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: Бойцов Иван Алексеевич
Группа: М8О–212Б–22
Вариант: 5
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Лабораторная работа №2

Цель работы

Целью является приобретение практических навыков в:

1. Управление потоками в ОС
2. 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 5

Отсортировать массив целых чисел при помощи четно-нечетной сортировки Бетчера.

Решение

Написать способ решения

Первый вариант решения

```
#include <iostream>
#include <windows.h>
#include <vector>
#include <algorithm>

#define MAX_THREADS 16 // Максимальное количество потоков, ограниченное программно

using namespace std;
```

```
vector<int> arr = {
        45, 23, 78, 90, 56, 12, 89, 34, 67, 91, 26, 48, 102, 67, 53, 88, 39, 73, 57, 21,
        64, 85, 77, 31, 94, 15, 82, 49, 37, 68, 96, 28, 13, 74, 99, 83, 11, 59, 105, 70,
        19, 32, 97, 66, 80, 20, 100, 17, 51, 86, 38, 79, 43, 24, 95, 76, 58, 18, 30, 84,
        61, 52, 63, 47, 33, 81, 27, 98, 46, 41, 36, 55, 16, 101, 92, 71, 40, 60, 54, 22,
        87, 75, 29, 93, 72, 65, 14, 50, 103, 35, 62, 44, 104, 25, 42, 109, 106, 110, 111,
108
};
int max_threads;
HANDLE ghMutex;

DWORD WINAPI OddEvenSort(LPVOID lpParam);

void RunOddEvenSort() {
    int n = arr.size();
    bool isSorted = false;

    while (!isSorted) {
        isSorted = true;
        vector<HANDLE> threads;

        // Четная фаза
        for (int i = 0; i < max_threads && i < (n + 1) / 2; i++) {
            int* phase = new int(1); // Четная фаза начинается с индекса 1
            HANDLE hThread = CreateThread(NULL, 0, OddEvenSort, phase, 0,
NULL);
            if (hThread != NULL) {
                threads.push_back(hThread);
            }
        }

        WaitForMultipleObjects(threads.size(), threads.data(), TRUE, INFINITE);
        for (auto hThread : threads) {
            CloseHandle(hThread);
        }
        threads.clear();

        // Нечетная фаза
        for (int i = 0; i < max_threads && i < n / 2; i++) {
            int* phase = new int(0); // Нечетная фаза начинается с индекса 0
            HANDLE hThread = CreateThread(NULL, 0, OddEvenSort, phase, 0,
NULL);
            if (hThread != NULL) {
                threads.push_back(hThread);
```

```
            }
        }

        WaitForMultipleObjects(threads.size(), threads.data(), TRUE, INFINITE);
        for (auto hThread : threads) {
            CloseHandle(hThread);
        }

        // Проверяем, отсортирован ли массив
        for (int i = 0; i < n - 1; i++) {
            if (arr[i] > arr[i + 1]) {
                isSorted = false;
                break;
            }
        }
    }
}

// Простая четно-нечетная сортировка Бутчера
void oddEvenSortSimple(vector<int>& arr) {
    bool isSorted = false;
    int n = arr.size();

    while (!isSorted) {
        isSorted = true;

        // Четная фаза
        for (int i = 0; i <= n - 2; i += 2) {
            if (arr[i] > arr[i + 1]) {
                swap(arr[i], arr[i + 1]);
                isSorted = false;
            }
        }

        // Нечетная фаза
        for (int i = 1; i <= n - 2; i += 2) {
            if (arr[i] > arr[i + 1]) {
                swap(arr[i], arr[i + 1]);
                isSorted = false;
            }
        }
    }
}
```

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

    // Ошибка инициализации решения
    if (argc != 2) {
        cerr << "Usage: " << argv[0] << " <max_threads>" << endl;
        system("pause");
        return 1;
    }

    max_threads = min(MAX_THREADS, atoi(argv[1]));

    // Ошибка создания Mutex
    ghMutex = CreateMutex(NULL, FALSE, NULL);
    if (ghMutex == NULL) {
        cerr << "CreateMutex error: " << GetLastError() << endl;
        system("pause");
        return 1;
    }

    cout << "Start programm..." << endl;


    RunOddEvenSort();

    CloseHandle(ghMutex);

    cout << "Sorted array: ";
    for (const auto& num : arr) {
        cout << num << " ";
    }
    cout << endl;

    system("pause");
    return 0;
}

DWORD WINAPI OddEvenSort(LPVOID lpParam) {
    int phase = *(int*)lpParam;
    delete (int*)lpParam;

    int n = arr.size();

    for (int i = phase; i < n - 1; i += 2) {
```

```
        WaitForSingleObject(ghMutex, INFINITE);
        if (arr[i] > arr[i + 1]) {
            swap(arr[i], arr[i + 1]);
        }
        ReleaseMutex(ghMutex);
    }

    return 0;
}
```

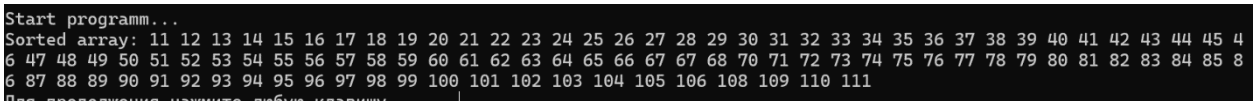
Второй вариант решения

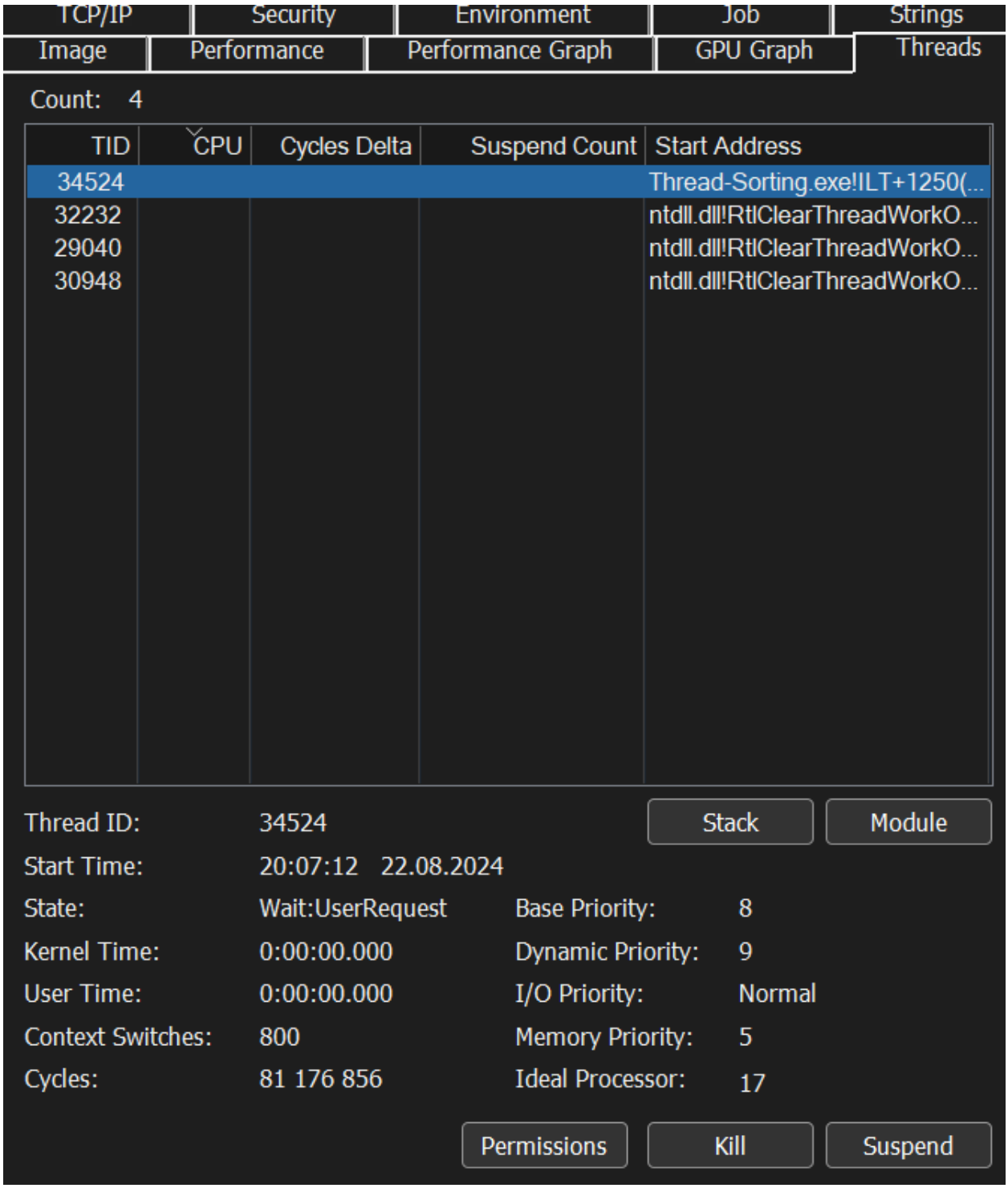
```
#include <iostream>
#include <vector>
#include <thread>
#include <mutex>
#include <condition_variable>
#include <windows.h>
#include <string>

using namespace std;

mutex mtx;
condition_variable cv;
int activeThreads = 0;
int maxThreads;

//Четно-нечетная сортировка Бетчера
void oddEvenSort(vector<int>& arr, int start, int step) {
        for (int i = start; i + 1 < arr.size(); i += step) {
                if (arr[i] > arr[i + 1]) {
                        swap(arr[i], arr[i + 1]);
                }
        }
}

void threadWorker(vector<int>& arr, int step, int index) {
        {
                unique_lock<mutex> lock(mtx);
                cv.wait(lock, [] { return activeThreads < maxThreads; });
                activeThreads++;
        }

        oddEvenSort(arr, index, 2);
```

```
		{
			lock_guard<mutex> lock(mtx);
			activeThreads--;
			cv.notify_one();
		}
	}
	//Параллельная четно-нечетная сортировка Бетчера
	void oddEvenSortParallel(vector<int>& arr, int maxThreads) {
		bool sorted = false;
		while (!sorted) {
			sorted = true;

			vector<thread> threads;

			//Четная фаза
			for (int i = 0; i < 2; i++) {
				threads.emplace_back(threadWorker, ref(arr), 2, i);
			}

			for (auto& t : threads) {
				t.join();
			}

			//Проверка на отсортированность массива
			for (int i = 0; i + 1 < arr.size(); i++) {
				if (arr[i] > arr[i + 1]) {
					sorted = false;
					break;
				}
			}
		}
	}
	int main(int args, char* argv[]) {

		if (args < 2) {
			cerr << "Usage: " << argv[0] << " <maxThreads>" << endl;
			system("pause");
			return 1;
		}

		maxThreads = stoi(argv[1]);

		vector<int> arr = { 24, 11, 13, 4, 6, 22, 15, 9, 1, 10 };
```

```
	cout << "Original array: ";

	for (int num : arr) {
		cout << num << " ";
	}

	cout << endl;

	oddEvenSortParallel(arr, maxThreads);

	cout << "Sorted array: ";
	for (int num : arr) {
		cout << num << " ";
	}
	cout << endl;

	system("pause");
	return 0;
}
```

Доказательство работы





Вывод

В результате выполнения данной лабораторной работы была написана программа на языке C++, осуществляющая сортировку массива целых чисел при помощи четно-нечетной сортировки Бутчера, которая обрабатывает данные в многопоточном режиме. Были получены практические навыки в управлении потоками в Windows и обеспечении синхроницизации между ними.