МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ

ФЕДЕРАЦИИ

Московский Авиационный Институт

(Национальный Исследовательский Университет)

Институт №8 "Компьютерные науки и прикладная математика"

Кафедра 806 "Вычислительная математика и программирование"

Лабораторная работа №2

По курсу «Операционные системы»

Студент: Лернер Ф. Л.

Группа: М8О-208Б-23

Преподаватель: Живалев Е. А.

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Москва, 2024

**Тема:** Управление потоками и синхронизация в ОС

**Цель работы:** Целью работы является приобретение практических навыков в:

* Управлении потоками в операционной системе.
* Организации синхронизации между потоками для эффективного использования многопоточности.

**Вариант:** 11. Наложить K раз медианный фильтр на матрицу, состоящую из целых чисел. Размер окна задается пользователем

**Задачи:**

1. Разработать программу на языке Си, реализующую многопоточную сортировку массива целых чисел методом слияния.
2. Ограничить максимальное количество одновременно работающих потоков с использованием заданного параметра.
3. Обеспечить корректную синхронизацию потоков с помощью стандартных средств операционной системы.
4. Провести исследование зависимости ускорения и эффективности алгоритма от входных данных и количества потоков.

**Описание решения:** Программное решение представляет собой многопоточную реализацию сортировки массива методом слияния. Основные компоненты программы:

* **Управление потоками:** Для создания потоков используется библиотека pthread. Максимальное количество одновременно работающих потоков задается пользователем в виде параметра запуска программы.
* **Синхронизация потоков:** Для управления количеством активных потоков используется мьютекс и глобальная переменная countOfActiveThreads. Увеличение и уменьшение счетчика активных потоков синхронизировано с помощью мьютекса.
* **Алгоритм сортировки:** Сортировка массива выполняется рекурсивно. При каждом делении массива создается новый поток, если количество активных потоков меньше заданного максимума. В противном случае обработка выполняется в текущем потоке.

Программа функционирует следующим образом:

1. Пользователь задает максимальное количество потоков и тип ввода данных (вручную или случайная генерация).
2. Если выбран ручной ввод, пользователь вводит размер массива и его элементы. В случае случайной генерации массив заполняется случайными числами.
3. Основной поток вызывает функцию сортировки, передавая в нее данные о массиве.
4. В процессе сортировки массив делится на части, которые обрабатываются либо в новых потоках, либо в текущем потоке, в зависимости от текущей загрузки.
5. После завершения сортировки выводится отсортированный массив.

**Репозиторий:** https://github.com/LernerF/labs\_os/tree/main

**Исходный код:** Программное обеспечение состоит из следующих файлов:

1. **main.c:** Инициализация программы, ввод данных и запуск сортировки.
2. **matrix\_utils.cpp –** Реализация управления потоками
3. **median\_filter.cpp -** Реализация функций фильтра.

Пример кода:

#include <iostream>

#include <pthread.h>

#include "matrix\_utils.h"

#include "median\_filter.h" // MAX\_THREADS, thread\_function

int main() {

rows = 6;

cols = 6;

window\_size = 3;

K = 1;

initialize\_matrix(rows, cols);

std::cout << "Original Matrix:\n";

print\_matrix(matrix, rows, cols);

pthread\_t threads[MAX\_THREADS];

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

int \*thread\_id = new int(i);

pthread\_create(&threads[i], nullptr, thread\_function, thread\_id);

}

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

pthread\_join(threads[i], nullptr);

}

std::cout << "Filtered Matrix:\n";

print\_matrix(matrix, rows, cols);

free\_matrix(rows);

pthread\_mutex\_destroy(&mutex); // Корректное завершение работы с мьютексом

return 0;

} **Вывод:** В ходе выполнения лабораторной работы была реализована многопоточная сортировка массива методом слияния. Программа корректно ограничивает количество одновременно работающих потоков, обеспечивая при этом эффективное использование многопоточности.

Результаты исследования зависимости ускорения и эффективности алгоритма от количества потоков и объема данных показали, что увеличение числа потоков до определенного предела ускоряет выполнение программы. Однако при чрезмерном увеличении числа потоков эффективность снижается из-за накладных расходов на управление потоками. Полученные результаты соответствуют теоретическим ожиданиям.