Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Операционные системы»

Управление потоками в операционной системе «UNIX».

Студент: Селивёрстов Д. С.
Преподаватель: Миронов Е. С.
Группа: М8О-201Б-21
Вариант 15
Дата:
Оценка:
Подпись:

Москва, 2023

Условие

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.

Задание

Отсортировать массив целых чисел при помощи TimSort.

Метод решения

Составленный алгоритм соответсвует принципу *completelylockless*, заключающийся в том, что каждый поток изменяет только те данные, которые не изменяют другие потоки.

Код программы

main.cpp

```
#include <bits/stdc++.h>

#include "utils.hpp"
#include "body.hpp"

int main()
{
    std::vector<int> arr;

    for (int i = 0; i < 16; ++i){
        arr.push_back(GetRandomInt(0, 32));
    }

    for (int elem : arr){
        std::cout << elem << " ";
    }
    std::cout << '\n';
```

```
    TimSort(arr, 2);

    for (int elem : arr){
        std::cout << elem << " ";
    }
    std::cout << '\n';

    return 0;
}
```

utils.hpp

```
#ifndef UTILS_HPP
#define UTILS_HPP

#include <random>

int GetRandomInt(int min, int max);

#endif
```

body.hpp

```
#ifndef BODY_HPP
#define BODY_HPP

#include <vector>
#include <fstream>
#include <pthread.h>

#include <iostream>

void TimSort(std::vector<int> &arr, int numOfThreads);

#endif
```

utils.cpp

```
#include "utils.hpp"

int GetRandomInt(const int min, const int max)
{
    std::random_device rd;
    std::mt19937 mt(rd());
    std::uniform_int_distribution<int> dist(min, max);
```

```
    return dist(mt);
}
```

Выводы

В процессе выполнения лабораторной работы были приобретены навыки практического применения создания, обработки и отслеживания состояния потоков. Для выполнения данного варианта задания использование примитивов синхронизации не требуется.