Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

**ПОТОКИ**

Студент: Армишев Кирилл Константинович

Группа: М8О–208Б–21

Вариант: 1

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2022.

**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 потоков 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 входящих данных и количества потоков. Получившиеся результаты необходимо объяснить.

**Вариант №1**

**Отсортировать массив целых чисел при помощи битонической сортировки**

**Общие сведения о программе**

Программа состоит из одного файла — main.c. Максимальное количество потоков указывается как аргумент программы. На вход программа получает массивы, равные кол-ву потоков.

1. **pthread\_create()** - создать новый поток
2. **pthread\_join()** - дождаться завершения потоками

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Изучить принципы работы pthread\_create и т.д.
2. Изучить алгоритм битонической сортировки
3. Реализовать многопоточный алгоритм
4. Провести тесты

**Основные файлы программы**

**main.c:**

#include "time.h"  
#include "stdio.h"  
#include "stdlib.h"  
#include "pthread.h"  
#define MAX 4  
  
int up = 1;  
int down = 0;  
  
void Exchange(int \* num1, int \* num2)  
{  
 int temp;  
  
 temp = \*num1;  
 \*num1 = \*num2;  
 \*num2 = temp;  
}  
  
void compare(int\* arr, int i, int j, int dir)  
{  
 if (dir == (arr[i] > arr[j])) {  
 Exchange(&arr[i], &arr[j]);  
 }  
}  
  
void bitonicmerge(int\* arr, int low, int c, int dir)  
{  
 int k = 0;  
 int i = 0;  
  
 if (c > 1) {  
 k = c / 2;  
 i = low;  
 while (i < low + k) {  
 compare(arr, i, i + k, dir);  
 i++;  
 }  
  
 bitonicmerge(arr, low, k, dir);  
 bitonicmerge(arr,low + k, k, dir);  
 }  
}  
  
void recbitonic(int\* arr, int low, int v, int dir)  
{  
 int k = 0;  
  
 if (v > 1) {  
 k = v / 2;  
 recbitonic(arr,low, k, up);  
 recbitonic(arr,low + k, k, down);  
 bitonicmerge(arr, low, v, dir);  
 }  
}  
  
  
void \*routine(void \*arg) {  
 int\* mass = (int \*) arg;  
 recbitonic(mass, 0, MAX, up);  
 for (int i = 0; i < MAX; i++)  
 printf("%d ", mass[i]);  
 printf("\n");  
 free(arg);  
 return NULL;  
}  
  
int main(int argc, char\* argv[]){  
 int n = atoi(argv[1]);  
 pthread\_t pid[n];  
  
 int data[n][MAX];  
 printf("Enter %d massives!\n", n);  
 for(int i = 0; i < n; i++){  
 for (int u = 0; u < MAX; u++)  
 scanf("%d", data[i]+u);  
 }  
  
 double start = clock();;  
  
 if (argc == 2) {  
 for (int i = 0; i < n; i++) {  
 int \*data2=malloc(sizeof(int) \* MAX);  
 for(int u=0; u<MAX; u++){  
 data2[u] = data[i][u];  
 }  
 if (pthread\_create(&pid[i], NULL, &routine, data2) != 0) {  
 perror("Couldn't create a thread\n");  
 return 1;  
 }  
 printf("Thread %d has started\n", i+1);  
 }  
 for(int i = 0; i < n; i++){  
 if (pthread\_join(pid[i], NULL) != 0){  
 return 2;  
 }  
 printf("Thread has finished execution!\n");  
 }  
 }  
 else{  
 printf("Please enter an appropriate program key !\n");  
 }  
 printf("Count of threads: %d\n", n);  
 printf("The program ran for %.4lf seconds\n", (clock() - start) / (CLOCKS\_PER\_SEC));  
 return 0;  
}

**Пример работы**

kirillarmishev@2 Lab3 % gcc -pthread -o lab3 main.c

-------------------------test 1-------------------------

kirillarmishev@2 Lab3 % ./lab3 4

Enter 4 massives!

4 2 5 3

89 323 3434 1

3232 4 5534 2

323 554 23 3

Thread 1 has started

Thread 2 has started

Thread 3 has started

Thread 4 has started

2 4 3232 5534

3 23 323 554

2 3 4 5

1 89 323 3434

Thread has finished execution!

Thread has finished execution!

Thread has finished execution!

Thread has finished execution!

Count of threads: 4

The program ran for 0.0006 seconds

**Вывод**

В результате данной лабораторной работы, я научился работать c потоками, реализовал многопоточный алгоритм битонической сортировки. Сравнивая результаты времени сортировки двух алгоритмов мы получили, что многопоточный алгоритм работает медленнее, что скорее всего связано с большим количеством системных вызовов, связанных с работой потоков. Полученные мною знания пригодятся при дальнейшей работе.