**Московский авиационный институт**

**(Национальный исследовательский университет)**

Факультет: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

**Лабораторная работа № 3**

**По курсу:  
«Операционные системы»**

Студент: Патрикеева Лидия

Группа: 80-207Б-18

Преподаватель: Миронов Е.С.

Дата:

Оценка:

Москва, 2019

**Цель работы**

Приобретение практических навыков в:

* Управление потоками в ОС
* Обеспечение синхронизации между потоками

**Задание**

Составить программу на языке Си, обрабатывающую данные в многопоточном режиме. При обработке использовать стандартные средства создания потоков операционной системы (Windows/Unix). При создании необходимо предусмотреть ключи, которые позволяли бы задать максимальное количество потоков, используемое программой. При возможности необходимо использовать максимальное количество возможных потоков. Ограничение потоков может быть задано или ключом запуска вашей программы, или алгоритмом.

Так же необходимо уметь продемонстрировать количество потоков, используемое вашей программой с помощью стандартных средств операционной системы.

Вариант задания № 9:

Расчет n точек, заданных функцией. На вход подается количество точек и границы отрезка на котором производится расчет функции. Функция f(Xn) = (sin(Xn)+cos(Xn))\* f(Xn-1), f(X0) = sin(X0)+cos(X0). Необходимо, чтобы при выводе точки были выведены последовательно относительно x.

**Исходный код**

lab3.c

#include <stdlib.h>

#include <stdio.h>

#include <windows.h>

#include <math.h>

HANDLE \*thread;

DWORD threadId;

double intervalBetweenPoints;

int nPointsForThread;

int nPoints;

double \*fun;

double a;

DWORD WINAPI ThreadFunc( LPVOID arg ){

int number = (int)arg;

if ( number > nPoints ) return 0;

if(number == 0){

double currentPoint = a;

fun[0] = sin(currentPoint) + cos(currentPoint);

currentPoint += intervalBetweenPoints;

for (int i = 1; i < nPointsForThread; i++){

fun[i] = ( sin(currentPoint) + cos(currentPoint) ) \* fun[i-1];

currentPoint += intervalBetweenPoints;

}

}

else{

int nPointsCounted = nPointsForThread \* number;

double currentPoint = a + intervalBetweenPoints \* nPointsCounted;

double mas[nPointsForThread];

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

mas[i] = sin(currentPoint) + cos(currentPoint);

currentPoint += intervalBetweenPoints;

}

nPointsCounted = nPointsForThread \* number;

WaitForSingleObject(thread[number-1], INFINITE);

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

if(nPointsCounted > nPoints - 1 ) break;

fun[nPointsCounted] = mas[j] \* fun[nPointsCounted - 1];

nPointsCounted ++;

}

}

return 0;

}

int main(int argc, char \*argv[]){

double b;

printf("Enter number of points: ");

scanf("%d", &nPoints);

printf("Enter interval: ");

scanf("%lf %lf", &a, &b);

int nThreads = nPoints;

if( b < a){

double x = a;

a = b;

b = x;

}

intervalBetweenPoints = (b - a + 1) / (double)nPoints;

if (argc >= 2) {

nThreads = atoi(argv[1]);

}

nPointsForThread = max(nPoints / nThreads, 1);

fun = (double\*)malloc(sizeof(double)\*nPoints);

thread = (HANDLE\*)malloc(sizeof(HANDLE) \* nThreads);

DWORD \*threadId = (DWORD\*)malloc(sizeof(DWORD) \* nThreads);

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

thread[i] = CreateThread(NULL, 0, &ThreadFunc,(LPVOID)i, 0, &threadId[i]);

}

printf("Function points : \n");

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

WaitForSingleObject(thread[i], INFINITE);

CloseHandle(thread[i]);

for(int j = i \* nPointsForThread; j < (i + 1) \* nPointsForThread && j < nPoints; j++){

printf("%d) ", j+1);

printf("%.25lf\n", fun[j]);

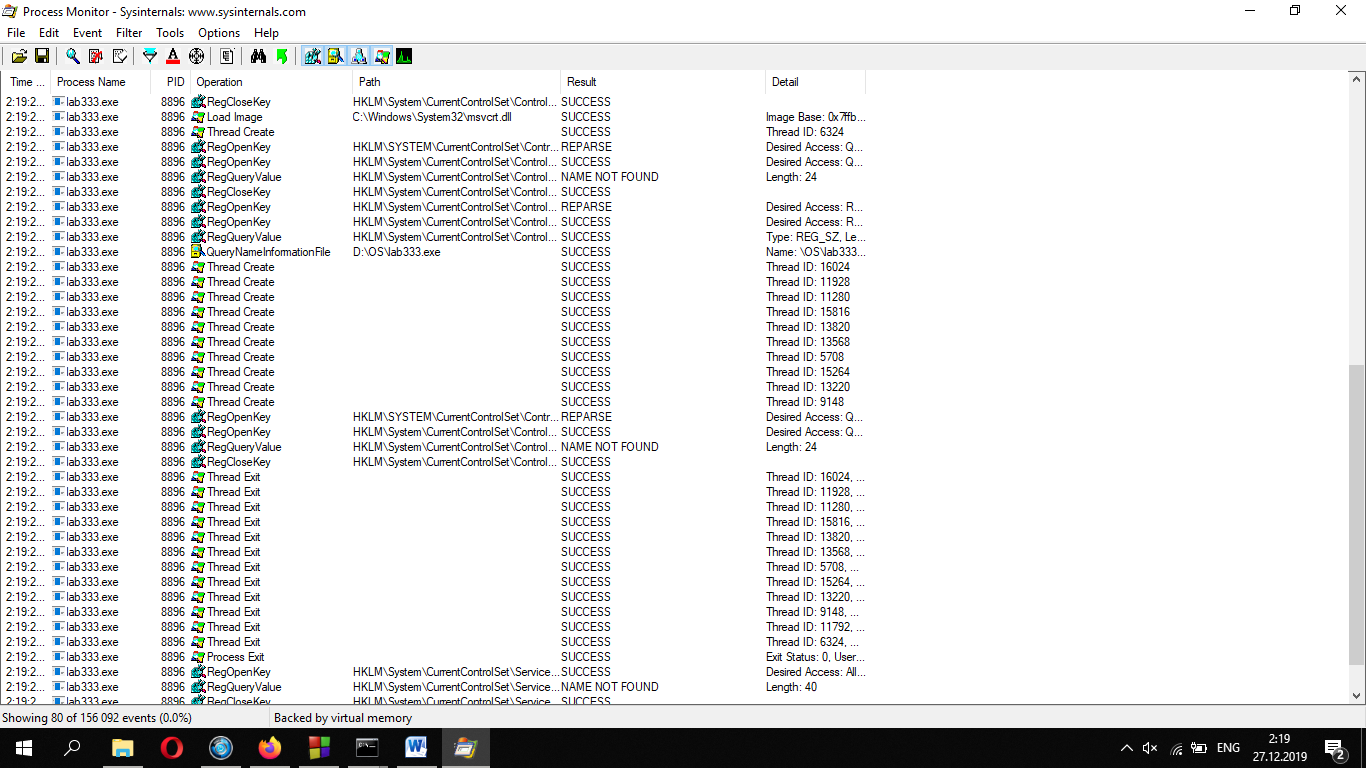
}

}

return 0;

}

**Procmon**

Ниже представлен скриншот из программы Process Monitor, которая позволяет наблюдать в реальном масштабе времени за действиями различных процессов в среде операционной системы Windows.

**Вывод**

В процессе выполнения данной лабораторной работы я научилась распараллеливать выполнение программы. Научилась работать и управлять несколькими потоками в рамках одного процесса, а так же обеспечивать синхронизацию между ними. Для решения задачи я научилась пользоваться функцией WaitForSingleObject, которая была необходима мне для того, чтобы дожидаться завершения нужного мне потока.

Из данной лабораторной работы могу сделать вывод, что использование многопоточности в программе бывает полезно. Такой способ позволяет разбить задачу на несколько более мелких задач, которые будут выполняться парралельно. Так же создать несколько потоков намного эффективнее, чем создать несколько процессов. Потоки делят одно адресное пространство и работают намного быстрее.