Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

Студент: Царев Михаил Вадимович

Группа: М8О-208Б-21

Вариант: 4

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2022

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

https://github.com/de1phin/OS/lab3

**Постановка задачи**

**Цель работы**

Целью является приобретение практических навыков в:

* Управление потоками в ОС
* 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 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.

Вариант 4: Отсортировать массив целых чисел при помощи TimSort

**Общие сведения о программе**

Программа состоит из одного исходного файла. Использует стандартные средства управления потоками (системные вызовы pthread\_create, pthread\_mutex\_(init|lock|unlock|trylock))

**Общий метод и алгоритм решения**

Работа осуществляется засчет четырех функций:

* Цикл, разбивающий массив на блоки
* Функция сортировки блока
* Цикл, запускающий слияние соседних отсортированных блоков
* Функция слияния соседних отсортированных блоков

Каждая из заданных функций может быть запущена как последовательно, так и параллельно, в отдельном потоке. При запуске может быть задан флаг максимального количества потоков, изначально установленный в 1 (всё выполняется последовательно). Существует вспомогательная функция void thread(void(\* *func*)(void\*), void\* *func\_args*), помогающая управлять потоками: если текущее количество потоков равно установленному пределу — функция вызывается на месте, то есть последовательно, если нет, то запускается отдельный поток.

Время работы в последовательном режиме, требуемое для сортировки 1000 чисел — 0,13 мс, в параллельном режиме (предел количества потоков = 100000) — 3,98 мс. Обосновывается это следующим:

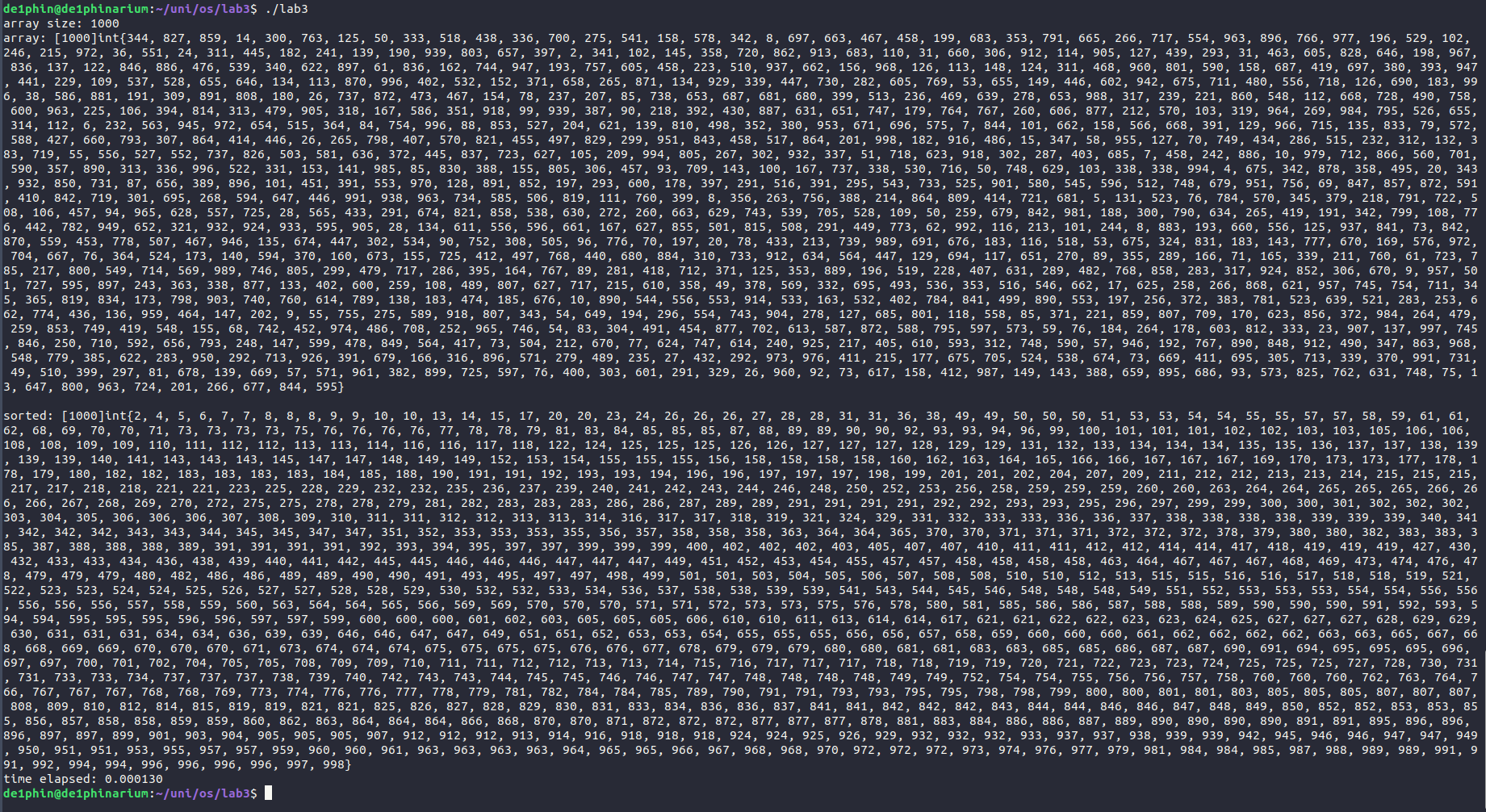
1. Сортировка представлят собой непрерывные вычислительные действия, а не ожидание блокирующих действий, следовательно переход в параллельный режим не дает никаких преимуществ, а только добавляет расходы на создание и удаление потоков, так как на самом деле программа работает не параллельно, а конкуррентно
2. Для оптимального времени работы сортировки, необходимо сливать блоки по определенному алгоритму с конца, когда всё уже отсортировано, что идет вопреки нашему распараллеливанию

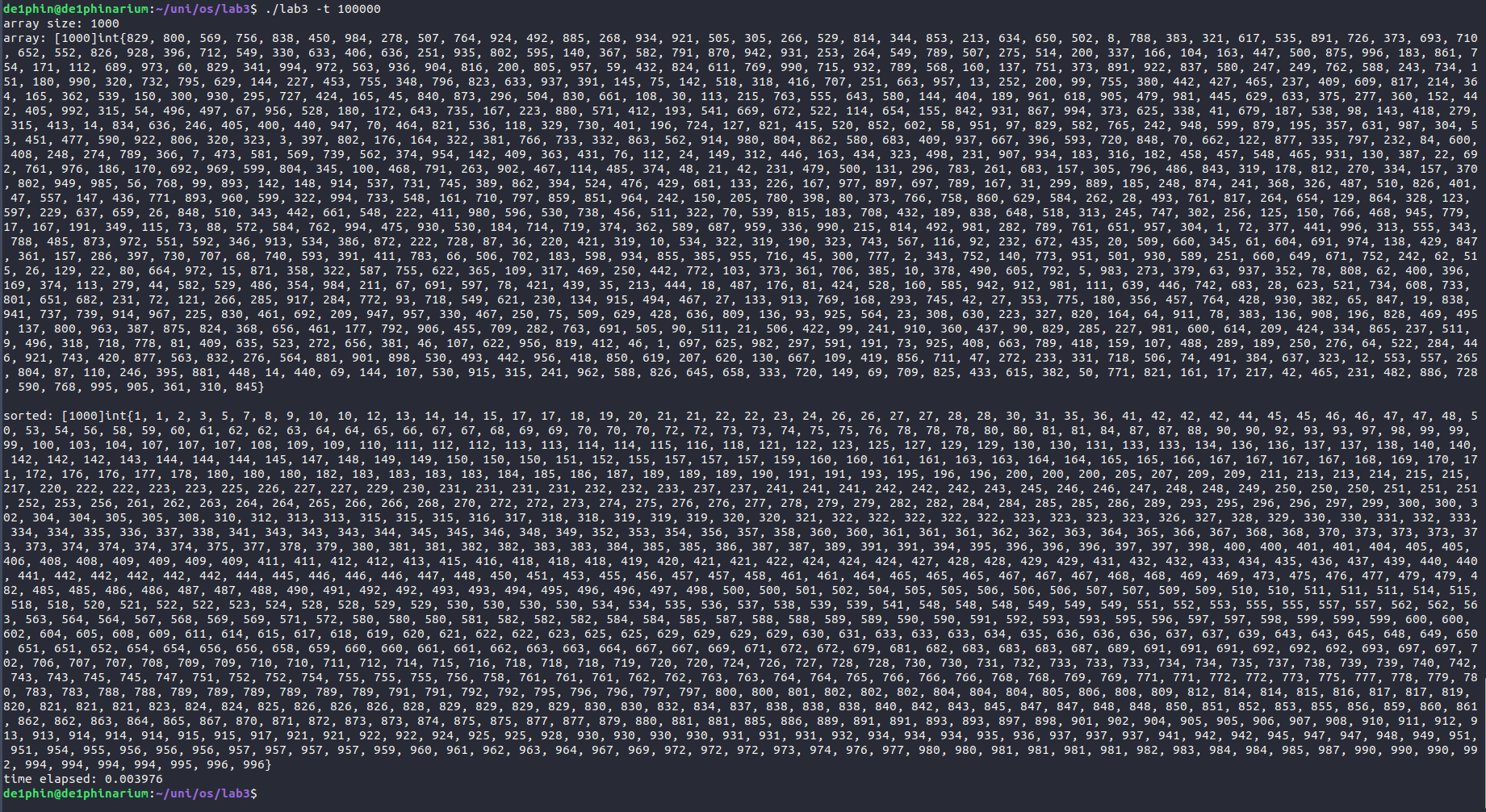
**Исходный код**

В репозитории.

**Демонстрация работы программы**

Один поток:

“Бесконечно” потоков:



**Выводы**

Реализовал алгоритм параллельной (конкуррентной) сортировки