Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

**Лабораторная работа №2 по курсу**

**«Операционные системы»**

**Тема работы**

**Управление процессами в ОС.**

**Обеспечение обмена данных между процессами посредством каналов**

Студент: Шаларь Игорь Павлович

Группа: М8О-208Б-20

Вариант: 5

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/IgShalar/OS/tree/main/os_lab2>

**Постановка задачи**

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Пользователь вводит команды вида: «число<endline>». 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.

**Общие сведения о программе**

Запуск:

\_$ cmake CMakeLists.txt

\_$ make

\_$ ./parent

Написано для Unix.

**parent.cpp**: получает название файла, создаёт дочерний процесс и создаёт  pipe.

Системные вызовы:

fork() - создаёт дочерний процесс.

dup2() - дублирует дескриптор файла.

execl()- заменяет текущий образ процесса новым образом процесса: выполняет child.cpp и выделяет ему другую область памяти(память parent становится недоступна).

**child.cpp**: получает число и название файла. Создаёт файл и выполняет проверку на простоту.

Системные вызовы:

\_.open() - создаёт или открывает файл.

**Общий метод и алгоритм решения**

Проверка на простоту:

Проверяю делители ли все числа, большие 1, до корня числа включительно.

**Исходный код**

parent.cpp:

#include<iostream>

#include<fstream>

#include<unistd.h>

using namespace std;

int main(){

int p1[2], p2[2];

if (pipe (p1) == -1){

perror("pipe error");

return -1;

}

// if (pipe (p2) == -1){

// perror("pipe error");

// return -1;

// }

string t;

const char \* s;

getline(cin, t);

s = t.c\_str();

int a;

cin >> a;

int id = fork();

switch (id){

case -1: {

perror("fork error");

return -1;

}

case 0: {

close(p1[0]);

if (dup2(p1[1], 1) == -1){

perror("dup2 error");

return -1;

}

close(p1[1]);

cout << a << endl;

// if (dup2(p2[0], 0) == -1){

// perror("dup2 error");

// return -1;

// }

// close(p2[1]);

// close(p2[0]);

break;

}

default: {

// if (dup2(p2[1], 1) == -1){

// perror("dup2 error");

// return -1;

// }

// close(p2[1]);

// close(p2[0]);

close(p1[1]);

if (dup2(p1[0], 0) == -1){

perror("dup2 error");

return -1;

}

close(p1[0]);

if (execl("b.out", s, NULL) < 0){

perror("execl error");

return -1;

}

}

}

return 0;

}

child.cpp:

#include<fstream>

#include<iostream>

using namespace std;

int main(int a, char \* s[]){

ofstream f;

f.open(\* s);

if (!f){

perror("Can`t create file");

return -1;

}

int b, c = 2;

cin >> b;

while (c \* c <= b){

if (b % c == 0){

f << b;

f.close();

return 0;

}

c++;

}

f.close();

return 0;

}

**Демонстрация работы программы**

**1)Ввод:**

test.txt

20

**test.txt:**

20

**2)Ввод:**

test.txt

3

**test.txt:**

(пусто)

**3)Ввод:**

test.txt

-1

**test.txt:**

(пусто)

**Выводы**

Получил навыки создания дочерних процессов и передачи информации между ними с помощью pipe-ов.