Московский Авиационный Институт
(Национальный Исследовательский Университет)
Факультет информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

**Лабораторная работа №0 по курсу**

**«Операционные системы»**

**Тема работы**

**“Освоение технологии «File mapping»”**

Студент: Зинин Владислав Владимирович
Группа: М8О-208Б-20
Вариант:5
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

**Содержание**

**Репозиторий**

https://github.com/frankeloff/OS

**Постановка задачи**

Задача: Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Результаты своей работы дочерний процесс пишет в созданный им файл. Пользователь вводит команды вида: «число». Далее это число передается от родительского процесса в дочерний. Дочерний процесс производит проверку на простоту. Если число составное, то в это число записывается в файл. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются. Обеспечить обмен данных между процессами посредством технологии «File mapping»

**Общие сведения о программе**

int id = fork () - создание дочернего процесса, в переменной id будет лежать “специальный код” процесса (-1 - ошибка fork, 0 - дочерний процесс, >0 - родительский);

void * mmap(void *_start_, size_t _length_, int _prot_ , int _flags_, int _fd_, off_t _offset_) где Функция mmap отражает _length_ байтов, начиная со смещения _offset_ файла (или другого объекта), определенного файловым описателем _fd_, в память, начиная с адреса _start_. Последний параметр (адрес) необязателен, и обычно бывает равен 0. Настоящее местоположение отраженных данных возвращается самой функцией mmap, и никогда не бывает равным 0. При удачном выполнении mmap возвращает указатель на область с отраженными данными. При ошибке возвращается значение MAP_FAILED (-1), а переменная _errno_ приобретает соответствующее значение. Флаги, использованные в работе:

PROT_READ|PROT_WRITE для записи и чтения

MAP_SHARED|MAP_ANONYMOUS Первый флаг нужен для того, чтобы разделить использование с другими процессами. Второй флаг обозначает следующее: отображение не резервируется ни в каком файле. Он необходим, поскольку мы не используем файл.

int munmap(void *_start_, size_t _length_) - удаляет все отражения из заданной области памяти, после чего все ссылки на данную область будут вызывать ошибку "неправильное обращение к памяти" (invalid memory reference). Отражение удаляется автоматически при завершении процесса. С другой стороны, закрытие файла не приведет к снятию отражения. Start – адрес, по которому необходимо очистить память, length – размер памяти, которую необходимо очистить. При удачном выполнении munmap возвращаемое значение равно нулю. При ошибке возвращается -1, а переменная *errno* приобретает соответствующее значение. (Вероятнее всего, это будет EINVAL).

**Общий метод и алгоритм решения**

Программа получает на вход имя файла, потом число. Число и кодовое слово записываются родительским процессом в массив, созданный через mmap, после чего родительский процесс ждет дочерний, тот в свою очередь проверяет число. Если оно удовлетворяет условиям задачи (составное), то оно записывается в файл, дочерний процесс записывает в массив кодовое слово и родительский процесс продолжает свою работу, дочерний ожидает. Если же число не соответствует условиям задачи (простое), то дочерний процесс записывает кодовое слово STOP в массив и завершается, после чего родительский процесс принимает кодовое слово и завершается.

## Исходный код

lab1.cpp

```cpp
#include <iostream>
#include <unistd.h>
#include <inttypes.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <string.h>
#include <fstream>
#include <sys/mman.h>
#define WRITE 1
#define WAIT 2
#define STOP 0

int32_t primaryTest(int32_t n)
{
    for (int i = 2; i <= n / 2; i++)
        if (n % i == 0)
            return 0;

    return 1;
}

int main()
{
    int32_t *buf;
    buf = (int32_t*)mmap(0, sizeof(int32_t) * 2, PROT_READ|PROT_WRITE, MAP_SHARED|MAP_ANONYMOUS, 0, 0); //MAP_ANONYMOUS - область памяти,
    if (buf == MAP_FAILED) {
        std:: cout << "An error with mmap function has been detected" << std:: endl;
        return EXIT_FAILURE;
    }
    buf[1] = WAIT;
    pid_t pid; // Идентификатор текущего потока

    std::string fileName; // Имя файла
    int32_t number; // Полученное число
    std::ofstream out; // Поток вывода

    std::cout << "Введите имя файла: ";
    std::getline(std::cin, fileName);

    switch (pid = fork())
    {
    case -1: // Ошибка создания потока
        std::cout << "При создании потока произошла ошибка!";
        return 1;

    case 0:  // Код потомка
        while(1) {
            while(buf[1] == WAIT) continue;
            if (buf[0] <= 0 || primaryTest(buf[0]))
            {
                std::cout << "Дочерний процесс завершился\n";
                buf[1] = STOP;
                exit(0);
            }
            else {
                out.open(fileName, std::ios_base::app); //app - добавить в конец
                out << buf[0] << "\n";
                out.close();
                std::cout << "Число добавлено" << '\n';
                buf[1] = WAIT;
            }
        }

    default: // Код родителя
        while(1) {
            // std::getline(std::cin, line);
            std::cin >> number;
            // number = atoi(line.data());
            buf[0] = number;
            buf[1] = WRITE;
            while(buf[1] == WRITE) continue;
            if (buf[1] == STOP)
            {
                munmap(buf, sizeof(int32_t) * 2);
                std::cout << "Родительский процесс завершился" << std::endl;
                return 0;
            }
        }
    }
}
```

## Демонстрация работы программы

Тест 1:

```
→src ./a.out
Введите имя файла: sadf
12
Число добавлено
13
Дочерний процесс завершился
Родительский процесс завершился
```

Тест 2:

```
→src ./a.out
Введите имя файла: asd
-1
Дочерний процесс завершился
Родительский процесс завершился
```

Тест 3:

```
→src ./a.out
Введите имя файла: фыв
12
Число добавлено
0
Дочерний процесс завершился
Родительский процесс завершился
```

## Выводы

Благодаря данной лабораторной работе я освоил принцип работы с файловыми системами и научился обеспечивать обмен данными между процессами посредством технологии «File mapping».