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: Каширин Кирилл Дмитриевич

Группа: М8О-208Б-20

Вариант: 8

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/lumses/OS>

**Постановка задачи**

Составить и отладить программу на языке Си, 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 С++ в UNIX-подобной операционной системе. Для компиляции требуется указать ключ –pthread и -lrt.

Сборка проекта происходит при помощи make-файла  
g++ -g -Wall -std=c++17 -pthread lab4.cpp -lrt -fsanitize=address

**Общий метод и алгоритм решения**

Программа на вход требует названия файла. Если такого файла не существует программа сразу завершается. Создаём два семафора, которые будут регулировать взаимодействие между дочерним и родительским процессором. Также создаем два файловых дескриптора, с помощью которых будет делать отображение на память вызовом mmap. Считываем построчно информацию из файла и передаем от родительского процессора через memptr1 дочернему. Он обрабатывает строку, полученную из memptr1 и результат кладёт в memptr2, который передаёт информацию из дочернего процесса в родительский. После завершения снимаем отображение файлов на память с помощью munmap и удаляем семафор функцией sem\_destroy.

**Исходный код**

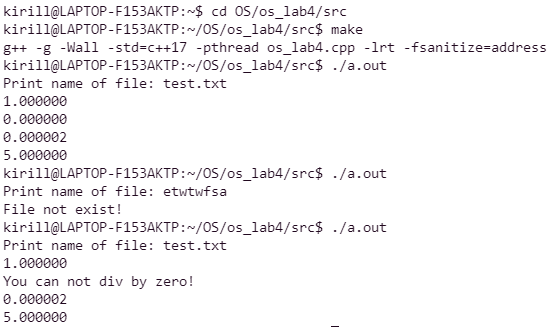
**os\_lab4.cpp**

****

****

****

******Демонстрация работы программы**



**Выводы**

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