Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №4 по курсу**

**«Операционные системы»**

Студент: Казанцев Данила Игоревич

Группа: М8О-207Б-21

Вариант: 3

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/thgdanilaya/mai\_os\_labs

**Постановка задачи**

**Цель работы**

Приобретение практических навыков в:

* Освоение принципов работы с файловыми системами
* Обеспечение обмена данных между процессами посредством технологии «File mapping»

**Задание**

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 решение задачи один или несколько дочерних процессов. Взаимодействие между процессами осуществляется через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

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

**Общие сведения о программе**

Программа компилируется из файла main.cpp

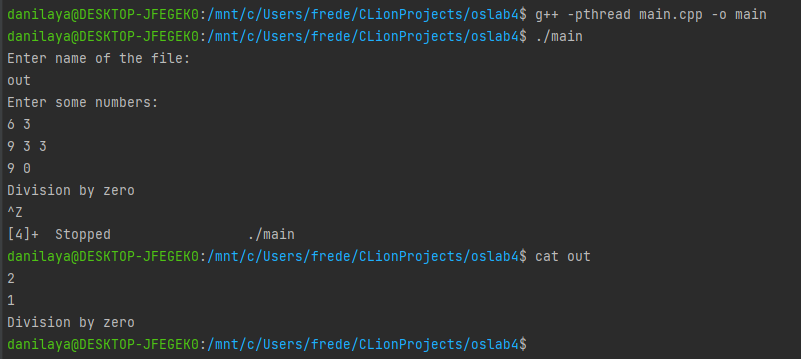
**Общий метод и алгоритм решения**

Создал структуру asd, чтобы хранить полученные от пользователя данные и операцию дочернего процесса. Создал для структуры отображенную память, доступную для обоих процессов, а для регулировки доступа процессов к памяти использую семафор.

**Исходный код**

#include <iostream>  
#include <fstream>  
#include <stdio.h>  
#include <unistd.h>  
#include <sys/mman.h>  
#include <fcntl.h>  
#include <semaphore.h>  
  
using namespace std;  
  
int human\_get(sem\_t \*semaphore) {  
 int s;  
 sem\_getvalue(semaphore, &s);  
 return s;  
}  
  
void human\_set(sem\_t \*semaphore, int n) {  
 while (human\_get(semaphore) < n) {  
 sem\_post(semaphore);  
 }  
 while (human\_get(semaphore) > n) {  
 sem\_wait(semaphore);  
 }  
}  
  
struct abc {  
 int num;  
 int st;  
};  
  
int main() {  
 int ans = 0;  
 abc \*mapped = (abc \*) mmap(0, sizeof(abc), PROT\_READ | PROT\_WRITE, MAP\_SHARED | MAP\_ANONYMOUS, 0, 0);  
 if (mapped == MAP\_FAILED) {  
 cout << "mmap error\n";  
 return -1;  
 }  
 sem\_unlink("\_sem");  
 sem\_t \*sem = sem\_open("\_sem", O\_CREAT, 0, 2);  
 string filename;  
 int n;  
 ofstream out;  
 cout << "Enter name of the file:\n";  
 getline(cin, filename);  
 cout << "Enter some numbers:\n";  
 int id = fork();  
 if (id < 0) {  
 cout << "fork error\n";  
 return -1;  
 }  
 if (id == 0) {  
 out.open(filename);  
 while (true) {  
 while (human\_get(sem) == 2) {  
 continue;  
 }  
 if (mapped->st == 1) {  
 if (mapped->num == 0) {  
 cout << "Division by zero\n";  
 out << "Division by zero" << endl;  
 exit(1);  
 } else {  
 ans /= mapped->num;  
 }  
 out << ans << endl;  
 ans = 0;  
 human\_set(sem, 2);  
 } else if (mapped->st == 2) {  
 if (mapped->num == 0) {  
 cout << "Division by zero\n";  
 out << "Division by zero" << endl;  
 exit(1);  
 } else {  
 ans /= mapped->num;  
 }  
 out << ans << endl;  
 out.close();  
 human\_set(sem, 0);  
 exit(0);  
 } else if (mapped->st == 0) {  
 if (ans == 0) {  
 ans = mapped->num;  
 } else if (mapped->num == 0) {  
 cout << "Division by zero\n";  
 out << "Division by zero" << endl;  
 exit(1);  
 } else {  
 ans /= mapped->num;  
 }  
 human\_set(sem, 2);  
 }  
 }  
 } else if (id > 0) {  
 while (human\_get(sem) != 0) {  
 char c;  
 scanf("%d%c", &n, &c);  
 mapped->num = n;  
 if (c == ' ') {  
 mapped->st = 0;  
 }  
 if (c == '\n') {  
 mapped->st = 1;  
 }  
 if (c == '\0') {  
 mapped->st = 2;  
 }  
 human\_set(sem, 1);  
 while (human\_get(sem) == 1) {  
 continue;  
 }  
 }  
 }  
 munmap(mapped, sizeof(abc));  
 sem\_close(sem);  
 sem\_destroy(sem);  
 return 0;  
}

**Демонстрация работы программы**



**Выводы**

Я приобрел практические навыки, необходимые для работы с отображаемой памятью и семафорами.