Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Студент: Николаева Елизавета Сергеевна

Группа: М8О-201Б-20

Вариант: 15

Преподаватель: Миронов Евгений Сергеевич

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/esnikolaeva/OS\_3

**Постановка задачи**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработки использовать стандартные средства создания потоков операционной системы (Windows/Unix). Ограничение потоков должно быть задано ключом запуска вашей программы.

Вариант 15) Перемножение полиномов. На вход подаётся N-полиномов, необходимо их перемножить.

**Общие сведения о программе**

Вся программа содержится в одном файле laba3.cpp

**Общий метод и алгоритм решения**

Считываются следующие данные: количество полиномов и их степени, количество потоков, затем коэффициенты этих полиномов, причем сначала первого и второго, затем добавляя по одному новому в процессе работы программы. Для удобства введен новый результирующий полином.

Далее запускается некоторое количество потоков (не больше N), в которые передаётся функция new\_pol. Распараллеливание происходит по следующему правилу – каждый новый поток работает с коэффициентами кратными данному номеру потока.  
new\_pol – функция, применяющая коэффициенты 3-х полиномов и их размеры, а также номер потока и их количество. Данная функция вычисляет новые коэффициенты после перемножения, записывая их в результирующий полином.

**Исходный код**

#include<iostream>

#include<fstream>

#include<thread>

#include<vector>

#include<malloc.h>

#include<unistd.h>

using namespace std;

void dosmth(){

sleep(10);

}

void free\_pol(int \*result, int \*pol, int size){

for (int i = 0; i<size; i++){

pol[i] = result[i];

result[i] = 0;

}

}

void print\_pol(int \*polynomial, int size){

for (int i = size-1; i >0; i--){

cout<<polynomial[i]<<"x^"<<i<<"+";

}

cout<<polynomial[0]<<endl;

}

void new\_polyn(int \*result, int \*minm, int \*maxm, int n, int m, int K, int np){

// cout<<"!np="<<np<<"! n="<<n<<" m="<<m<<" K="<<K<<" np="<<np<<endl; fflush;

for(int i = 0; i<n; i++) {

//cout<<"!np="<<np<<" jbegin="<<(K-(i%K)+np)%K<<endl; fflush;

for(int j = (K-(i%K)+np)%K ; j < m; j = j + K){

// cout<<"!np="<<np<<"! i="<<i<<" j="<<j<<" K="<<K<<endl;fflush;

result [i+j] = result[i+j] + (minm[i] \* maxm[j]);

// cout<<"!np="<<np<<"! i="<<i<<" result="<<result[j]<<endl;fflush;

}

}

}

int main(){

int N;

int answere\_degree=0;

cout<<"please, enter the number of polynominals"<<std::endl;

cin>>N;

if (N<2){

cout<<"Error!";

return 1;

}

int K;

cout<<"please, insert number of threads. Insert -1 if there are no restrictions:"<<endl;

cin>>K;

if (!K || (K<0 && K != -1)){

cout << "Error"<<endl;

return 1;

}

if (K == -1) {K = 2;}

int a, max; max = -1;

cout<<"please,enter degree of first polynominal"<<endl;

cin>>a;

int idea2[N];

idea2[0]= a;

answere\_degree = a ;

for (int j = 1;j < N; j++){

cout<<"please, enter degree of next polynomial" << endl;

int degree;

cin>>degree;

if (degree > max) {max = degree;}

idea2[j] = degree;

answere\_degree += (degree-1) ;

}

// for(int i = 0; i<N; i++){

// cout<<"degree:"<<i<<"="<<idea2[i]<<endl;

// }

// cout<<"\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*"<<endl;

int \*pol1 = new int [max];

//cout << "answere degree="<<answere\_degree<<endl;

int \*pol = new int[answere\_degree];

cout<<"please, enter coefficients of first polynominal, starting with the oldest"<<endl;

for (int i = a-1; i>=0; i--){

cin>>pol[i];

}

fflush;

int \*result = new int[answere\_degree];

for(int i = 0; i<answere\_degree; i++){

result[i] = 0;

}

// for (int i = a-1 ; i >=0; i--){

// result[i] = pol[i];

// }

// cout<<"test1"; fflush;

int k = 1;

if (N!=1){

thread th[K];

while (N > 1){

int b;

cout << k << endl;

b = idea2[k];

k = k + 1;

cout<<"please, enter coefficients of " << k <<" polynominal, starting with the oldest"<<std::endl;

for (int i = b-1; i>=0; i--){

cin>>pol1[i];

}

cout<<" "; print\_pol(pol, a);fflush;

cout<<"\*"<<endl;

cout<<" ";print\_pol(pol1, b); fflush;

cout<<"\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_"<<endl;

//cout << "test K="<<K<<" a="<<a<<endl; fflush;

/\*for (int i = 0 ; i<a; i++){

cout<<"pol["<<i<<"]="<<pol[i]<<endl;

}

for (int j = 0; j<b; j++){

cout<<"pol1["<<j<<"]="<<pol1[j]<<endl;

}

cout <<"result:";

for (int i = 0; i<answere\_degree; i++){

cout <<result[i]<<" ";

}

cout<<endl;

cout <<"answere\_degree="<<answere\_degree<<endl;\*/

for (int i = 0; i<K; i++){

if (a>b){

th[i] = thread(new\_polyn, result, pol, pol1, a, b, K, i);

//cout<<"i="<<i<<"=>";

}

else{

th[i] = thread(new\_polyn, result, pol1, pol, b, a, K, i);

//cout<<"2i="<<i<<"=>";

}

}

for(int i = 0; i<K; i++){

th[i].join();

}

//cout<<"after join"<<endl; fflush;

for (int i = b-1; i>=0; i--){

pol1[i] = 0;

}

a = a + b - 1;

N = N - 1;

print\_pol(result, answere\_degree);

free\_pol(result, pol, answere\_degree);

}

}

delete[] result;

delete[] pol;

delete[] pol1;

}

**Выводы**

Я приобрела навыки в управлении потоками в ОС Unix и распараллеливании выполнения сложных задач для увеличения производительности.