**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные системы»

**Лабораторная работа № 3**

Тема: Управление процессами в ОС

Студент: Калугин К. А.

Группа: М8О-207Б-19

Преподаватель: Миронов Е.С.

Дата:

Оценка:

Москва, 2021

**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.

16 вариант) Задаётся радиус окружности. Необходимо с помощью метода Монте-Карло рассчитать её площадь

**Алгоритм решения задачи.**

Вначале я создаю функцию для второго потока. Она реализовывает метод Монте-Карло для поиска площади круга. Это реализуется при помощи функции rand, которая определяет координаты случайных точек. Далее, в main`е я считываю аргументы консоли и создаю потоки, которые считают площадь круга.

**Листинг программы**

#include <stdio.h>

#include <windows.h>

#include <time.h>

#include <pthread.h>

#include <math.h>

struct thrdata {

    int rand;

    float r;

};

int all = 0;

double hit = 0;

int count = 0;

pthread\_mutex\_t mutex;

void\* threadFunction (void\* thread\_data) {//функция для потока

    float r = ((struct thrdata\*) thread\_data)->r;

    srand (((struct thrdata\*) thread\_data)->rand);

    while (count < all) {

        double a = rand ();

        double b = rand ();

        int c = rand ();

        if ((c % 2) == 1) {

            a \*= -1;

        }

        c = rand ();

        if ((c % 2) == 1) {

            b \*= -1;

        }

        a \*= r / RAND\_MAX;

        b \*= r / RAND\_MAX;

        pthread\_mutex\_lock(&mutex);

        if ((a \* a) + (b \* b) <= (r \* r)) {

            hit ++;

        }

        count ++;

        pthread\_mutex\_unlock(&mutex);

    }

    pthread\_exit(0);

    return NULL;

}

int main (int argc, char\*\* argv) {

    DWORD err;

    HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

    struct thrdata data;

    data.rand = time(NULL);

    int number = 0;

    float r = 0;

    double S = 0;

    while(\*argv[1]) {

        number = number \* 10 + (\*argv[1] ++ - '0');

    }

    printf ("Enter radius: ");

    scanf ("%f", &r);

    printf ("Enter number of dots: ");

    scanf ("%d", &all);

    data.r = r;

    pthread\_t thread [number];

    int timecounter = GetTickCount();

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

        void\* thread\_data = &data;

        if (pthread\_create (&thread [i], NULL, threadFunction, thread\_data)) {

            err = GetLastError ();

            SetConsoleTextAttribute(hConsole, (WORD) (0 | 4));

            printf ("ERROR\nCannot create a thread. Code of error: %d.", err);

            SetConsoleTextAttribute(hConsole, (WORD) (0 | 7));

        }

        data.rand ++;

    }

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

        pthread\_join (thread [i], NULL);

    }

    timecounter = GetTickCount() - timecounter;

    S = (2 \* r \* 2 \* r) \* (hit / all);

    printf ("S = %f\n", S);

    printf ("Inacuracy = %f\n", S - (r \* r \* 3.1415926535));

    printf ("Procent of inacuracy = %f%c \n", (100 \* abs(S - (r \* r \* 3.1415926535)) / (r \* r \* 3.1415926535)), 37);

    printf ("Worktime = %d ms\n", timecounter);

    pthread\_mutex\_destroy(&mutex);

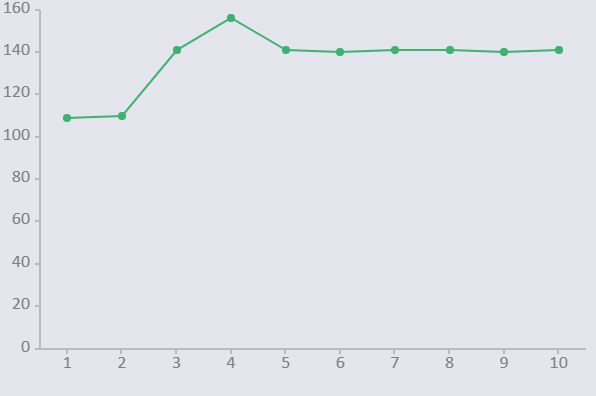
    return 0;

}

**Графики эффективности**

Количество потоков (шт.)/Время (ms)

При 1000000 точек:



Время при 1 потоке – 110 мс.

При 2 потоках – 109 мс.

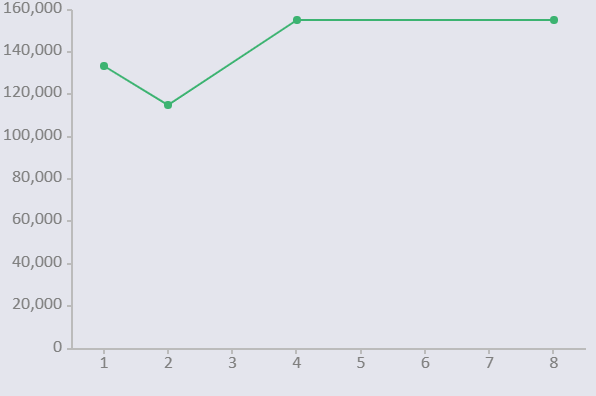
При 3 потоках – 140 мс.

При 4 потоках – 156 мс.

Следовательно добавление дополнительного потока сначала улучшает производительность всего на 1%, а потом ухудшает на 27%.

Добавление 4 потока увеличивает время выполнения на 41% по сравнению с 1 потоком.

При 1000000000 точек:



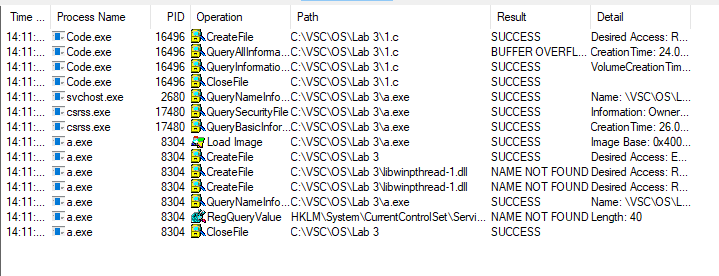
Время при 1 потоке – 107297 мс.

При 2 потоках – 102391 мс.

При 4 потоках – 147703 мс.

Следовательно добавление дополнительного потока сначала улучшает производительность на 5%, а после добавления еще двух ухудшает на 37%.

**ProcMon**



**Выводы**

Я изучил работу с потоками в Windows. Также, я узнал поточность своего процессора и ее влияние на быстродействие компьютера.

**Список литературы**

1. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. —СПб.: Издательский дом «Питер», 2018. — С. 111 - 123