并行计算课程报告

姓　　名：	牟鑫一
学　　号：	20161001764
班　　级：	191174
指导老师：	李程俊

一、 题目说明

查找给定范围内的所有素数，不设下界，即最小素数为 2，上界为 10000 的 n（整数）倍，n 为运行程序时传入的参数，为了体现并行的特性，所以取较大的运算量。

二、 程序设计

程序运行时传入线程数以及参数 n，保存到 main()函数默认参数 argv[1]，argv[0]为程序本身，只接受一个参数，所以 argc 的值只允许为 2，若参数个数不为 2 则报错并结束程序。程序查找 1 到 10000 * argv[1]之间的所有素数，通过多进程查找，各尽程记录找到的素数和个数，最终汇总各进程找到的素数以及总的素数个数，输出找到的素数个数并将所有素数写入到文件中保存。

三、 串行源代码

```
#include <stdio.h>
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <vector>

using namespace std;

//checks wheather or not a number is prime with in O(n) time
//判断是否素数
bool isPrime(int num)
{
    if (num < 2)
        return false;
    for (int i = 2; i <= (int)sqrt((double)num); i++) {
        if (num % i == 0)
            return false;
    }
    return true;
}

int n;//上界
int* curPrimes;//临时保存找到的素数
```

```
int k;//素数个数
vector <int> allPrimes;//保存所有素数

int main(int argc, char** argv)
{
    /*
    //打印参数信息
    int x = argc;//传入的参数个数
    printf("传入参数个数: %d\n", x);//格式化输出
    for (int i = 0; i < x; i++)
        cout << "参数" << i + 1 << ": " << argv[i] << endl;
    */

    //check if we have the right number of arguments
    //只接受除运行程序命令 argv[0]外的一个参数 argv[1]
    if (argc != 2)
    {
        cout << "Invalid number of arguments" << endl;
        return 0;
    }
    //calculate n as per requrenment
    n = 10000 * atoi(argv[1]);
    //we cant have more than n/2 primes becuase even numbers for sure cant b
e prime
    //素数不可能超过 n/2,因为偶数不可能是素数
    curPrimes = new int[n / 2];//分配 n/2 空间的数组

    //calculate prime numbers each process will calculate its own portion
    for (int i = 1; i < n; i++)
        if (isPrime(i))
            curPrimes[k++] = i;
    //add the primes calculated here to our large prime vector
    for (int i = 0; i < k; i++)
        allPrimes.push_back(curPrimes[i]);
    //clear the memory of our primes
    delete[] curPrimes;

    string file_name = "primes.txt";
    cout << "\nOverall " << allPrimes.size() << " primes. " << endl;
    cout << "所有素数将写入到文件 " << file_name << " 中！" << endl;
    //output all primes to a file
    FILE* stream;
    freopen_s(&stream, "primes.txt", "w", stdout);
    for (int i = 0; i < allPrimes.size(); i++)
```

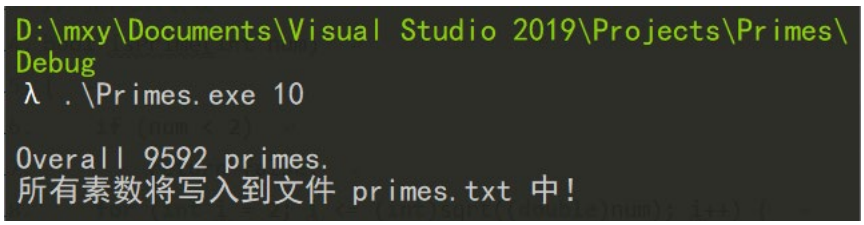
```
    {
        cout << allPrimes[i] << " ";
        if (i % 20 == 0)
            cout << "\n";
    }
    printf("\n");
    return 0;
}
```

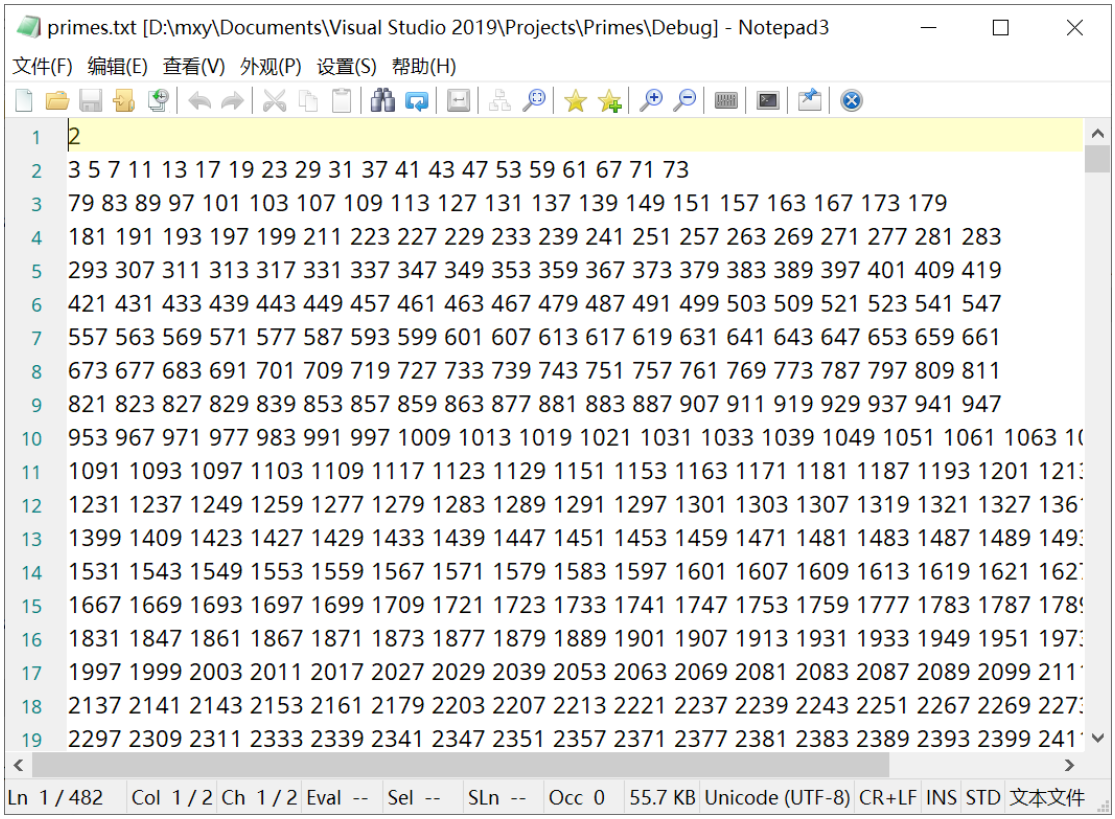
四、 串行测试

测试方法：

在程序所在文件夹路径下，命令行模式输入命令`.\Primes.exe  <input>`，其中<input>为整数 n，即程序所求 10000 * n 以内的素数

测试：100000（10000 * 10）以内的素数





五、 并行源代码

```
#include <mpi.h>
#include <stdio.h>
#include <iostream>
#include <cstdlib>
#include <cmath>
#include <vector>

#define _CRT_SECURE_NO_WARNINGS

using namespace std;

//checks wheather or not a number is prime with in O(n) time
//判断是否素数
bool isPrime(int num)
{
    if (num < 2)
        return false;
    for (int i = 2; i <= (int)sqrt((double)num); i++) {
        if (num % i == 0)
            return false;
    }
    return true;
}

int n;//上界
int world_size; //number of processes involved in our calculations
int* curPrimes;//the primes that we will calculate in THIS process
int k;//素数个数
int* recvPrimes;//temperary variable for primes the main process will recieve from secondary ones
vector <int> allPrimes;//保存所有素数

int main(int argc, char** argv)
{
    /*
    //打印参数信息
    int x = argc;//传入的参数个数
    printf("传入参数个数: %d\n", x);//格式化输出
    for (int i = 0; i < x; i++)
        cout << "参数" << i + 1 << ": " << argv[i] << endl;
    */

```

```
42.     //check if we have the right number of arguments
43.     //只接受除运行程序命令 argv[0]外的一个参数 argv[1]
44.     if (argc != 2)
45.     {
46.         cout << "invalid number of arguments" << endl;
47.         return 0;
48.     }
49.     //calculate n as per requrenment
50.     n = 10000 * atoi(argv[1]);
51.     //we cant have more than n/2 primes becuase even numbers for sure cant b
    e prime
52.     //素数不可能超过 n/2,因为偶数不可能是素数
53.     curPrimes = new int[n / 2];//分配 n/2 空间的数组
54.
55.     //init mpi
56.     MPI_Init(NULL, NULL);
57.
58.     //find out how many processes are there
59.     //当前有多少个进程
60.     MPI_Comm_size(MPI_COMM_WORLD, &world_size);
61.     //find out which process are we in
62.     //找出当前所在进程
63.     int world_rank;
64.     MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
65.
66.     //calculate prime numbers each process will calculate its own portion
67.     for (int i = n * world_rank / world_size; i < n * (world_rank + 1) / wor
    ld_size; i++)
68.         if (isPrime(i))
69.             curPrimes[k++] = i;
70.     //if we are not in the main process(process with rank 0) send our prime
    numbers to main process
71.     if (world_rank != 0)
72.     {
73.         MPI_Send(&k, 1, MPI_INT, 0, 0, MPI_COMM_WORLD);
74.         MPI_Send(curPrimes, k, MPI_INT, 0, 0, MPI_COMM_WORLD);
75.     }
76.     int recvSize;
77.     if (world_rank == 0)
78.     {
79.         //this runs only on main process
80.         //add the primes calculated here to our large prime vector
81.         for (int i = 0; i < k; i++)
82.             allPrimes.push_back(curPrimes[i]);
```

```
        //clear the memory of our primes
        delete[] curPrimes;
        //go through all the recieved primes and add them to our vector
        for (int i = 1; i < world_size; i++)
        {
            MPI_Recv(&recvSize, 1, MPI_INT, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            cout << "Recieved " << recvSize << " primes\nretrieving" << endl;
            recvPrimes = new int[recvSize];
            MPI_Recv(recvPrimes, recvSize, MPI_INT, i, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE);
            for (int j = 0; j < recvSize; j++)
                allPrimes.push_back(recvPrimes[j]);
            delete[] recvPrimes;
        }

        string file_name = "primes.txt";
        cout << "\nOverall " << allPrimes.size() << " primes. " << endl;
        cout << "所有素数将写入到文件 " << file_name << " 中！" << endl;
         //output all primes to a file
         FILE* stream;
         freopen_s(&stream, "primes.txt", "w", stdout);
         for (int i = 0; i < allPrimes.size(); i++)
         {
             cout << allPrimes[i] << " ";
             if (i % 20 == 0)
                 cout << "\n";
         }
         printf("\n");
    }
    MPI_Finalize();
    return 0;
}
```

六、 并行测试

测试方法：

在程序所在文件夹路径下，命令行模式输入命令 `mpiexec -n <number of processes> MPI_Primes.exe <input>`，其中<number of processes>即进程数，<input>为整数 n，即程序所求 10000 * n 以内的素数

测试 1：单进程 10000（10000 * 1）以内的素数

```
D:\mxy\Documents\Visual Studio 2019\Projects\MPI_Pri
mes\x64\Debug (master -> origin)
λ mpiexec -n 1 MPI_Primes.exe 1

Overall 1229 primes.
所有素数将写入到文件 primes.txt 中!
```

测试 2：4 进程 100000（10000 * 10）以内的素数

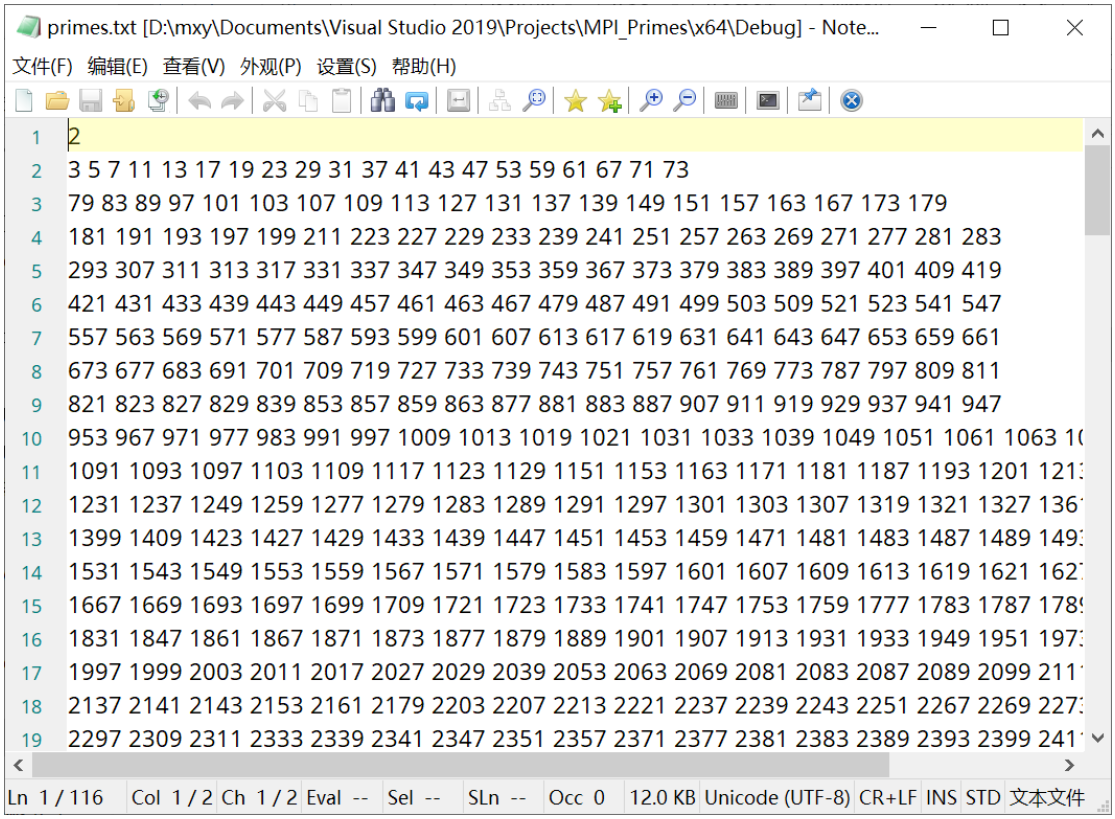
```
D:\mxy\Documents\Visual Studio 2019\Projects\MPI_Pri
mes\x64\Debug (master -> origin)
λ mpiexec -n 4 MPI_Primes.exe 10
Recieved 2371 primes
retrieving
Recieved 2260 primes
retrieving
Recieved 2199 primes
retrieving

Overall 9592 primes.
所有素数将写入到文件 primes.txt 中!
```

测试 3：10 进程 100000（10000 * 10）以内的素数

```
D:\mxy\Documents\Visual Studio 2019\Projects\MPI_Pri
mes\x64\Debug (master -> origin)
λ mpiexec -n 10 MPI_Primes.exe 10
Recieved 1033 primes
retrieving
Recieved 983 primes
retrieving
Recieved 958 primes
retrieving
Recieved 930 primes
retrieving
Recieved 924 primes
retrieving
Recieved 878 primes
retrieving
Recieved 902 primes
retrieving
Recieved 876 primes
retrieving
Recieved 879 primes
retrieving

Overall 9592 primes.
所有素数将写入到文件 primes.txt 中!
```


primes.txt [D:\mxy\Documents\Visual Studio 2019\Projects\MPI_Primes\x64\Debug] - Note...
文件(F) 编辑(E) 查看(V) 外观(P) 设置(S) 帮助(H)
2
3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67 71 73
79 83 89 97 101 103 107 109 113 127 131 137 139 149 151 157 163 167 173 179
181 191 193 197 199 211 223 227 229 233 239 241 251 257 263 269 271 277 281 283
293 307 311 313 317 331 337 347 349 353 359 367 373 379 383 389 397 401 409 419
421 431 433 439 443 449 457 461 463 467 479 487 491 499 503 509 521 523 541 547
557 563 569 571 577 587 593 599 601 607 613 617 619 631 641 643 647 653 659 661
673 677 683 691 701 709 719 727 733 739 743 751 757 761 769 773 787 797 809 811
821 823 827 829 839 853 857 859 863 877 881 883 887 907 911 919 929 937 941 947
953 967 971 977 983 991 997 1009 1013 1019 1021 1031 1033 1039 1049 1051 1061 1063
1091 1093 1097 1103 1109 1117 1123 1129 1151 1153 1163 1171 1181 1187 1193 1201
1231 1237 1249 1259 1277 1279 1283 1289 1291 1297 1301 1303 1307 1319 1321 1327
1399 1409 1423 1427 1429 1433 1439 1447 1451 1453 1459 1471 1481 1483 1487 1489
1531 1543 1549 1553 1559 1567 1571 1579 1583 1597 1601 1607 1609 1613 1619 1621
1667 1669 1693 1697 1699 1709 1721 1723 1733 1741 1747 1753 1759 1777 1783 1787
1831 1847 1861 1867 1871 1873 1877 1879 1889 1901 1907 1913 1931 1933 1949 1951
1997 1999 2003 2011 2017 2027 2029 2039 2053 2063 2069 2081 2083 2087 2089 2099
2137 2141 2143 2153 2161 2179 2203 2207 2213 2221 2237 2239 2243 2251 2267 2269
2297 2309 2311 2333 2339 2341 2347 2351 2357 2371 2377 2381 2383 2389 2393 2399
Ln 1 / 116 Col 1 / 2 Ch 1 / 2 Eval -- Sel -- SLn -- Occ 0 12.0 KB Unicode (UTF-8) CR+LF INS STD 文本文件