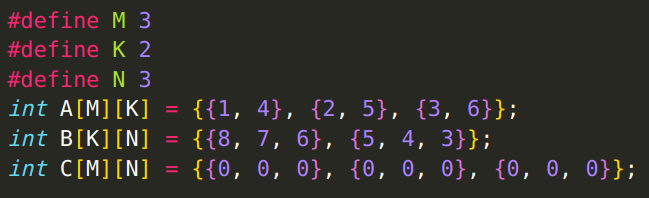
**实验2. 项目乘法：**

**1.** 实验思路：

给定两个矩阵A和B，其中A是具有M行、K列的矩阵， B为K行、N列的矩阵， A和B的矩阵积为矩阵C， C为M行、N列。矩阵C中第i行、第j列的元素Cij就是矩阵A第i行每个元素和矩阵B第j列每个元素乘积的和，即。

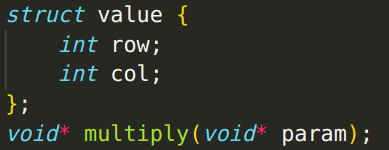
由于每个Ci j的计算用一个独立的工作线程，因此它将会涉及生成M×N个工作线程。主线程(或称为父线程)将初始化矩阵A和B，并分配足够的内存给矩阵C，它将容纳矩阵A和B的积。这些矩阵将声明为全局数据，以使每个工作线程都能访问矩阵A、B和C。

①静态初始化矩阵A、B、C：

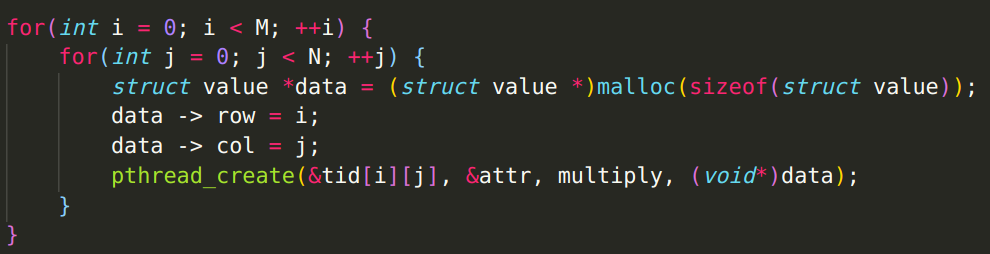


②向每个线程传递参数：

父线程将生成MxN个工作线程，给每个线程传递行i和列j的值，工作线程利用行和列的值来计算矩阵积。这需要向每个线程传递两个参数。在本实验中利用Pthread中的struct生成一个数据结构：



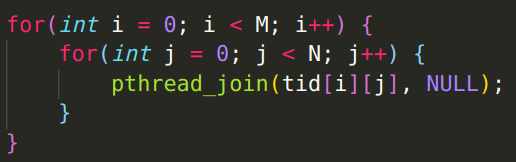
并采用如下算法生成工作线程：



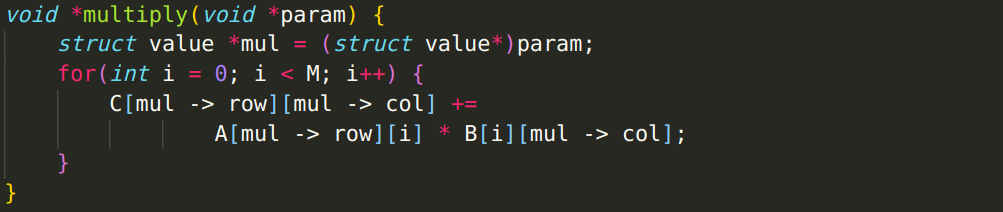
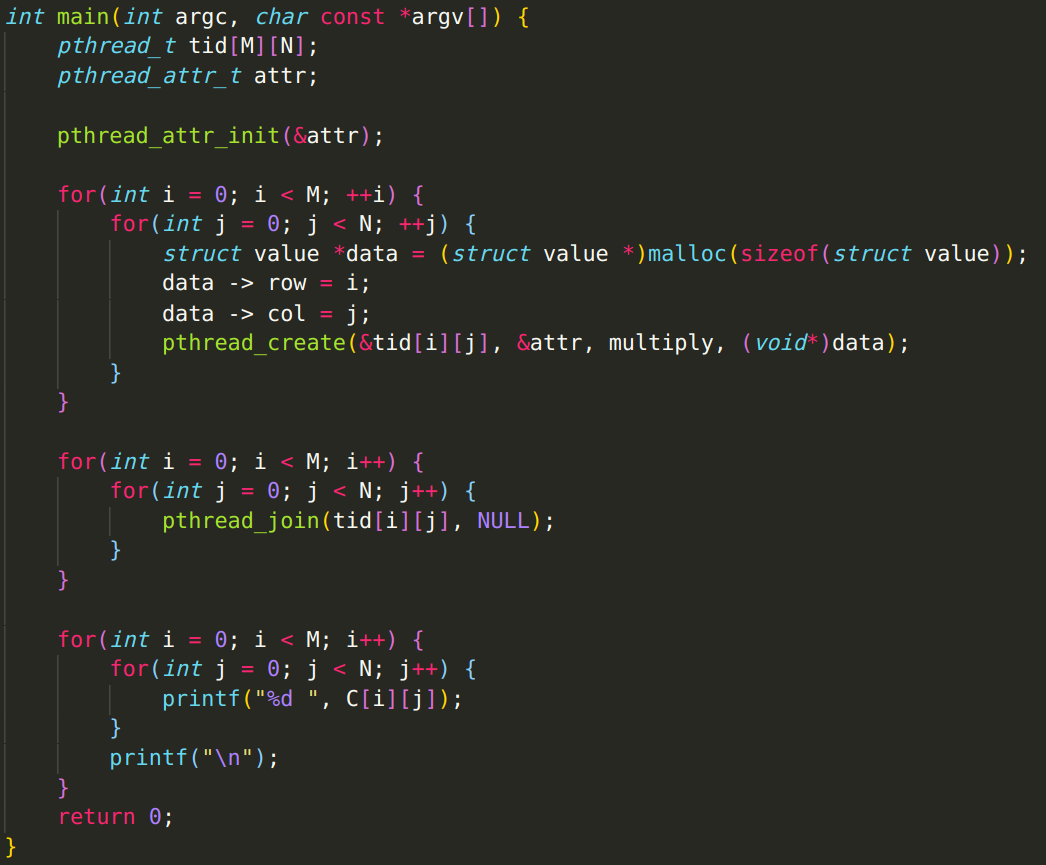
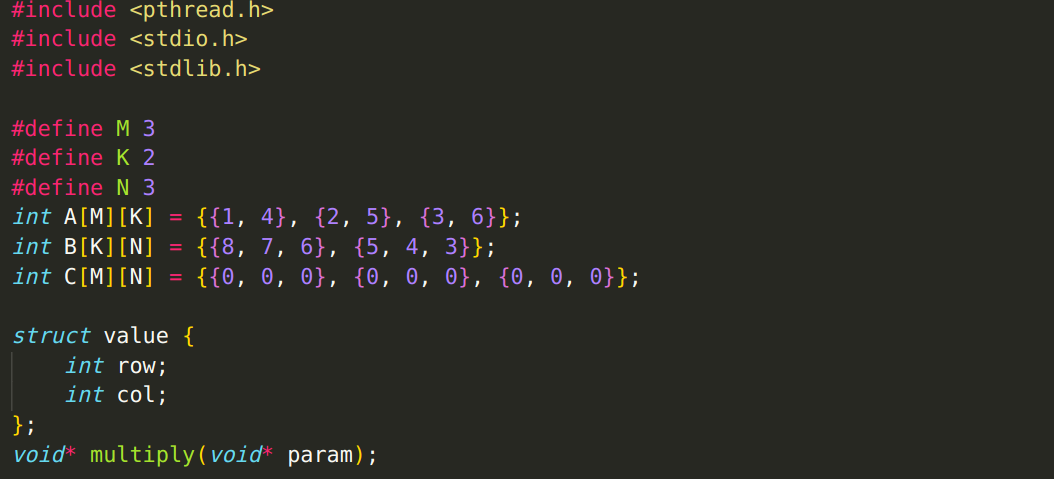
数据指针将被传递给pthread\_create()函数，然后又将它作为参数传递给作为独立线程运行的函数。

③等待线程结束：

一旦所有的工作线程结束，主线程将输出包含在矩阵C中的积。这需要主线程等待所有的工作线程完成其工作，然后才能输出矩阵积的值。本实验用使用下面的方式来使主线程等待其他线程的结束：



**2.** 程序代码：



**3.** 实验结果：

