Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Операционные системы»

Управление потоками в операционной системе

Студент: Симонов Сергей Яковлевич
Группа: М80 – 206Б-18
Вариант: 22
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2019

Содержание

Постановка задачи

Составить программу на языке СИ, обрабатывающие данные в многопоточном режиме.

Вариант 22: Перемножение полиномов. На вход подается N-- полиномов, необходимо их перемножить.

Общие сведения о программе

Программа компилируется из одного файла main.c. В данной программе используются следующие заголовочные файлы: #include <iostream>,#include <vector>,#include <string>,#include <thread>,#include <mutex>. Используются следующие системные вызовы для работы с потоками:

mtx.lock() – блокировка кода, для последовательного прохода по нему потоками.
mtx.unlock() – разблокировка кода.
vector<thread> – создание вектора потоков.
.join() – блокирует и ждёт завершения порождённого потока.

Общий метод и алгоритм решения

Сначала считывается колличество потоков – N. Затем последовательно считываются полиномы как строка, с помощью функции Get() они конвертируются из строки в целочисленный тип. Одновременно с последовательным считыванием полиномов происходит высчитывание степени. Когда степень посчитана, создается вектор потоков, осуществляющий перемножение полиномов при помощи функции mul(). Конечный полином хранится в векторе. После подсчета конечного полинома он выводится.

Основные файлы программы

main.c

```
#include <iostream>
#include <vector>
#include <string>
#include <thread>
#include <mutex>


using namespace std;

int NOabs(int val)
```

```
{
    if (val >= 0) {
        return -val;
    }
}

vector<int> Get(string &s)
{
    vector<int> result;

    int val = 0, i = 0;
    int flag = 0;

    while (i < s.size()) {
        if ((s[i] >= '0' && s[i] <= '9') || (s[i] == '-')) {
            if (s[i] == '-') {
                flag = 1;
            }
            if (s[i] >= '0' && s[i] <= '9') {
                const int d = s[i] - '0';
                val = val * 10 + d;
            }
        }

        if (s[i + 1] == ' ' || i + 1 == s.size()) {
            if (flag == 0) {
                result.push_back(val);
                val = 0;
            } else if (flag == 1) {
                val = NOabs(val);
                result.push_back(val);
                val = 0;
            }
        }

        ++i;
    }

    return result;
}

mutex mtx;

void* mul(vector<int> &a, vector<int> &b, vector<int> &c, int j)
{
```

```
    mtx.lock();

        j -= 1;

        for (int i = 0; i < b.size(); ++i) {
            c[i + j] += a[j] * b[i];
        }
    mtx.unlock();
}

int main()
{
    string str, str1;
    string a;
    int n;
    cin >> n;
    vector<int> res;
    vector<int> res1;
    vector<int> result;
    vector<int> result1;

    vector<int> ressss;

    vector<string> vec;
    int step = 0, delta = 0;
    for (int i = 0; i <= n; ++i) {
        getline(cin, a);

        str1 = str;
        str = a;

        vec.push_back(a);

        res = Get(str);
        res1 = Get(str1);

        step += res.size();

        if (i >= 2) {
            step -= 1;
        }

        if ((i == 1) && (n == 1)) {
            cout << vec[i - 1] << " " << endl;
            break;
```

```
}

for (int j = 0; j < step - result.size() + 1; ++j) {
   result.push_back(0);
}
if (i == 2) {

   vector<thread> re;
   re.reserve(result.size());

   for(int j = 0; j < result.size(); ++j) {

      re.emplace_back([&res, &res1, &result, j] {
         mul(res, res1, result, j);
      });
   }

   for(int l = 0; l < result.size(); ++l) {
      re[l].join()    ;
   }
}

if (i >= 3) {

   vector<thread> re;
   re.reserve(result.size());

   res1.clear();

   for (int s = 0; s < result.size(); ++s) {
      res1.push_back(result[s]);
      result[s] = 0;
   }


   for (int j = 0; j < result.size(); ++j) {

      re.emplace_back([&result, &res, &res1, j] {
         mul(res1, res, result, j);
      });
   }


   for (int l = 0; l < result.size(); ++ l) {
      re[l].join();
```

```
            }

            re.clear();
        }
    }

    for (int i = 0; i <= n; ++i) {
        cout << vec[i] << " " << endl;
    }

    cout << endl;

    if (n >= 2) {
        for (int j = 0; j < result.size(); ++j) {
            if (result[j] != 0) {
                cout << result[j] << " ";
            }
        }
    } else if (n == 1) {
        cout << vec[n] << " " << endl;
    }

    cout << endl;

    return 0;
}
```

Демонстрация работы программы

```
sergey@sergey-RedmiBook-14:~/labs/OS/lab3$ ./a.out
4
1 1 1
1 1
1 1 1
1 1 1

1 1 1
1 1
1 1 1
1 1 1

1 4 9 13 13 9 4 1
sergey@sergey-RedmiBook-14:~/labs/OS/lab3$ ./a.out
1
1 0 1 0 1 0 1
```

```
1 0 1 0 1 0 1

1 0 1 0 1 0 1

sergey@sergey-RedmiBook-14:~/labs/OS/lab3$ ./a.out
3
1 2 3
1 2 3
4 5 6 7 8 9

1 2 3
1 2 3
4 5 6 7 8 9

4 21 66 129 192 228 254 249
sergey@sergey-RedmiBook-14:~/labs/OS/lab3$ ./a.out
2
1 1
1 1

1 1
1 1

1 2 1


sergey@sergey-RedmiBook-14:~/labs/OS/lab3$ strace -c ./a.out
5
1 1
1 1
1 1
1 1
1 1

1 1
1 1
1 1
1 1
1 1

1 5 10 10 5 1
% time     seconds  usecs/call     calls    errors syscall
------ ----------- ----------- --------- ---------
----------------
25.36    0.000791          33        24           mmap
17.79    0.000555          31        18           clone
11.80    0.000368          19        19           mprotect
 7.34    0.000229          33         7         7 access
 6.86    0.000214          19        11           read
 6.64    0.000207          26         8           write
 6.51    0.000203          34         6           openat
```

```
 5.13    0.000160          20         8           fstat
 4.04    0.000126          32         4           munmap
 3.91    0.000122          20         6           close
 2.66    0.000083          83         1           execve
 1.15    0.000036          12         3           brk
 0.80    0.000025          25         1           arch_prctl
 0.00    0.000000           0         2           rt_sigaction
 0.00    0.000000           0         1           rt_sigprocmask
 0.00    0.000000           0         2           futex
 0.00    0.000000           0         1           set_tid_address
 0.00    0.000000           0         1           set_robust_list
 0.00    0.000000           0         1           prlimit64
------ ----------- ----------- --------- ---------
----------------
100.00    0.003119                   124         7 total
```

Вывод

Одно из главных отличий использования многопоточности от использования процессов состоит в том, что потоки делят между собой одно адресное пространство, но параллельная обработка одних и тех же данных требует требует ответственного подхода к написанию программы. Итак, я научился правильно пользоваться многопоточным программированием т.е. своевременно блокировать потоки дабы не происходило смешение данных, что влияет на конечный результат. Разобрал быстрое преобразование Фурье, но к огромному сожалению запрограммировать его не получилось.