Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Управление потоками в ОС**

Студент: Головенко Анатолий Валерьевич

Группа: М8О–212Б–22

Вариант: 8

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**Постановка задачи**

Цель работы

Целью является приобретение практических навыков в:  
· Управление потоками в ОС  
· Обеспечение синхронизации между потоками

Задание

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

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

В отчете привести исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков. Получившиеся результаты необходимо объяснить.

Вариант 8: Есть К 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 компилируется из файла sort.cpp. В программе используются следующие системные вызовы:

1. **pthread\_create** - создает новый поток
2. **pthread\_join** - присоединяется к потоку
3. **pthread\_exit** - завершает работу потока с указанным кодом

Программа реализует сложение поэлементно и попарно. На каждом шаге я складываю чётное количество массивов. Если их количество нечётное, я оставляю последний нечётный массив, не трогаю. Результат сложения записываю в первый массив, а второй зануляю. Таким образом у меня получится в конце только один непутевой массив, который и будет ответом.

**Общий метод и алгоритм решения**.

Для реализации задачи необходимо:

1. Написать thread\_work
2. Написать sm
3. Написать многопоточность в thread\_work с использованием sm
4. Произвести анализ зависимости времени работы от количества потоков.

**Основные файлы программы**

**main.cpp:**

#include <iostream>

#include <pthread.h>

#include <string>

#include <chrono>

void printmass(int \*ms, int size, int ln\_of\_one\_vector)

{

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

{

if (i % ln\_of\_one\_vector == 0)

{

std::cout << std::endl;

}

std::cout << ms[i] << " ";

}

std::cout << std::endl;

}

typedef struct

{

int \*arr;

int start, step;

int size;

} arguments;

void \*sm(void \*input)

{

int \*arr = ((arguments \*)input)->arr;

int start = ((arguments \*)input)->start;

int step = ((arguments \*)input)->step;

int size = ((arguments \*)input)->size;

for (int i = start; i < start + size; ++i)

{

arr[i] += arr[i + step];

arr[i + step] = 0;

}

pthread\_exit(0);

}

void thread\_work(int \*data, int threads\_count, int count\_vectors, int ln\_of\_one\_vector)

{

pthread\_t threads[threads\_count];

int step = ln\_of\_one\_vector, created = 0, mass = count\_vectors;

while (mass > 1)

{

int num\_mass = mass / 2;

int pos = 0, i = 0;

for (int i = 0; i < num\_mass; i++)

{

arguments \*arg = (arguments \*)malloc(sizeof(arguments));

arg->arr = data;

arg->start = pos;

arg->step = step;

arg->size = ln\_of\_one\_vector;

pthread\_create(&threads[created], NULL, sm, arg);

++created;

pos += 2 \* step;

if (created == threads\_count)

{

for (int q = 0; q < created; ++q)

{

pthread\_join(threads[q], NULL);

}

created = 0;

}

}

for (int q = 0; q < created; ++q)

{

pthread\_join(threads[q], NULL);

}

if (mass % 2 == 0)

{

mass /= 2;

}

else

{

mass /= 2;

++mass;

}

step \*= 2;

}

}

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

{

int count\_vectors, ln\_of\_one\_vector, count\_threads = 4;

std::cout << "How many arrays would you like to operate with?" << std::endl;

std::cin >> count\_vectors;

std::cout << "What size of vector would you like?" << std::endl;

std::cin >> ln\_of\_one\_vector;

// std::cout << "How many threads would you like?" << std::endl;

// std::cin >> count\_threads;

int data[count\_vectors \* ln\_of\_one\_vector];

for (int i = 0; i < count\_vectors \* ln\_of\_one\_vector; ++i)

{

data[i] = rand() % 10;

}

// printmass(data, count\_vectors \* ln\_of\_one\_vector, ln\_of\_one\_vector);

auto start = std::chrono::high\_resolution\_clock::now();

thread\_work(data, count\_threads, count\_vectors, ln\_of\_one\_vector);

auto end = std::chrono::high\_resolution\_clock::now();

// printmass(data, ln\_of\_one\_vector, ln\_of\_one\_vector);

auto duration = std::chrono::duration\_cast<std::chrono::microseconds>(end - start);

std::cout << "Программа сработала за " << duration.count() << " microseconds" << std::endl;

}

**Пример работы**

anatolii@MacBook-Pro-Anatolii ~/D/O/h/s/build (main)> ./main

How many arrays would you like to operate with?

10

What size of vector would you like?

10

How many threads would you like?

1

7 9 3 8 0 2 4 8 3 9

0 5 2 2 7 3 7 9 0 2

3 9 9 7 0 3 9 8 6 5

7 6 2 7 0 3 9 9 9 1

7 2 3 6 5 5 8 1 4 7

1 3 8 4 8 0 4 6 0 3

2 6 9 4 1 3 7 8 8 3

8 1 5 3 5 4 3 6 5 9

5 4 9 1 7 5 5 4 1 8

8 3 5 2 2 6 6 7 8 4

48 48 55 44 35 34 62 66 44 51

Программа сработала за 726 microseconds

anatolii@MacBook-Pro-Anatolii ~/D/O/h/s/build (main)> ./main

How many arrays would you like to operate with?

10

What size of vector would you like?

10

How many threads would you like?

2

7 9 3 8 0 2 4 8 3 9

0 5 2 2 7 3 7 9 0 2

3 9 9 7 0 3 9 8 6 5

7 6 2 7 0 3 9 9 9 1

7 2 3 6 5 5 8 1 4 7

1 3 8 4 8 0 4 6 0 3

2 6 9 4 1 3 7 8 8 3

8 1 5 3 5 4 3 6 5 9

5 4 9 1 7 5 5 4 1 8

8 3 5 2 2 6 6 7 8 4

48 48 55 44 35 34 62 66 44 51

Программа сработала за 942 microseconds

anatolii@MacBook-Pro-Anatolii ~/D/O/h/s/build (main)> ./main

How many arrays would you like to operate with?

10

What size of vector would you like?

10

How many threads would you like?

3

7 9 3 8 0 2 4 8 3 9

0 5 2 2 7 3 7 9 0 2

3 9 9 7 0 3 9 8 6 5

7 6 2 7 0 3 9 9 9 1

7 2 3 6 5 5 8 1 4 7

1 3 8 4 8 0 4 6 0 3

2 6 9 4 1 3 7 8 8 3

8 1 5 3 5 4 3 6 5 9

5 4 9 1 7 5 5 4 1 8

8 3 5 2 2 6 6 7 8 4

48 48 55 44 35 34 62 66 44 51

Программа сработала за 521 microseconds

**Таблица зависимости времени работы от количества потоков:**

|  |  |
| --- | --- |
| **Количество потоков** | **Время работы, микросекунды** |
| **1** | **24 624** |
| **2** | **25 097** |
| **3** | **22 432** |
| **4** | **19 865** |
| **5** | **18 767** |
| **6** | **18 546** |
| **7** | **18 354** |

**Вывод**

В процессе работы я изучил работу с многопоточностью в ОС. Одной из моих первоначальных ошибок было то, что я забыл добавить pthread\_join из-за чего потоки не успевали завершиться, и в результате массив портился. В процессе замеров времени выполнения я выяснил, что основной прирост в производительности происходит при использовании 4 потоков. Дальнейшее увеличение числа потоков не дает серьезного прироста в производительности.