Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

**Тема работы**

**“Межпроцессорное взаимодействие через memory-mapped files”**

Студент: Ханнанов Руслан Маратович

Группа: М8О-208Б-20

Вариант: 14

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/Naksen/OS>

**Постановка задачи**

Составить и отладить программу на языке С\C++, осуществляющую работу с процессами и взаимодействие между ними в одной из двух операционных систем. В результате работы программа (основной процесс) должен создать для решения задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через отображаемые файлы (memory-mapped files).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

**Общие сведения о программе**

Программа написана на языке C++ в UNIX-подобной операционной системе. Для компиляции требуется указать ключ -pthread и -lrt.

Сборка проекта происходит при помощи Сmake-файла.

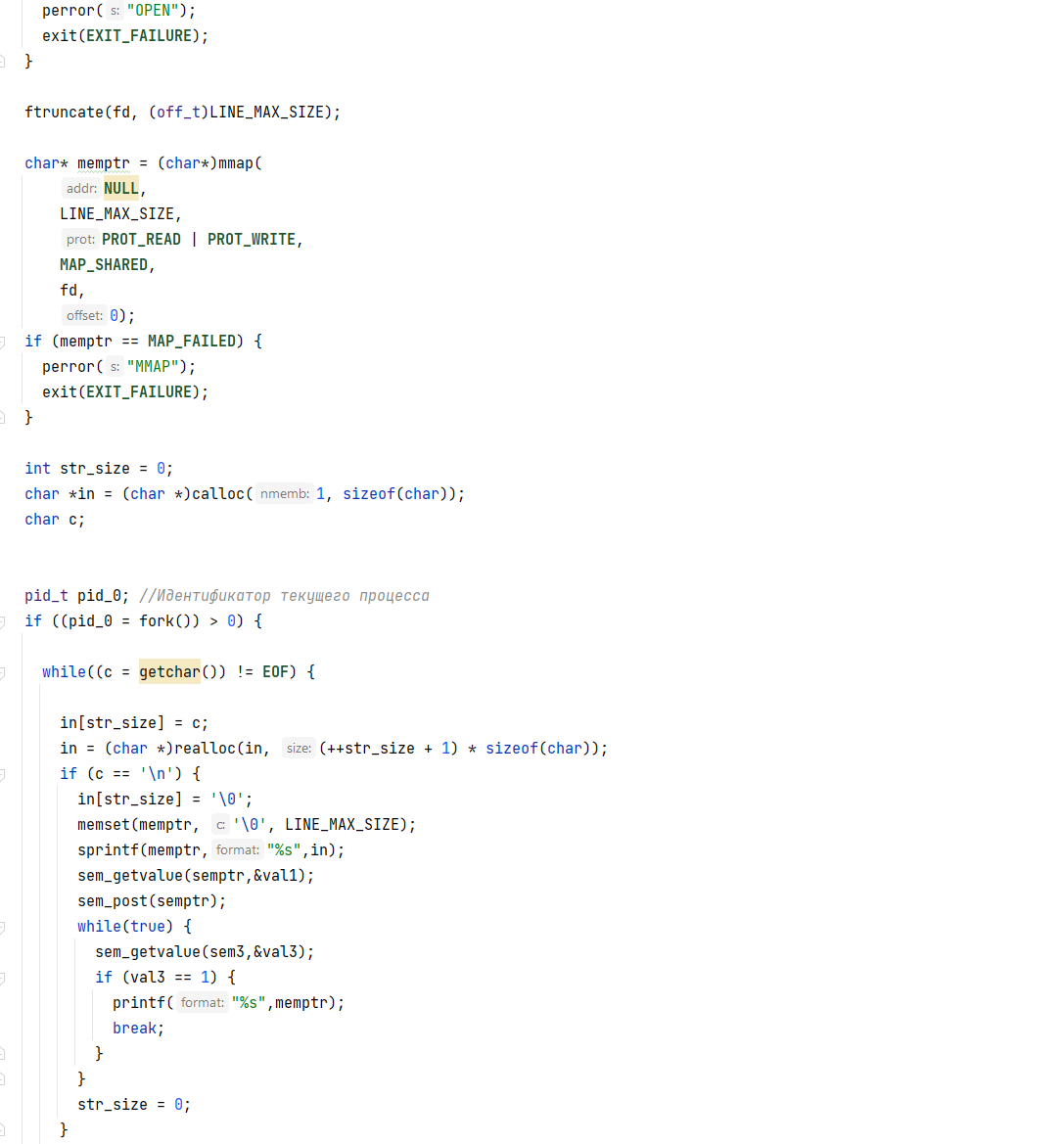
**Общий метод и алгоритм решения**

Программа создает и отражает на общую для процессов память буферный файл. В этот буферный файл построчно считывается информация, его размер при это фиксированный и заранее установленный равным максимальной длине строки. В программе используется три семафора: первый – для общения между родительский и первым дочерним процессами, второй – для общения между первым и вторым дочерними процессами, третий – для общения между вторым дочерним и родительским процессом. Как только считывается новая строка, дочерни процесс понимает, что появились данные, которые нужно обработать, он переводит строку в нижний регистр и передает сигнал о том, что нужно строку нужно обработать второму дочернему процессу, после чего второй дочерний процесс убирает задвоенные пробелы и сигнализирует с помощью третьего семафора о том, что строка полностью обработана. Родительский процесс понимает это и печатает обработанную строчку. Благодаря такому подходу в любой момент времени мы используем кусок памяти равный только лишь максимальной длине строки. После завершения снимает отображение файлов на память с помощью minmap и удаляем семафор функцией sem\_destroy.

**Исходный код**

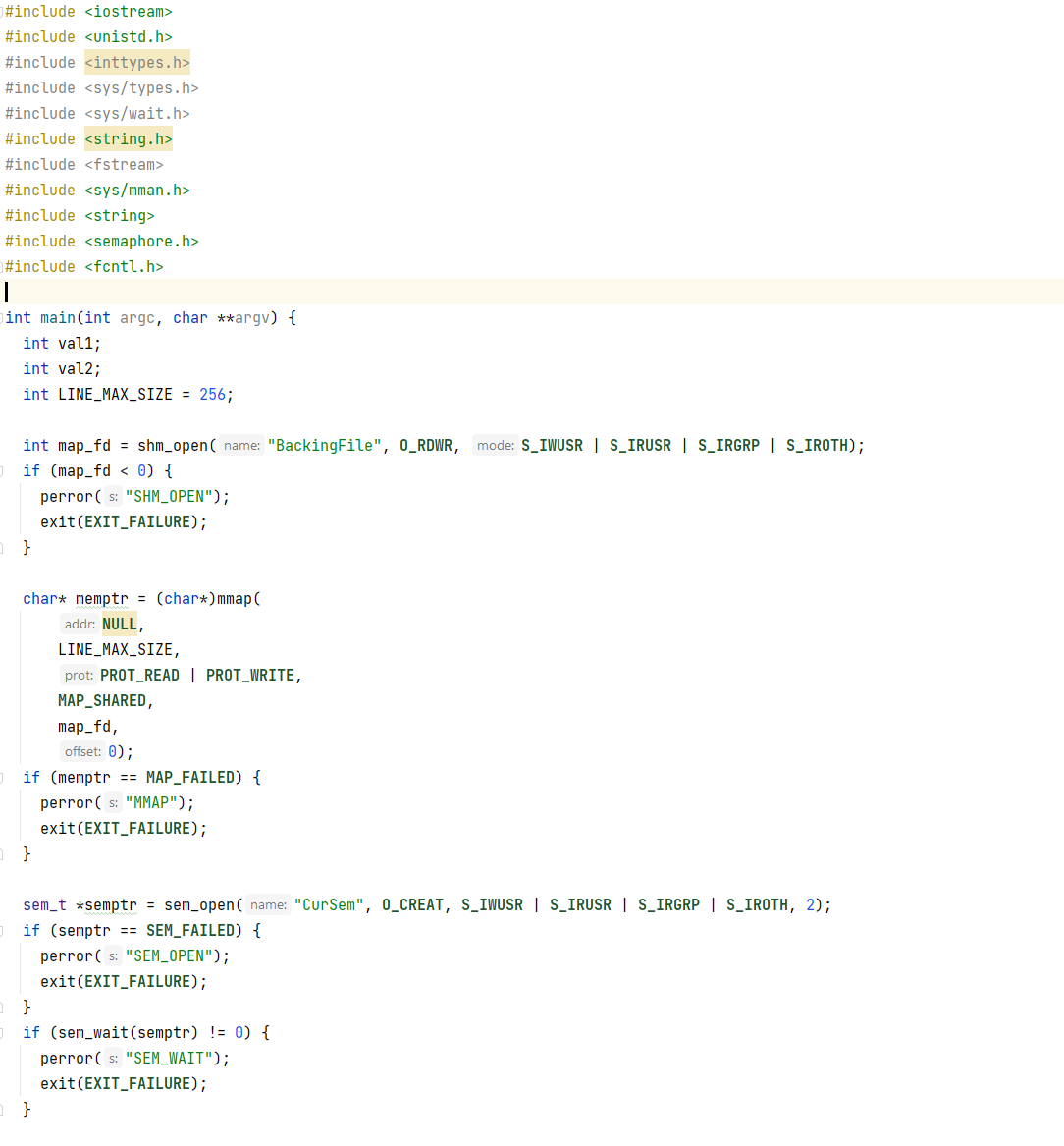
**main.cpp:**





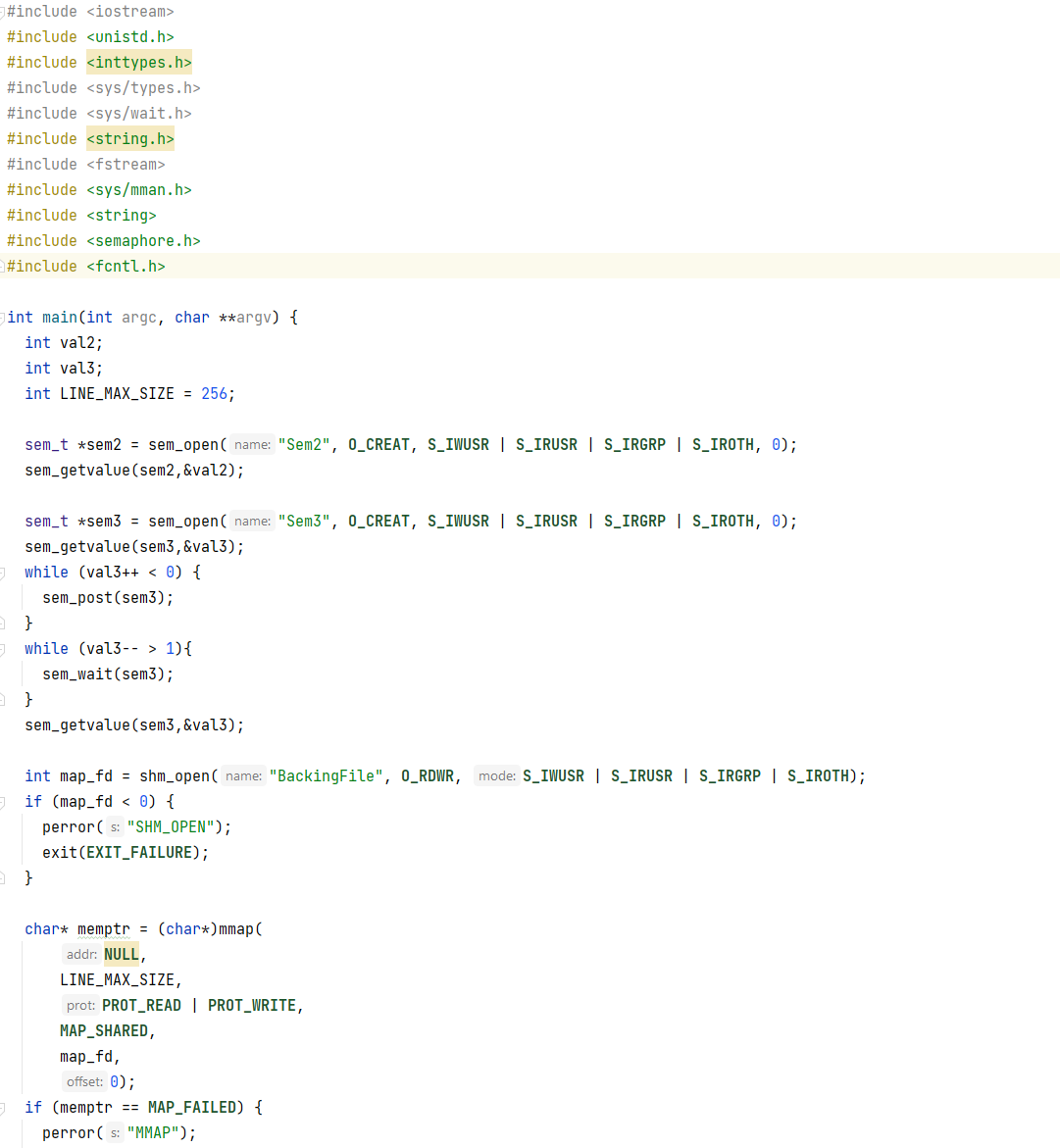


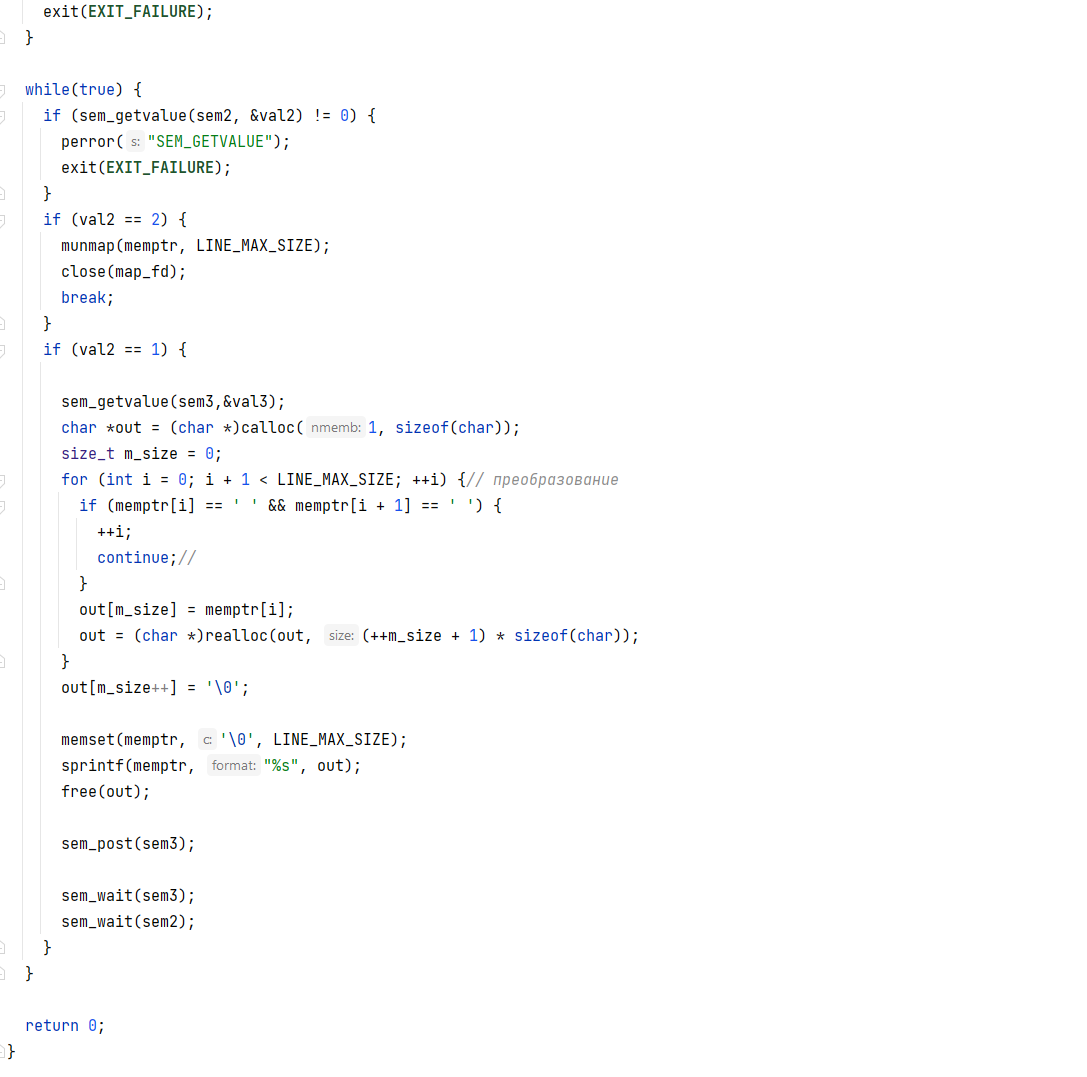
**child1.cpp:**



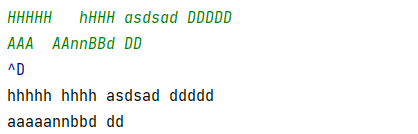


**child2.cpp:**





**Демонстрация работы программы**



**Выводы**

Эта лабораторная работа ознакомила и научила меня работать с расширяемой памятью. Научился синхронизировать работу процессов и поток с помощью семафоров. В отличие от лабораторной работы №2, где мы вызывали read и write, взаимодействие между процессами через mmapep-files происходит эффективнее и требует меньше памяти.