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**

Студент: Москвин Артём Артурович

Группа: М8О-208Б-20

Вариант: 1

Преподаватель: Миронов Евгений Сергеевич

Оценка: \_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_

Москва, 2021

**Содержание**

1. Репозиторий
2. Постановка задачи
3. Общие сведения о программе
4. Исходный код
5. Демонстрация работы программы
6. Выводы

**Репозиторий**

<https://github.com/Pert002/os_lab>4

**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 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.

**Вариант 1**: Пользователь вводит команды вида: «число число число<endline>». Далее эти числа передаются от родительского процесса в дочерний. Дочерний процесс считает их сумму и выводит её в файл. Числа имеют тип int. Количество чисел может быть произвольным.

**Общие сведения о программе**

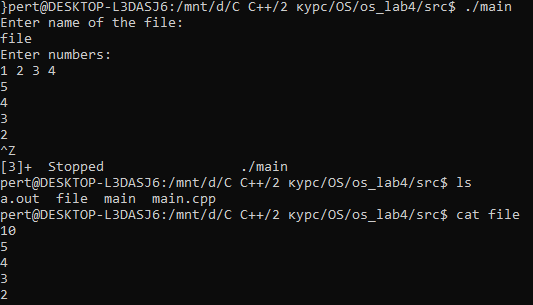
Программа представляет из себя один файл main.cpp.

**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 get(sem\_t \*semaphore) { int sval; sem\_getvalue(semaphore, &sval); //помещает текущее значение семафора, заданного в semaphore, в виде целого, на которое указывает sval. return sval; } void set(sem\_t \*semaphore, int n) { while (get(semaphore) < n) { sem\_post(semaphore); //увеличивает (разблокирует) семафор, на который указывает semaphore } while (get(semaphore) > n) { sem\_wait(semaphore); //уменьшает (блокирует) семафор, на который указывает semaphore } } struct mmf { int num; int st; }; int main() { int sum = 0; mmf\* mapped = (mmf\*)mmap(0, sizeof(mmf), PROT\_READ|PROT\_WRITE, MAP\_SHARED|MAP\_ANONYMOUS, 0, 0); if (mapped == MAP\_FAILED) { cout << "mmap error\n"; return -1; } sem\_unlink("\_sem"); //удаляет именнованный семафор sem\_t \*sem = sem\_open("\_sem", O\_CREAT, 0, 2); /\* создаёт новый семафор; флаг O\_CREAT означает, что семафор создаётся, если ещё не существует; в 3 значении (0) задаются права для нового семафора; в 4 значении (2) задаётся начальное значение нового семафора \*/ string filename; int n; ofstream outfile; cout << "Enter name of the file:\n"; getline(cin, filename); cout << "Enter numbers:\n"; int id = fork(); if (id < 0) { cout << "fork error\n"; return -1; } if (id == 0) { outfile.open(filename); while(1) { while(get(sem) == 2) { continue; } if (mapped->st == 1) { sum += mapped->num; outfile << sum << endl; sum = 0; set(sem, 2); } else if (mapped->st == 2) { sum += mapped->num; outfile << sum << endl; outfile.close(); set(sem, 0); exit(0); } else if (mapped->st == 0) { sum += mapped->num; set(sem, 2); } } } else if (id > 0) { while(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; cout << "123\n"; } set(sem, 1); while(get(sem) == 1) { continue; } } } munmap(mapped, sizeof(mmf)); sem\_close(sem); sem\_destroy(sem); return 0; }

**Демонстрация работы программы**

Ввод в консоль:



**Выводы**

Проделав лабораторную работу, я приобрёл практические навыки, необходимые для работы с отображаемой памятью и семафорами.