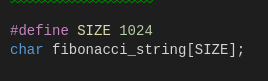
实验2 多线程程序实验

1.用线程生成Fibonacci数列

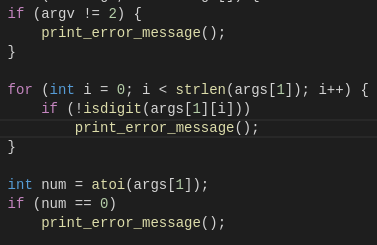
用pthread线程库，按照第四章习题4.11的要求生成并输出Fibonacci数列

我的代码如下：

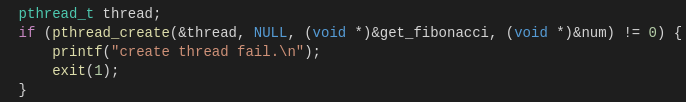
先声明全局变量：



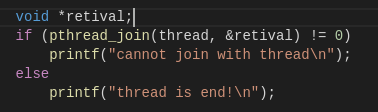
判断输入参数，获得要生成的斐波那契数列的个数num：



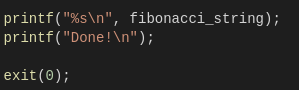
创建线程计算斐波那契数列,并把参数num传递过去：



等待线程完成斐波那契数列的计算：

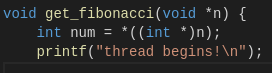


最后等待线程结束后main函数输出斐波那契数列：

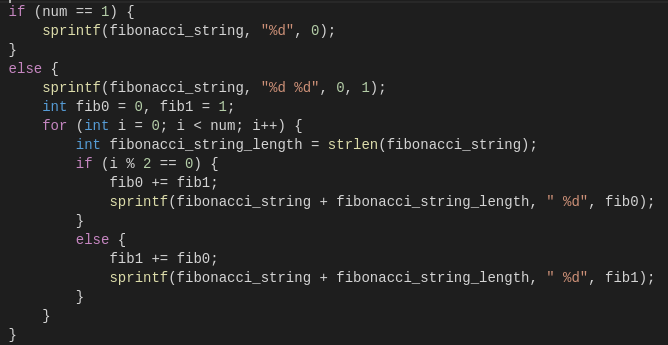


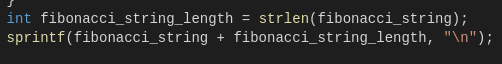
关于斐波那契数列的计算：

先转换参数为int变量：



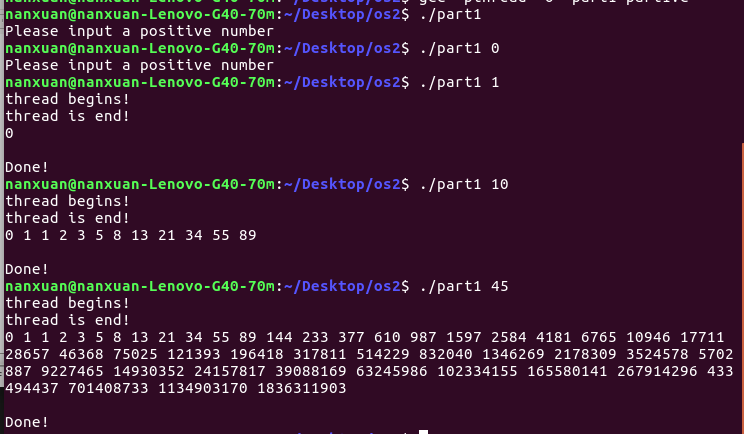
根据num值计算斐波那契数列：

最后加上个换行符



编译的时候一开始是使用ppt给的命令，但是有问题，后来在stackoverflow上找到linux的编译命令：

运行结果如下：



2.多线程矩阵乘法

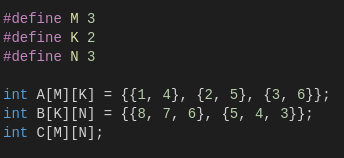
矩阵乘法

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

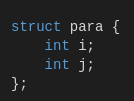
**要求**：每个Ci j的计算用一个独立的工作线程，因此它将会涉及生成M×N个工作线程。主线程(或称为父线程)将初始化矩阵A和B，并分配足够的内存给矩阵C，它将容纳矩阵A和B的积。这些矩阵将声明为全局数据，以使每个工作线程都能访问矩阵A、B和C。

我的代码参考了教材第四章结尾的实验提示，具体如下：

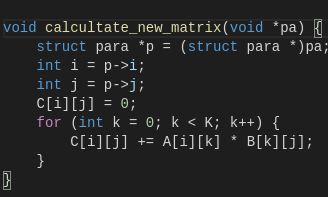
先声明全局变量：



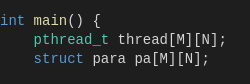
声明一个结构体用来传递多参数：



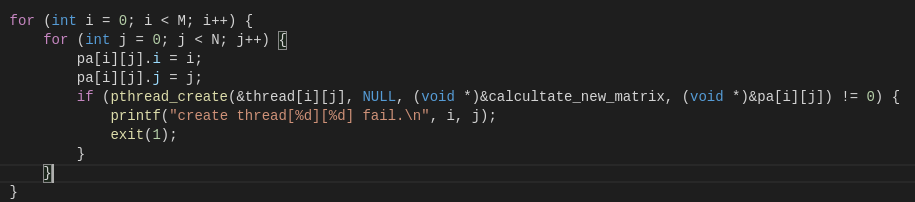
计算矩阵乘法的函数：



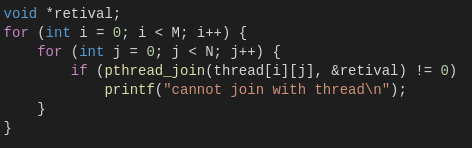
在main函数里声明线程和结构体的二维数组



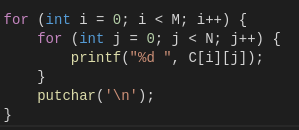
创建线程，每个线程计算矩阵C的一个元素：



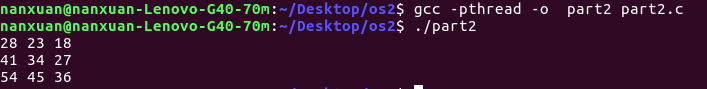
等待所有线程结束：



最后输出矩阵：



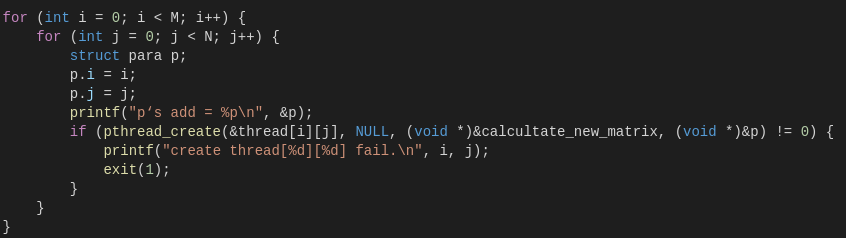
运行结果如下：



经过验证，运行结果正确。

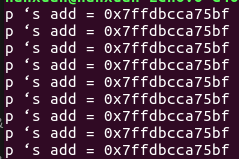
3.实验遇到的问题：

在进行第二个实验的时候，我的代码一开始是这样声明在循环中声明结构体的



这样做，创建的线程无法正确获得参数，后来问了同学才知道是因为在循环中声明的结构体是局部变量，在那一次循环结束后，该变量生命周期结束，该指针指向的值会被销毁，该指针会指向别的结构体

为了证明是不是因为这样，我在创建进程前多加了一句打印出指针地址的语句，运行结果如下：



因此在循环中同一个地址指向不同的值，前面循环结束后，地址会指向第二个循环的值，因此会出现不同线程通过参数获取的值是同一个的现象。

4.实验总结：

通过这次实验，以及在网上查的资料，学习了一些创建多线程的相关知识，在实验中遇到问题应该要多思考，多联系已经学过的知识。