

**课程设计报告**

**题目： 并行编程原理与实践**

**院 系： 计算机科学与技术学院**

**姓 名：**

**专业班级：**

**学 号：**

**指导教师： 陆枫**

**报告日期： 2022年7月5日**

**计算机科学与技术学院**

**目录**

[实验一 1](#_Toc108188935)

[1. 实验目的与要求 1](#_Toc108188936)

[1.1 标题3 1](#_Toc108188937)

[2. 算法描述 1](#_Toc108188938)

[3. 实验方案（含开发与运行环境描述） 1](#_Toc108188939)

[4. 实验结果与分析 1](#_Toc108188940)

[实验二 2](#_Toc108188941)

[1. 实验目的与要求 2](#_Toc108188942)

[2. 算法描述 2](#_Toc108188943)

[3. 实验方案（含开发与运行环境描述） 2](#_Toc108188944)

[4. 实验结果与分析 2](#_Toc108188945)

[实验小结 3](#_Toc108188946)

# 实验一 杨辉三角

## 实验目的与要求

1. 串行环境下的杨辉三角输出

在串行环境下编写打印杨辉三角的C语言小程序

1. OpenMP环境下的杨辉三角输出

使用OpenMP编写打印杨辉三角的C语言小程序。

1. MPI环境下的杨辉三角输出

编写使用MPI打印杨辉三角形的C语言小程序。

## 算法描述

杨辉三角的基本思想是，每一行的第一个和最后一个数为1，每行的第n个数为上一行的第n-1个数和第n个数之和。利用这种方法，我们需要编写一个双循环程序，对每行的每个数（除第一个和最后一个）进行计算。

OpenMP是一种用于共享内存并行系统的多线程程序设计方案，支持C语言，只需要在程序的合适位置加入正确的指令即可简单地实现并行化。在串行程序的基础上，由于一行的数的计算需要依赖于上一行的结果，因此每行的计算无法实现并行化。而一行中的数的计算并不互相关联，因此可以实现行间计算的并行化。

与OpenMP不同，MPI需要我们自己决定各个进程的工作，但其基本思想与OpenMP的方法类似，是不同行的计算串行实现，而一行的计算并行实现。我们可以令进程0为主进程，由主进程向三个子进程发送待求和的两个数、由子进程运算后将结果交还给主进程。

## 实验方案（含开发与运行环境描述）

开发和运行环境：Ubuntu 20.04.4 LTS，使用gcc进行编译，在educoder平台上进行最终评测。

串行：按照1.2所述思路正常编写程序即可。

OpenMP：在串行基础上，双循环的内层循环前加入指令#pragma omp parallel for 即可。

MPI：如前所述，在进程0中读取输入的n值，并负责将待计算的每组被加数循环发送给进程1，2，3，进程1，2，3将得到的数进行相加后发送回进程0。当进程0接收到所有待计算的值后，向进程1~3发送一个结束信号，随后打印结果。

## 实验结果与分析

将编写的代码提交至educoder平台进行评测，均通过测试。但从程序运行时间来看，使用OpenMP和MPI编写的并行杨辉三角运算程序较串行结果并无性能提升，反而性能有所下降。这可能与程序的并行化程度不够高，导致大量时间浪费在进程间通信有关。

# 实验二 并行排序算法实践

## 实验目的与要求

1. 串行环境的排序算法

在串行环境下编写串行排序的C语言小程序，并按要求输出排序结果。

1. pthread环境下的并行排序算法设计

使用pthread实现多线程排序算法的C语言小程序，并按要求输出排序结果。

1. OpenMP环境下的并行排序算法设计

编写使用OpenMP的多线程排序C语言小程序，并按要求输出排序结果。

1. MPI环境下的并行排序算法设计

编写使用MPI的多线程排序C语言小程序，并按要求输出排序结果。

## 算法描述

常见的排序算法有很多，包括冒泡排序、选择排序、快速排序等。但冒泡排序、选择排序等方法时间复杂度较高，为O(n2)，且每步操作间彼此关联，因此可并行程度并不高。而快速排序的方法时间较低，为O(nlogn)，且由于每步操作均会将待排序串分为两部分进行递归排序，因此将其串行化较为容易。本次实验中采用快速排序作为串行排序的结果和并行化修改的基础。

pthread是Linux系统下的一套创建和操纵线程的框架。对于快速排序算法而言，如果每次将序列的两个子序列进行排序的过程均创建新线程，那么最终产生的线程数将随待排序串长度的增长，以指数的形式增长，会带来极大的系统资源的浪费。因此，在本次实验中，我只简单地将被排序串第一次分为两部分的过程使用两个进程进行操作，并在最终将得到的两个串进行归并得到最终结果。

OpenMP是一种用于共享内存并行系统的多线程程序设计方案，支持C语言，只需要在程序的合适位置加入正确的指令即可简单地实现并行化。我们只需要标记出需要并行化的语句，即可将相关操作交由编译器实现，而不需要关注其中的细节。在此，我忽略掉前述的线程数爆炸的问题，而是希望OpenMP进行优化。

MPI与pthread类似，都需要指定每个线程需要进行的工作。我们只需要对pthread程序进行一定的修改，使之符合MPI的形式即可。

## 实验方案（含开发与运行环境描述）

开发和运行环境：Ubuntu 20.04.4 LTS，使用gcc进行编译和测试，最终在educoder平台上进行评测。

串行：使用快速排序算法，编写串行排序程序编写。

pthread：根据2.2所述，将待排序串分为两部分，每部分创建一个新线程调用快速排序函数，两个子线程运行完毕后由主线程将这两个排序完毕的子串进行归并，得到最终结果。

OpenMP：较为简单，在串行程序的基础上，在排序函数递归调用前添加语句#pragma omp parallel sections，在每个递归调用语句前添加语句#pragma omp section即可。

MPI：与pthread类似，0号进程用于接收程序输入，并将待排序串的左右两部分传递给进程1、2，接收进程1、2的计算结果并对结果进行归并得到排序结果；1、2号进程从0号进程处接收待排序字串，调用快速排序函数完成排序后将结果交回进程0。

## 实验结果与分析

将编写的代码提交educoder平台，均能顺利通过测试。根据运行结果，pthread程序运行时间与串行结果较为接近、OpenMP程序除一组测试集时间长于串行外与串行性能相当；而MPI程序运行时间则远长于串行程序。这样的运行结果可能是由于被排序串长度较小，无法发挥并行程序优势。

# 实验小结

在实验中，我们编写了杨辉三角的计算和排序算法两组程序来体验将程序并行化的方法。但从评测结果来看，两组实验虽然均被成功改写为并行化的形式，但由于我并未对算法进行修改，而是沿用了串行化的算法，仅对其局部进行了并行化的改写，导致并行的程序并没有得到有效的性能优化，而是性能持平甚至有所下降。可见，并行编程并非简单地使用工具将一个串行程序改写为并行程序那么简单，而是需要结合算法的优化，将程序的算法改造成并行化程度高、不同线程彼此依赖较少的算法，这样才能尽可能的发挥出并行计算的优势。

此外，由于pthread这类库由Linux系统提供，我也只能在Linux系统下完成对程序的调试，这使我继《计算机系统基础》课程后又一次在集中Linux下使用gdb进行调试，并学习和了解了如何使用gdb进行多线程程序的调试。