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: Муханов Андрей Ильич

Группа: М8О-209Б-23

Вариант: 12

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Демонстрация работы программы
4. Выводы

**Репозиторий**

https://github.com/andreymuch/OC\_MAI

**Постановка задачи**

**Цель работы**

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 задания: Наложить K раз фильтры эрозии и наращивания на матрицу, состоящую из вещественных чисел. На выходе получается 2 результирующие матрицы

**Демонстрация работы программы**

Для оценки эффективности работы программы проведем небольшое исследование, целью которого является выявление зависимости времени обработки данных от количества задействованных потоков. Для эксперимента была создана матрица размером 100×100, заполненная случайными вещественными числами. Программа выполняла стандартные операции обработки данных, а время выполнения фиксировалось при различном количестве потоков.

| **Число потоков** | **Время обработки** | **K** |
| --- | --- | --- |
| **2** | **8435 microseconds** | **35** |
| **4** | **5246 microseconds** | **35** |
| **8** | **4114 microseconds** | **35** |
| **12** | **3382 microseconds** | **35** |
| **16** | **5398 microseconds** | **35** |

Из таблицы видно, что при увеличении числа потоков наблюдается значительное сокращение времени обработки данных, особенно при переходе от двух к четырем и далее к восьми потокам. Однако после достижения максимального эффективного числа потоков (в данном случае 12) производительность начинает снижаться. Это связано с дополнительными накладными расходами на синхронизацию потоков и ограничениями аппаратных ресурсов, которые становятся заметными при увеличении их числа сверх оптимального значения.

Таким образом, можно сделать вывод, что для достижения наилучших результатов в многопоточной обработке данных важно учитывать аппаратные ограничения системы и не превышать оптимального количества потоков.

**Выводы**

1. **Практическое освоение многопоточности:**  
   Выполненная работа позволила приобрести практические навыки управления потоками в операционной системе. Использование стандартных средств для создания и синхронизации потоков продемонстрировало возможности эффективной параллельной обработки данных.
2. **Оптимизация с учетом аппаратных ограничений:**  
   В ходе исследования было выявлено, что увеличение числа потоков существенно ускоряет обработку данных только до определенного предела, который зависит от характеристик аппаратного обеспечения. После достижения оптимального числа потоков дальнейшее увеличение приводит к снижению эффективности из-за накладных расходов на управление потоками и синхронизацию.
3. **Зависимость производительности от структуры алгоритма:**  
   Для данной задачи использование многопоточности позволило добиться значительного сокращения времени обработки данных. Однако ключевым фактором остается структура самого алгоритма. Задачи, требующие минимальной синхронизации между потоками, демонстрируют более высокую производительность при масштабировании.
4. **Практическая значимость:**  
   Проведенное исследование и полученные результаты могут быть использованы при разработке программ для высокопроизводительной обработки данных, где важно учитывать аппаратные ограничения и оптимизировать алгоритмы с учетом многопоточности.
5. **Рекомендации для будущих разработок:**
   * Использовать профилирование программы для выявления оптимального числа потоков в зависимости от задачи и доступных ресурсов.
   * Проектировать алгоритмы с минимальными зависимостями между потоками, чтобы снизить накладные расходы на синхронизацию.
   * Учитывать характеристики целевой платформы (количество ядер, поддержка многопоточности), чтобы эффективно распределять нагрузку.

Полученные результаты подчеркивают важность грамотного использования многопоточности для достижения максимальной производительности при обработке больших объемов данных.