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: Попов Матвей Романович

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. Общий метод и алгоритм решения
5. Исходный код
6. Демонстрация работы программы
7. Выводы

**Репозиторий**

[https://github.com/.../os\_lab](https://github.com/.../os_lab2)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.

**Общий метод и алгоритм решения**

Создаём структуру bebra, в которой будем хранить полученное от пользователя число и операцию дочернего процесса. Создадим для структуры bebra отображённую память, доступную для обоих процессов, а для регулировки доступа процессов к памяти используем 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 bebra

{

    int num;

    int st;

};

int main()

{

    int sum = 0;

    bebra\* mapped = (bebra\*)mmap(0, sizeof(bebra), 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(1)

        {

            while(human\_get(sem) == 2)

            {

                continue;

            }

            if (mapped->st == 1)

            {

                sum += mapped->num;

                out << sum << endl;

                sum = 0;

                human\_set(sem, 2);

            }

            else if (mapped->st == 2)

            {

                sum += mapped->num;

                out << sum << endl;

                out.close();

                human\_set(sem, 0);

                exit(0);

            }

            else if (mapped->st == 0)

            {

                sum += 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(bebra));

    sem\_close(sem);

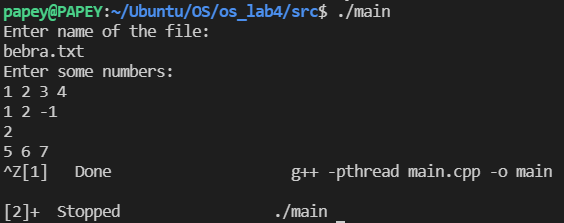
    sem\_destroy(sem);

    return 0;

}

**Демонстрация работы программы**

Ввод в консоль:



Содержимое файла bebra.txt:

10

2

2

18

**Выводы**

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