Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа № 2 по курсу**

**«Операционные системы»**

Студент: Румынина Екатерина Александровна

Группа: М8О-201Б-21

Вариант: 7

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/KetRum0/mai_os_labs>

**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 каналы (pipe). Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Вариант 7: Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса вводит имя файла, которое будет использовано для открытия файла с таким именем на чтение. Стандартный поток ввода дочернего процесса переопределяется открытым файлом. Дочерний процесс читает команды из стандартного потока ввода. Стандартный поток вывода дочернего процесса перенаправляется в pipe1. Родительский процесс читает из pipe1 и прочитанное выводит в свой стандартный поток вывода. Родительский и дочерний процесс должны быть представлены разными программами. В файле записаны команды вида: «число число число». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип float. Количество чисел может быть произвольным.

**Общие сведения о программе**

main.c - основная программа, которая считывает ввод и перенаправляет его в родительский процесс

parent.c – программа для реализации родительского процесса

child.c – программа для реализации дочернего процесса

parent.h – заголовочный файл с объявлением ParentRoutine

**Общий метод и алгоритм решения**

В родительском процессе создается pipe и дочерний процесс с помощью системного вызова fork. Дочерний процесс с помощью dup2 перенаправляет файловый дескриптор STDOUT на запись в pipe, а затем запускает программу child.c с помощью execlp. В child.c выполняется задание по варианту и ответ записывается в pipe, из которого затем его прочитает родительский процесс.

**Исходный код**

**main.c**

#include "parent.h"

#include <stdio.h>

#include <stdlib.h>

int main(void) {

char name[256];

scanf("%s",name);

float result;

result = ParentRoutine(name);

printf("%g\n",result);

return 0;

}

**parent.h**

#ifndef OS\_LABS\_PARENT\_H

#define OS\_LABS\_PARENT\_H

#include <stdio.h>

float ParentRoutine(char \*filename);

#endif //OS\_LABS\_PARENT\_H

**parent.c**

#include <unistd.h>

#include <stdlib.h>

#include <stdio.h>

#include <fcntl.h>

int ParentRoutine(char \*filename){

    // char name[256];

    // scanf("%s",name);

    int file;

    if ((file = open(filename, O\_RDONLY))==-1){

        perror("open error");

        exit(EXIT\_FAILURE);

    }

    int fd[2];

    if (pipe(fd) == -1){

        perror("pipe error");

        exit(EXIT\_FAILURE);

    }

    int id = fork();

    if (id == -1){

        perror("fork error");

        exit(EXIT\_FAILURE);

    } else if (id == 0){ //child

        close(fd[0]);

        dup2(file,STDIN\_FILENO);

        close(file);

        dup2(fd[1],STDOUT\_FILENO);

        close(fd[1]);

        if (execlp("./child", "./child", NULL) == -1){

            perror("execlp error");

            exit(EXIT\_FAILURE);

        }

    } else { //parent

        close(fd[1]);

        close(file);

        float result;

        if (read(fd[0], &result, sizeof(float)) == -1){

            perror("read from pipe error");

            exit(EXIT\_FAILURE);

        }

        close(fd[0]);

        // printf("%g\n",result);

        return result;

    }

}

**child.c**

#include <unistd.h>

#include <stdio.h>

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

float res;

res = 0;

float x;

while (scanf("%f",&x)!=EOF){

res+=x;

}

write(1,&res,sizeof(float));

}

**Демонстрация работы программы**

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab2$ cat test.txt

2 3 4 -1 0.1 0.11

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab2$ ./lab2

test.txt

8.21

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab2$ cat test2.txt

1.1 2.2 3.33 -6 0

ket@ket-laptop:~/Desktop/mai\_os\_labs/lab2$ ./lab2

test2.txt

0.63

**Выводы**

В результате выполнения данной лабораторной работы я изучила управление процессами в ОС и обеспечение данными между процессами посредством каналов.