

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: Мусаелян Ярослав
Группа: М8О-207Б-21
Вариант: 10
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

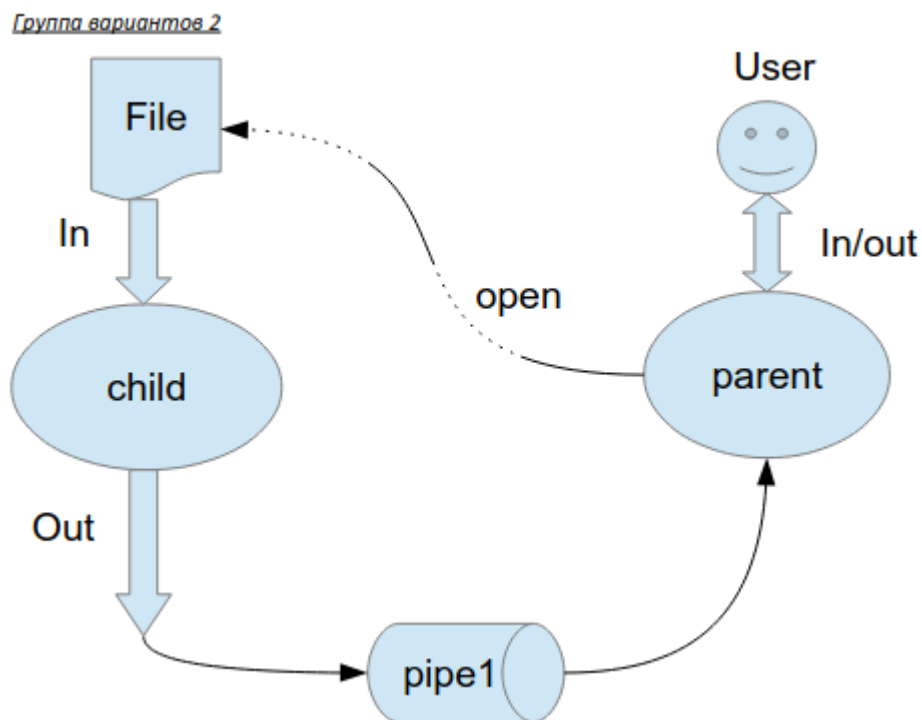
<https://github.com/YMusaelyan/os>

Постановка задачи

Цель работы

Приобретение практических навыков в управлении процессами в ОС, обеспечение обмена данными между процессами посредством каналов.

Задание



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.

10 вариант) В файле записаны команды вида: «число<endline>». 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 файлов main.cpp, child.cpp. В программе используются следующие системные вызовы:

1. pipe() - для передачи информации между процессами.
2. fork() - создает процесс
3. execlp() - передает процесс другой программе
4. read() - читает данные из файла.
5. dup2() - создает копию файлового дескриптора
6. close() - закрывает файловый дескриптор
7. exit() - завершает процесс

Общий метод и алгоритм решения

Пользователь пишет название файла. Открываем заданный файл на чтение. Создаем пайп и дочерний процесс. Обрабатываем ошибки при создании. Переопределяем поток ввода на файл и поток вывода на пайп. Создадим функцию на проверку на числа. Если число составное, то функция выдает значение 1, если отрицательное или простое (для однозначности добавим к этим числам 0 и 1) — значение -1. Если число составное, то записываем результат, разделяя построчно, в остальных случаях передаем знак конца файла и завершаем дочерний процесс. В родительском процессе считываем из пайпа, если не знак конца файла, то выводим результат, иначе завершаем родительский процесс.

Исходный код

main.cpp

```
#include <unistd.h>
#include <iostream>
#include <string>
#include <fcntl.h>
using namespace std;

int main(){

    string file_name;
    cin >> file_name;

    int file = open(file_name.c_str(), O_RDONLY);
    if (file == -1) {
        cerr << "error file\n";
        return 0;
    }

    int pipefd[2];
    if (pipe(pipefd) == -1){
        cerr << "error pipe\n";
        return 0;
    }
```

```

    }

    pid_t id = fork();
    if (id == -1){
        cerr << "error fork";
        return 0;
    } else if (id == 0){
        execlp("./child", to_string(file).c_str(), to_string(pipefd[0]).c_str(),
to_string(pipefd[1]).c_str(), NULL);
    } else {
        char p;
        while (true){
            read(pipefd[0], &p, sizeof(p));
            if (p == '\0') {
                exit(0);
            } else {
                putchar(p);
            }
        }
    }

    close(pipefd[0]);
    close(pipefd[1]);
    close(file);

    return 0;
}

```

child.cpp

```

#include <unistd.h>
#include <sstream>
#include <iostream>

using namespace std;

int func(int number) {
    int composite = 0;

    if (number < 2) {
        composite = -1;
    } else {
        for (int i = 2; i * i <= number; i++) {
            if (number % i == 0) {
                composite = 1;
                break;
            }
        }
        if (composite != 1) {
            composite = -1;
        }
    }
    return composite;
}

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

    int pipefd[2];

```

```

pipefd[0] = atoi(argv[1]);
pipefd[1] = atoi(argv[2]);

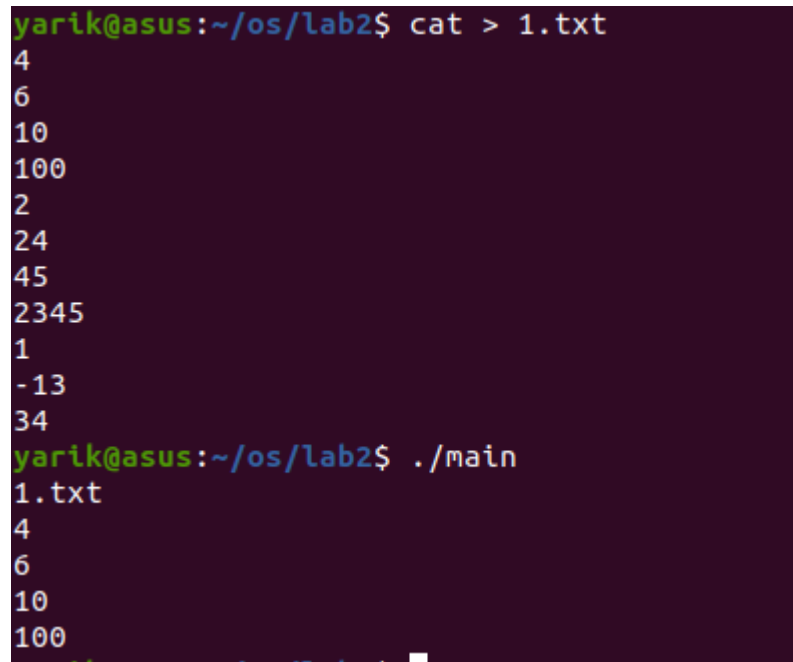
int file = stoi(argv[0]);

if (dup2(file, 0) == -1) {
    cerr << "error dub\n";
    return 0;
}
if (dup2(pipefd[1], 1) == -1) {
    cerr << "error dub\n";
    return 0;
}

int n;
while (cin >> n) {
    if (func(n) == 1) {
        cout << n << "\n";
    } else {
        cout << '\0';
        exit(0);
    }
}
close(pipefd[0]);
close(pipefd[1]);
}

```

Демонстрация работы программы



```

yarik@asus:~/os/lab2$ cat > 1.txt
4
6
10
100
2
24
45
2345
1
-13
34
yarik@asus:~/os/lab2$ ./main
1.txt
4
6
10
100

```

Выводы

Данная лабораторная работа была очень полезной. Я познакомился с процессами и с управлением процессов в ОС, приобретение практические навыки в управлении процессами в ОС, научился обеспечивать обмен данных между процессами посредством каналов.