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

“Изучение взаимодействий между процессами”

Студент: Молчанов Владислав Дмитриевич
Группа: М8О-208Б-20
Вариант: 10
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2021

Содержание

Репозиторий

github.com/molch4nov/OS/

Постановка задачи

Программа должна осуществлять работу с процессами и взаимодействие между ними в операционной системе Linux. 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

parent.cpp – работа родительского процесса

child.cpp – работа дочернего процесса

В своей программе я использую библиотеку fstream для работы с файлом, библиотеку unistd.h для работы с процессами и системными вызовами.

Основные команды в моей программе:

int fd[2] - создание массива из 2 дескрипторов, 0 - чтение (read), 1 - передача (write)

pipe(fd) - создание неименованного канала для передачи данных между процессами, т.е. конвейер, с помощью которого выход одной команды подается на вход другой

int id = fork () - создание дочернего процесса, в переменной id будет лежать “специальный код” процесса (-1 - ошибка fork, 0 - дочерний процесс, >0 - родительский). Чтобы посмотреть, какую ошибку выводит fork, можем воспользоваться errno.

int execve(const char *filename, char *const argv[], char *const envp[]) (и другие вариации exec) - замена образа памяти процесса

int dup2(int oldfd, int newfd) - переназначение файлового дескриптора

int putchar(char c) - функция записывает символ в текущей позиции в стандартный поток вывода (stdout) и перемещает внутренний указатель положения файла в следующую позицию.
close(...) – закрытие файла, файловых дескрипторов

Сборка проекта происходит при помощи make-файла
g++ child.cpp –o child
g++ parent.cpp

Общий метод и алгоритм решения
Имеется два процесса: родительский и дочерний. Родительский процесс считывает названия файла из стандартного ввода, из которого дочерний процесс осуществляет чтение. Так же родительский процесс выводит на экран все, что ему придет через pipe (взаимодействие между процессами осуществляется с помощью него). Производится перенаправление потока ввода на файл, потока вывода на pipe, происходит деление чисел, введенных в файл и выводится в стандартный поток вывода. Работа между процессами обеспечивается с помощью таких команд, как dup2, close.

Исходный код

Parent.cpp

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

int main() {
    string s;
    cout << "Enter a filename with tests:";     //ввод название и перевод
к массиву чаров
    cin >> s;
    const char * buf = s.c_str();

    if(access(buf, R_OK) == -1){//проверка имеет ли вызвавший процесс
права доступа к buf на запись
        cout << "File not exist" << endl;
        exit(EXIT_FAILURE);
    }

    int fd[2];
    pipe(fd);   //создание канала чтения и записи


    pid_t pid = fork();
    if(pid == -1){
        perror("fork");
        return -1;
    }
    else if(pid == 0){
        int file = open(buf, O_RDONLY);
        if (file == -1) {
            perror("open");
            exit(EXIT_FAILURE);
        }
        if (dup2(fd[1], 1) == -1) {
            exit(1);
        }
        if (dup2(file, 0) == -1) {
            exit(1);
        }
```

```
        close(fd[0]);
        close(fd[1]);
        close(file);
        execl("child", "child", NULL);
    }
    close(fd[1]);
    char c;
    while(read(fd[0], &c, sizeof(char)) > 0){
        putchar(c);
    }
    close(fd[0]);
    return 0;

}
```

Child.cpp

```
#include <stdio.h>
#include <unistd.h>

using namespace std;

int main(){
    int n; char c;
    while(scanf("%d%c", &n, &c) != EOF){
        if(n < 0){
            break;
        }
        for (int i = 2; i * i <= n; i++) {
            if (n % i == 0) {
                printf("%d\n ", n);
                break;
            }
            else {
                break;
            }
        }
    }
    return 0;
}
```

Демонстрация работы программы

vladislav@DESKTOP-OL36FK8:/mnt/c/Users/vlad-/Desktop/os_lab2/src$ ./main

Enter a filename with tests:test.txt.txt

10

vladislav@DESKTOP-OL36FK8:/mnt/c/Users/vlad-/Desktop/os_lab2/src$ ./main

Enter a filename with tests:test.txt.txt

vladislav@DESKTOP-OL36FK8:/mnt/c/Users/vlad-/Desktop/os_lab2/src$

Выводы

Эта лабораторная работа ознакомила и научила меня работать с межпроцессорным взаимодействием между, с переопределениями потоков ввода/вывода. Также я научился работать с системными вызовами и файловыми дескрипторами.