Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №3 по курсу**

**«Операционные системы»**

**РАБОТА С ОБЩЕЙ ПАМЯТЬЮ**

Студент: Козырев Пётр Андреевич

Группа: М8О–212Б–22

Вариант: 10

Преподаватель: Соколов Андрей Алексеевич

Оценка: \_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_

Москва, 2023.

**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.

Вариант 10:

В файле записаны команды вида: «число». Дочерний процесс производит проверку этого числа на простоту. Если число составное, то дочерний процесс пишет это число в стандартный поток вывода. Если число отрицательное или простое, то тогда дочерний и родительский процессы завершаются. Количество чисел может быть произвольным.

**Общие сведения о программе**

Программа компилируется из файла main.cpp. Помимо этого, есть программа child.cpp, которая запускается при помощи execlp. Обе программы собираются при помощи системы сборки CMake.

**Общий метод и алгоритм решения**.

Для реализации поставленной задачи необходимо:

1. Считать название тестового файла
2. C помощью функции fstat заполнить структуру stat; узнать размер файла
3. С помощью функции mmap отразить файл в память
4. Создать дочерний процесс
5. Дождаться завершения работы алгоритма в дочернем процессе
6. Очистить память и закрыть файл

**Основные файлы программы**

**Main.cpp**

#include <sys/types.h>

#include <sys/stat.h>

#include <fcntl.h>

#include <unistd.h>

#include <sys/mman.h>

#include <sys/wait.h>

#include <signal.h>

#include <iostream>

#include <string>

int main(int argc, char \*argv[]) {

    if(argc != 2){

        std::cerr << "No name test file" << std::endl;

        return 1;

    }

     // Путь дочернего процесса

    const char\* CHILD\_PATH = "child";

    // Путь тестируемого файла

    std::string path {"../../test/"};

    std::string name {argv[1]};

    path += name;

    // Открываем файл для чтения и записи

    int fd = open(path.c\_str(), O\_RDONLY);

    if (fd == -1) {

        std::cerr << "Error not open file" << std::endl;

        return 1;

    }

    // Определяем размер файла

    struct stat fileStats;

    if (fstat(fd, &fileStats) == -1) {

        std::cerr << "Failed to get file information" << std::endl;

        close(fd);

        return 1;

    }

    // Выполняем отображение файла в память

    char\* fileData = (char\*)mmap(NULL, fileStats.st\_size, PROT\_READ, MAP\_SHARED, fd, 0);

    if (fileData == MAP\_FAILED) {

        std::cerr << "Failed to map the file into memory" << std::endl;

        close(fd);

        return 1;

    }

    // Create fork

    pid\_t pid = fork();

    if( pid == -1){

        perror("Pid errror\n");

        return 1;

    }

    if( pid == 0){

        // child process

        std::string size = std::to\_string(fileStats.st\_size);

        execl(CHILD\_PATH, fileData,size.c\_str(),NULL);

        exit(EXIT\_SUCCESS);

    }

    else{

        int status;

        waitpid(pid, &status, 0);

        if (WIFEXITED(status)) {

            int exitStatus = WEXITSTATUS(status);

            std::cout << "\nChild process exited with status: " << exitStatus << std::endl;

        } else if (WIFSIGNALED(status)) {

            int signalNumber = WTERMSIG(status);

            std::cout << "\nChild process was terminated by signal: " << signalNumber << std::endl;

        }

    }

    // Закрываем файл и освобождаем память

    if (munmap(fileData, fileStats.st\_size) == -1) {

        std::cerr << "Failed to free memory" << std::endl;

    }

    close(fd);

    return 0;

}

**Child.cpp**

#include "prime.cpp"

#include <unistd.h>

#include <fcntl.h>

#include <iostream>

#include <string>

#include <fstream>

int main(int argc, char\* argv[]){

    std::string fileData = argv[0];

    int size = atoi(argv[1]);

    int i = 0;

    while(i < size){

        int sign = 1;

        if(fileData[i] == '-'){

            sign = -1;

            i++;

        }

        if(fileData[i] >= '0' and fileData[i]<='9'){

            std::string t = "";

            t += fileData[i];

            i++;

            while(i<size and fileData[i] >= '0' and fileData[i]<='9'){

                t += fileData[i];

                i++;

            }

            int r = std::stoi(t);

            r \*= sign;

            if(isPrime(r)){

                    std::cout << r;

                 std::cout << " Prime" << std::endl;

            }

             else

                 return 0;

        }

        else

            ++i;

    }

    return 0;

}

**Пример работы**

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

**Вывод**

В данной лабораторной работе были изучены принципы работы с файловыми системами и освоены навыки обмена данными между процессами с использованием технологии "File mapping".

Для достижения поставленных целей была составлена и отлажена программа на языке C++, которая осуществляет работу с процессами и обменивается данными между ними в выбранной операционной системе. Основной процесс создает один или несколько дочерних процессов для решения поставленной задачи. Взаимодействие между процессами происходит через системные сигналы/события и/или через отображаемые файлы (memory-mapped files).

Особое внимание было уделено обработке системных ошибок, которые могут возникнуть в результате работы программы. Это позволяет программе быть более надежной и устойчивой к возможным сбоям на уровне операционной системы.

В результате выполнения лабораторной работы были достигнуты поставленные цели - были освоены принципы работы с файловыми системами и технология "File mapping", а также была разработана и отлажена программа, обеспечивающая работу с процессами и взаимодействие между ними.