Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение
высшего образования
«Московский авиационный институт
(Национальный исследовательский университет)»

Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

ОТЧЁТ

по лабораторной работе № 2
по курсу «Операционные системы»

Студент: Князьков Николай Дмитриевич
Группа: М80–203БВ–24
Вариант: 13
Преподаватель: Соколов А. А.
Оценка:_____
Дата:_____
Подпись:_____

Москва, 2025

Лабораторная работа № 2

1.1 Постановка задачи

Цель работы — изучение принципов многопоточной обработки данных и механизмов синхронизации потоков.

Задание заключается в разработке многопоточного приложения на языке Си, реализующего операцию свёртки матрицы вещественных чисел. Необходимо исследовать зависимость ускорения и эффективности параллельного алгоритма от числа потоков и размеров обрабатываемых данных.

1.2 Метод решения

Матрица разбивается по строкам между потоками. Каждый поток последовательно обрабатывает назначенные строки. Для синхронизации используется комбинация семафоров, мьютекса и условной переменной. После завершения каждой итерации выполняется обмен входной и выходной матриц.

1.3 Оценка производительности

Ускорение параллельного алгоритма вычисляется по формуле:

$$S(p) = \frac{T_1}{T_p},$$

где T_1 — время выполнения программы при одном потоке, T_p — время выполнения при использовании p потоков.

Эффективность определяется выражением:

$$E(p) = \frac{S(p)}{p}.$$

Эксперименты показали, что при увеличении числа потоков время выполнения уменьшается. Ускорение возрастает практически линейно до достижения аппаратных ограничений. Эффективность остаётся высокой для средних размеров данных. Полученные результаты являются корректными и устойчивыми.

1.4 Основные фрагменты программы

1.4.1 Глобальные данные

```
double **matrix;
double **temp;
int N;
int M;
int K;
int W;
int max_threads;
sem_t semaphore;
pthread_mutex_t mutex;
pthread_cond_t cond;
int current_row;
int threads_completed;
```

Листинг 1.1: Глобальные переменные

1.4.2 Функция свёртки

```
void apply_filter(int row) {
    int half = W / 2;
    for (int j = 0; j < M; j++) {
        double sum = 0.0;
        int count = 0;
        for (int wi = -half; wi <= half; wi++) {
            for (int wj = -half; wj <= half; wj++) {
                int ni = row + wi;
                int nj = j + wj;
                if (ni >= 0 && ni < N && nj >= 0 && nj < M) {
                    sum += matrix[ni][nj];
                    count++;
                }
            }
        }
        temp[row][j] = sum / count;
    }
}
```

Листинг 1.2: Функция свёртки строки матрицы

1.4.3 Функция потока

```
void* worker() {
    for (int k = 0; k < K; k++) {
        sem_wait(&semaphore);
        while (1) {
            pthread_mutex_lock(&mutex);
            int row = current_row++;
            pthread_mutex_unlock(&mutex);
            if (row >= N) break;
            apply_filter(row);
        }
        pthread_mutex_lock(&mutex);
        threads_completed++;
        if (threads_completed == max_threads) {
            double **swap = matrix;
            matrix = temp;
            temp = swap;
            current_row = 0;
            threads_completed = 0;
            pthread_cond_broadcast(&cond);
        } else {
            pthread_cond_wait(&cond, &mutex);
        }
        pthread_mutex_unlock(&mutex);
        sem_post(&semaphore);
    }
    return NULL;
}
```

Листинг 1.3: Функция потока

1.4.4 Создание потоков

```
pthread_t threads[max_threads];
for (int i = 0; i < max_threads; i++) {
    pthread_create(&threads[i], NULL, worker, NULL);
}
for (int i = 0; i < max_threads; i++) {
    pthread_join(threads[i], NULL);
}
```

Листинг 1.4: Создание и ожидание потоков

1.5 Выводы

В ходе выполнения лабораторной работы был разработан и протестирован многопоточный алгоритм свёртки матрицы. Программа работает корректно и стабильно при различных размерах данных и количестве потоков. Экспериментальные результаты показали ожидаемое ускорение и высокую эффективность параллельных вычислений. Полученные результаты находятся в пределах нормы и подтверждают корректность реализации.