Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

Студент: Кривошапкин Егор Борисович

Группа: М8О-209Б-23

Вариант: 8

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**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/EgorX2000/os\_labs/tree/main/2

**Постановка задачи**

**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). Ограничение потоков должно быть задано ключом запуска программы.  
Необходимо уметь продемонстрировать количество потоков, используемых программой, с помощью стандартных средств операционной системы.

Привести исследование зависимости ускорения и эффективности алгоритма от входящих данных и количества потоков. Объяснить получившиеся результаты.

Вариант 8: Есть K массивов одинаковой длины. Необходимо сложить эти массивы. Необходимо предусмотреть стратегию, адаптирующуюся под количество массивов и их длину (по количеству операций)

**Общие сведения о программе**

Программа написана на языке Си в операционной системе Windows. Для запуска программы в качестве аргумента командной строки необходимо указать максимальное количество используемых потоков.

**Общий метод и алгоритм решения**

Пусть на вход от пользователя поступило n потоков. Тогда каждый поток будет обрабатывать k/n массивов, оставшиеся k % n также равномерно распределяются по потокам. Каждый поток поэлементно прибавляет доступные ему массивы к массиву-результату. Если заданное пользователем максимальное количество потоков превышает количество исходных массивов, то

**Исходный код**

main.cpp

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

#include <windows.h>

#define ARRAY\_SIZE 20000

#define MAX\_ARRAYS 100000

typedef struct {

    int\* result;

    int\*\* arrays;

    int start\_index;

    int end\_index;

    int array\_length;

} ThreadData;

DWORD WINAPI SumArrays(LPVOID param) {

    ThreadData\* data = (ThreadData\*)param;

    for (int i = data->start\_index; i < data->end\_index; i++) {

        for (int j = 0; j < data->array\_length; j++) {

            data->result[j] += data->arrays[i][j];

        }

    }

    return 0;

}

double getTime() {

    LARGE\_INTEGER freq, val;

    QueryPerformanceFrequency(&freq);

    QueryPerformanceCounter(&val);

    return (double)val.QuadPart / (double)freq.QuadPart;

}

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

    double start\_time = getTime();

    if (argc < 2) {

        printf("Usage: %s <max\_threads>\n", argv[0]);

        return 1;

    }

    int max\_threads = atoi(argv[1]);

    if (max\_threads <= 0) {

        printf("Invalid max\_threads value.\n");

        return 1;

    }

    int\*\* arrays = (int\*\*)malloc(MAX\_ARRAYS \* sizeof(int\*));

    // FILE\* data = fopen("data.txt", "w");

    // srand(time(NULL));

    for (int i = 0; i < MAX\_ARRAYS; i++) {

        arrays[i] = (int\*)malloc(ARRAY\_SIZE \* sizeof(int));

        for (int j = 0; j < ARRAY\_SIZE; j++) {

            arrays[i][j] = rand() % 100;

            // fprintf(data, "%d ", arrays[i][j]);

        }

        // fputc('\n', data);

    }

    int\* result = (int\*)calloc(ARRAY\_SIZE, sizeof(int));

    int threads\_to\_use = (MAX\_ARRAYS < max\_threads) ? MAX\_ARRAYS : max\_threads;

    int arrays\_per\_thread = MAX\_ARRAYS / threads\_to\_use;

    int remaining\_arrays = MAX\_ARRAYS % threads\_to\_use;

    HANDLE\* threads = (HANDLE\*)malloc(threads\_to\_use \* sizeof(HANDLE));

    ThreadData\* thread\_data =

        (ThreadData\*)malloc(threads\_to\_use \* sizeof(ThreadData));

    int current\_array = 0;

    for (int i = 0; i < threads\_to\_use; i++) {

        thread\_data[i].result = result;

        thread\_data[i].arrays = arrays;

        thread\_data[i].array\_length = ARRAY\_SIZE;

        thread\_data[i].start\_index = current\_array;

        if (i < remaining\_arrays) {

            current\_array += arrays\_per\_thread + 1;

        } else {

            current\_array += arrays\_per\_thread;

        }

        thread\_data[i].end\_index = current\_array;

        threads[i] = CreateThread(NULL, 0, SumArrays, &thread\_data[i], 0, NULL);

    }

    WaitForMultipleObjects(threads\_to\_use, threads, TRUE, INFINITE);

    /\*

    for (int i = 0; i < ARRAY\_SIZE; i++) {

        printf("%d ", result[i]);

    }

    putchar('\n');

    for (int i = 0; i < MAX\_ARRAYS; i++) {

        CloseHandle(threads[i]);

        free(arrays[i]);

    }

    \*/

    free(arrays);

    free(result);

    free(threads);

    free(thread\_data);

    printf("%0.2lfms\n", (getTime() - start\_time) \* 1000);

    return 0;

}

**Демонстрация работы программы**

PS C:\Users\aSUS\Documents\C++\os\_labs\2\src> cd .\build\

PS C:\Users\aSUS\Documents\C++\os\_labs\2\src\build> ./main 1

33458.84ms

PS C:\Users\aSUS\Documents\C++\os\_labs\2\src\build> ./main 2

31183.74ms

PS C:\Users\aSUS\Documents\C++\os\_labs\2\src\build> ./main 5

30220.99ms

PS C:\Users\aSUS\Documents\C++\os\_labs\2\src\build> ./main 10

29515.58ms

**Выводы**

Язык Си позволяет пользователю взаимодействовать с потоками операционной системы. Для этого при работе на системе Windows требуется подключить библиотеку <windows.h>.

Создание потоков происходит быстрее, чем создание процессов, а все потоки используют одну и ту же область данных. Поэтому многопоточность – один из способов ускорить обработку каких-либо данных: выполнение однотипных, не зависящих друг от друга задач, можно поручить отдельным потокам, которые будут работать параллельно.

Средствами языка Си можно совершать системные запросы на создание потока, ожидания завершения потока, а также использовать различные примитивы синхронизации.