Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №0 по курсу**

**«Операционные системы»**

**Тема работы**

**“Потоки”**

Студент: Зинин Владислав Владимирович

Группа: М8О-208Б-20

Вариант:13

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**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/frankeloff/OS

**Постановка задачи**

Задача: Есть набор 128 битных чисел, записанных в шестнадцатеричном представлении, хранящихся в файле. Необходимо посчитать их среднее арифметическое. Округлить результат до целых. Количество используемой оперативной памяти должно задаваться "ключом"

**Общие сведения о программе**

Для реализации поставленной задачи нам нужны следующие библиотеки:  
<unistd.h> - для работы с системными вызовами в Linux

<limits.h> - для определения характеристик общих типов переменных. <stdlib.h> - для того, чтобы можно было пользоваться функциями, отвечающими за работу с памятью.

<time.h> - для функций, работающих со временем (нужно для строчки

srand(time(NULL)) - для генерации рандомных чисел с использованием

текущего времени).

<pthread.h> - для работы с потоками.

<ctype.h> - для классификации и преобразования отдельных символов.

<sys/stat.h> - для доступа к файлам.

<fcntl.h> - для работы с файловым дескриптором.

<inttypes.h> - макросы для использования с функциями printf и scanf.

<string.h> - для использования функций над строками.

Для работы с потоками я использую такие системные вызовы, как pthread\_create, отвечающий за создание потока, которая в случае успешного выполнения функция возвращает 0. Если произошли ошибки, то могут быть возвращены следующие значения:

* **EAGAIN** – у системы нет ресурсов для создания нового потока, или система не может больше создавать потоков, так как количество потоков превысило значение PTHREAD\_THREADS\_MAX (например, на одной из машин, которые используются для тестирования, это магическое число равно 2019)
* **EINVAL** – неправильные атрибуты потока (переданные аргументом attr)
* **EPERM** – Вызывающий поток не имеет должных прав для того, чтобы задать нужные параметры или политики планировщика.

Сигнатура pthread\_create следующая: **int pthread\_create(\*ptherad\_t, const pthread\_attr\_t \*attr, void\* (\*start\_routine)(void\*), void \*arg);**

Где функция получает в качестве аргументов указатель на поток, переменную типа pthread\_t, в которую, в случае удачного завершения сохраняет id потока. pthread\_attr\_t – атрибуты потока. В случае если используются атрибуты по умолчанию, то можно передавать NULL. start\_routin – это непосредственно та функция, которая будет выполняться в новом потоке. arg – это аргументы, которые будут переданы функции.

Также я использую pthread\_join, отвечающий за ожидание завершения потока, имеющий тип возвращаемого значения int и принимающий 2 аргумента: указатель на поток и указатель на указатель в качестве аргумента для хранения возвращаемого значения.

Помимо системных вызовов, связанных с потоками, в моей программе имеются следующие системные вызовы:

off\_t lseek(...) - устанавливает смещение для файлового дескриптора в значение аргумента offset.

int open(...) - открытие файлового дескриптора.

void exit(...) - выход из процесса с заданным статусом.

int close(...) - закрытие файлового дескриптора.

Программа собирается и запускается при помощи следующих команд:

make

./generator.exe filename count (например, ./generator.exe nums 100)

./main.exe filename thread\_number memory\_amount (пример: ./main.exe nums 2 300).

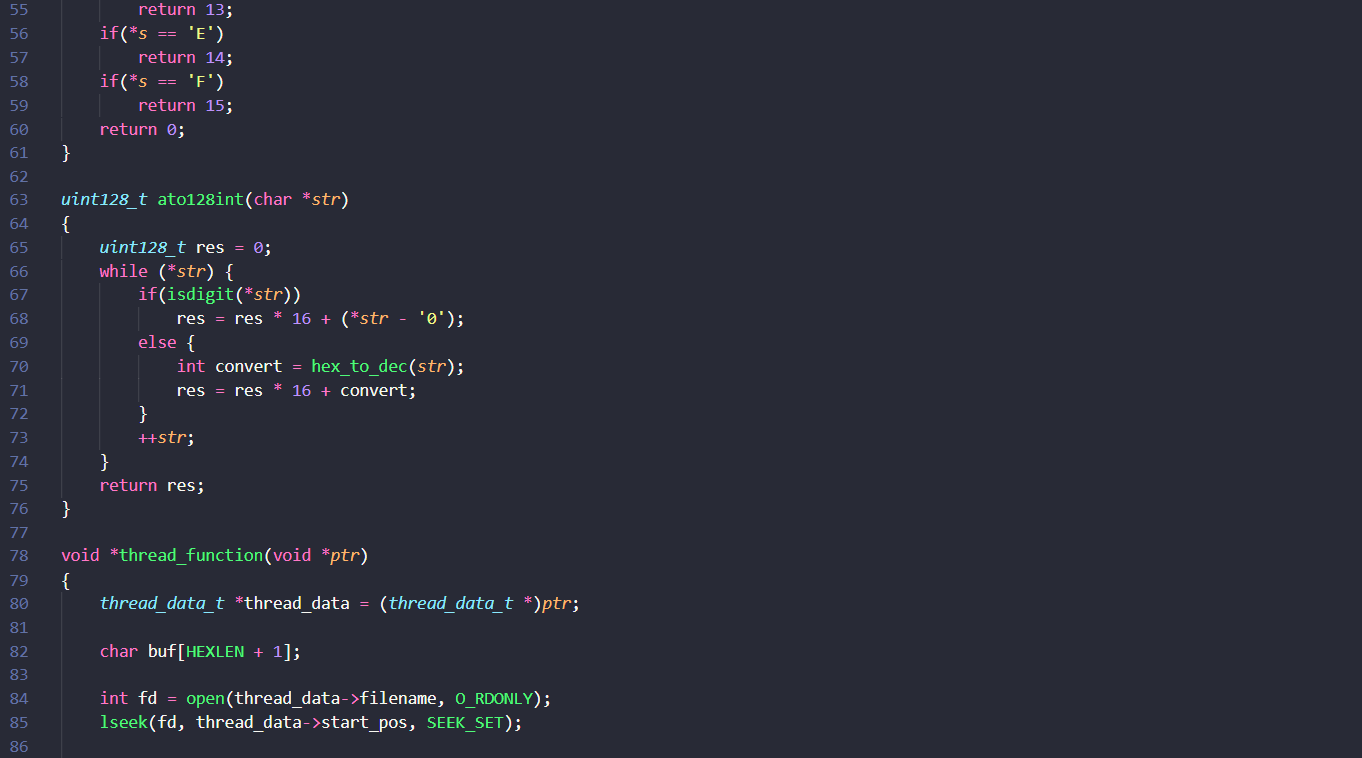
**Общий метод и алгоритм решения**

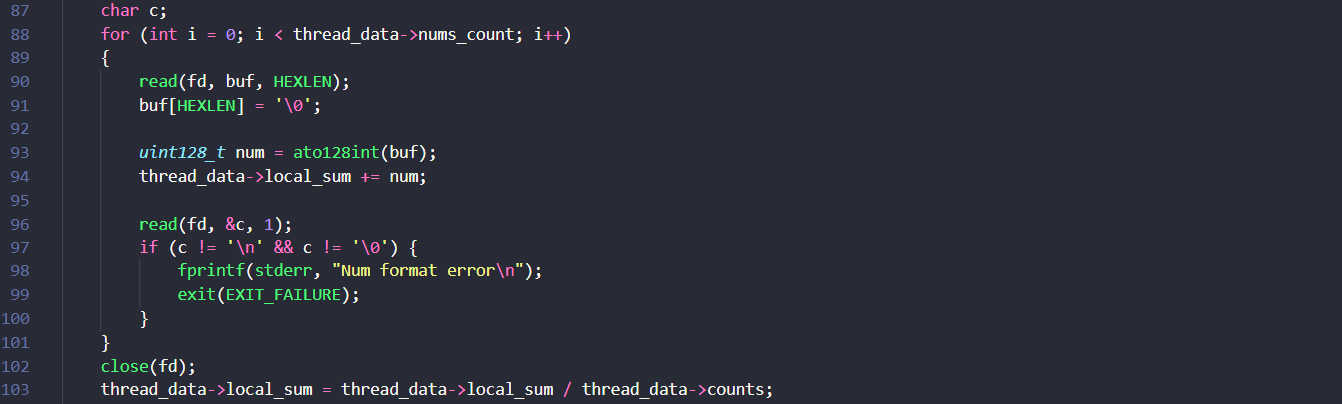
Программа на вход получает имя файла, в котором лежат необходимые нам числа (файл так же создается через определенную программу, написанную нами, которая генерирует набор случайных 128-битных чисел), а также количество потоков и количество памяти для них. Сначала мы делаем проверку, хватит ли нам заданной памяти для создания потоков или нет. Если нет, то завершаем программу. Также мы делаем проверку на то, слишком ли много потоков создано для данного количества чисел. Если да, зовершаем программу. Далее мы создаем две структуры, одна хранит в себе массив из потоков, а другая локальные данные для каждого из потоков. Каждый поток подсчитывает свою локальную сумму, после чего в конце работы программы локальные суммы складываются и подсчитывается среднее арифметическое.

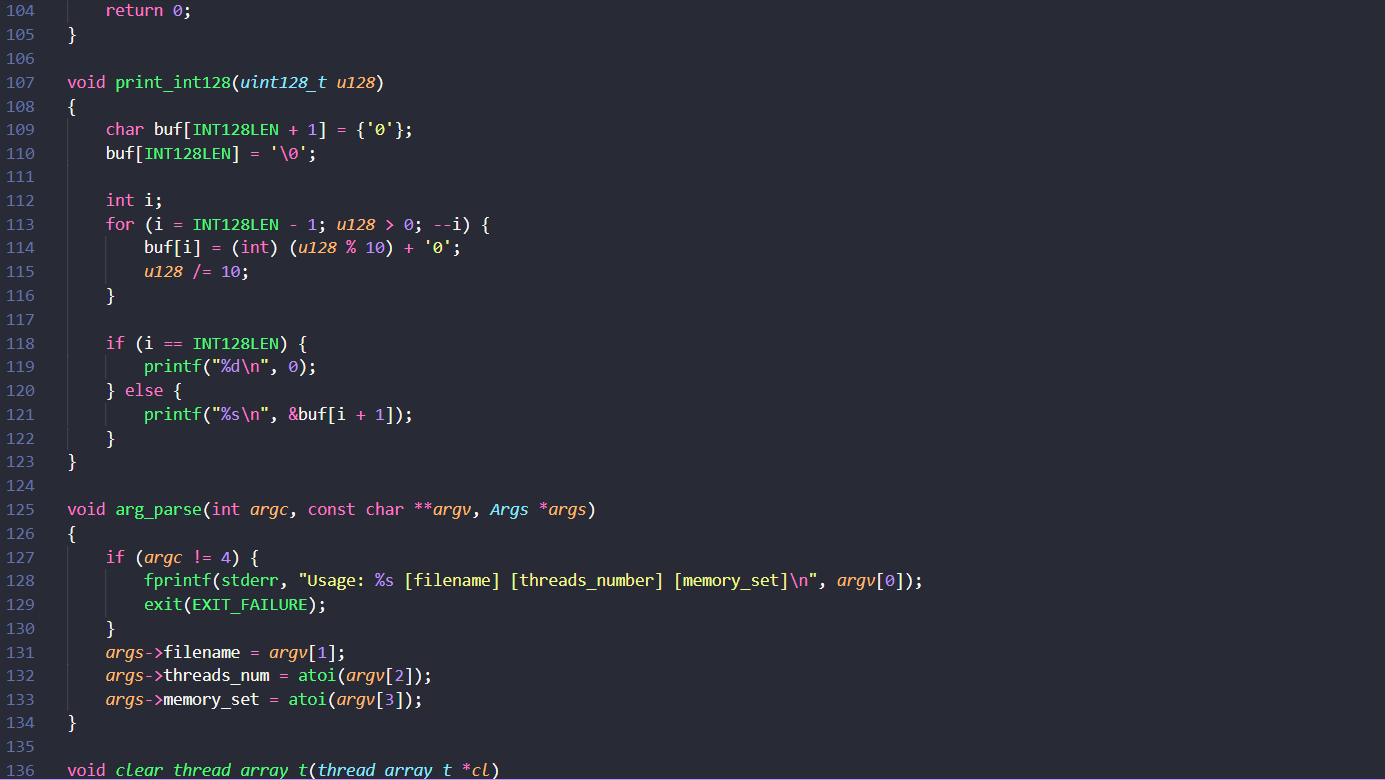
**Исходный код**

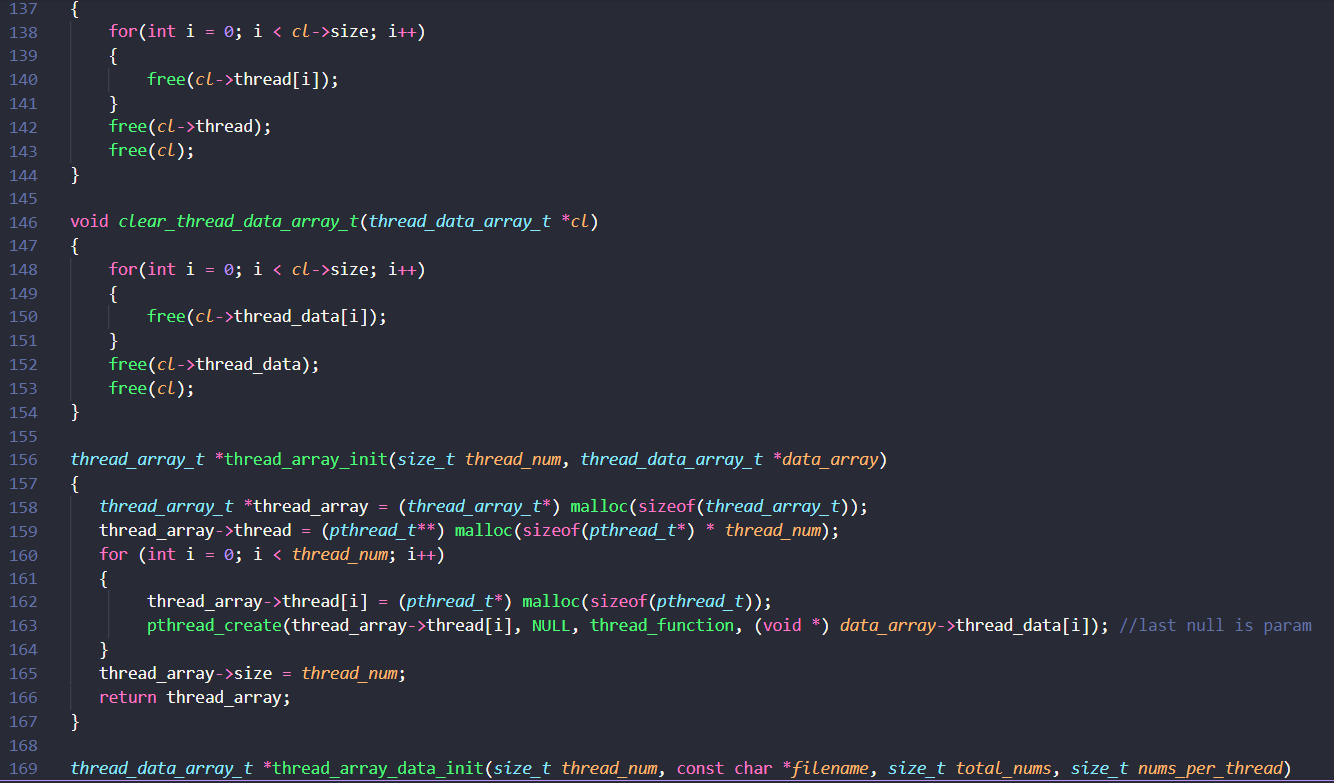




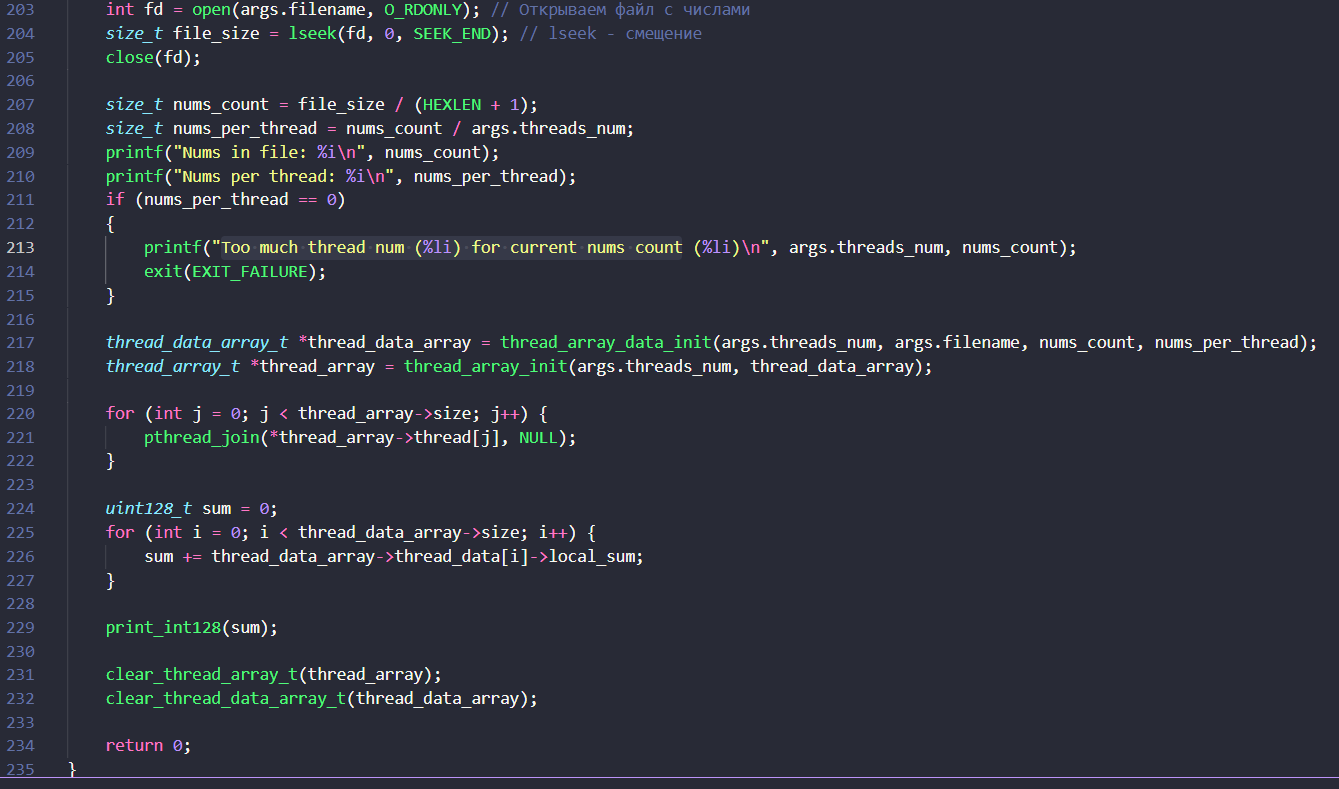






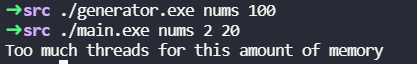




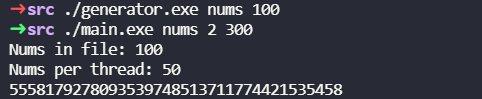


**Демонстрация работы программы**

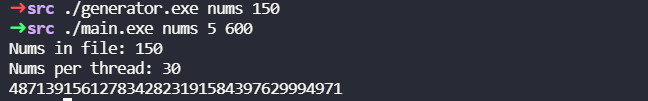
Тест 1:



Тест 2:



Тест 3:



**Выводы**

Благодаря данной лабораторной я успешно ознакомился с работой потоков в Linux и тем, как они устроены. Во время выполнения своего задания я изучил многие системные вызовы и научился применять их в своей программе, а также я узнал многие тонкости работы с потоками.