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: Литовченко Анна
Группа: М8О-207Б-21
Вариант: 5
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/Annalitov/OS

Постановка задачи

Цель работы

Приобретение практических навыков в:

- Управление процессами в ОС
- Обеспечение обмена данных между процессами посредством каналов

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 каналы (pipe).

Необходимо обрабатывать системные ошибки, которые могут возникнуть в результате работы.

Группа вариантов 1

Родительский процесс создает дочерний процесс. Первой строчкой пользователь в консоль родительского процесса пишет имя файла, которое будет передано при создании дочернего процесса. Родительский и дочерний процесс должны быть представлены разными программами. Родительский процесс передает команды пользователя через pipe1, который связан с стандартным входным потоком дочернего процесса. Дочерний процесс при необходимости передает данные в родительский процесс через pipe2. Результаты своей работы дочерний процесс пишет в созданный им файл. Допускается просто открыть файл и писать туда, не перенаправляя стандартный поток вывода.

5 вариант) 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

Программа компилируется из файла main.cpp. Дочерний процесс представлен программой child.cpp.

В программе используются следующие системные вызовы:

1. pipe() - создаёт однонаправленный канал данных, который можно использовать для взаимодействия между процессами.
2. fork() - создает копию текущего процесса, который является дочерним процессом для текущего процесса
3. execl() - загружает и запускает другие программы, известные как "дочерние" процессы.

4. close() - закрывает файл
5. read() - читает количество байт(третий аргумент) из файла с файловым дескриптором(первый аргумент) в область памяти(второй агрумент).
6. write() - записывает в файл с файловым дескриптором(первый аргумент) из области памяти(второй аргумент) количество байт(третий аргумент).
7. perror() – вывод сообщения об ошибке.

Общий метод и алгоритм решения

С помощью вызова fork() создаются родительский и дочерний процессы. Родительский процесс считывает название будущего файла и число, затем передает всё в дочерний процесс, который создает файл и выполняет проверку данного числа. Если число удовлетворяет условиям, то он записывает его в файл, в противном случае передает родительскому процессу информацию о том, что число не подошло.

Исходный код

main.cpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <sys/types.h>
#include <unistd.h>
#include <fstream>
#include <errno.h>
#include <signal.h>
#include <sys/wait.h>


using namespace std;

int main(){

                                        fstream f;

                                        string filename;
                                        cout<<"Enter a filename: "<<endl;

                                        cin >> filename;
```

```
int fd_1[2];
int fd_2[2];
int pipe_1[2];
int pipe_2[2];

if (pipe(pipe_1) == -1){
        perror("pipe");
        exit(EXIT_FAILURE);
}

if (pipe(pipe_2) == -1){
        perror("pipe");
        exit(EXIT_FAILURE);
}


string num;

pid_t id = fork();

if (id == -1){

        perror("fork");
        exit(EXIT_FAILURE);
} else if (id == 0) {

        fd_1[0] = pipe_1[0];
        fd_1[1] = pipe_1[1];
        fd_2[0] = pipe_2[0];
        fd_2[1] = pipe_2[1];
        execl("./child", to_string(fd_1[0]).c_str(),
to_string(fd_1[1]).c_str(), to_string(fd_2[0]).c_str(), to_string(fd_2[1]).c_str(), filename.c_str(), NULL);

} else {

        cout << "Enter a number: " << endl;
        cin >> num;
        int s_size = num.size();
        char str_array[s_size];
        for (int k = 0; k < s_size; ++k) {
                str_array[k] = num[k];
        }
        write(pipe_1[1], &s_size, sizeof(int));
```

```
                write(pipe_1[1], str_array, sizeof(char)*s_size);
                int flag_0;
                read(pipe_2[0], &flag_0, sizeof(int));
                if (flag_0 == 1) {
                        cout << "The number is prime or negative" << endl;
                }
        }

        close(pipe_1[0]);
        close(pipe_1[1]);
        close(pipe_2[0]);
        close(pipe_2[1]);

        return 0;

}
```

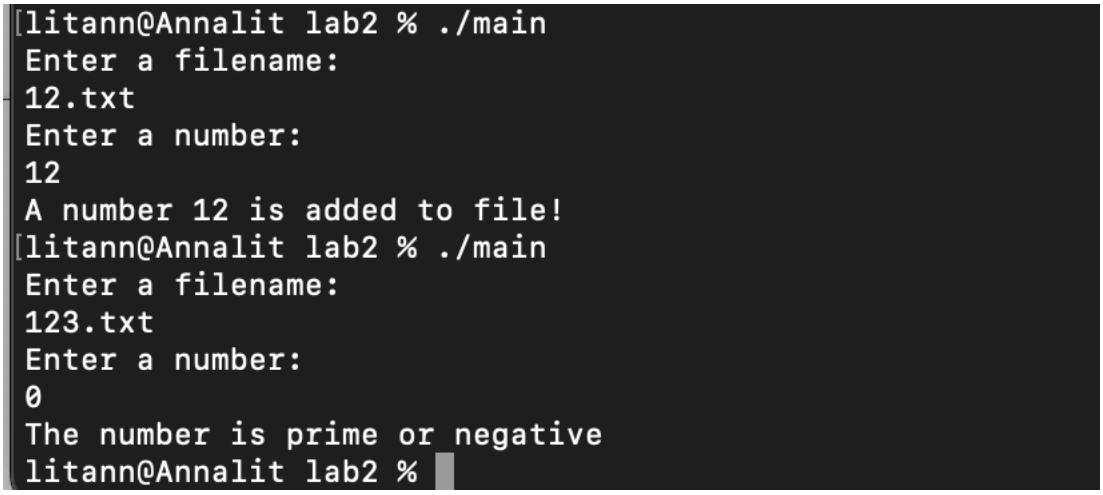
child.cpp

```
#include <iostream>
#include <string>
#include <cstdlib>
#include <sys/types.h>
#include <unistd.h>
#include <fstream>
#include <errno.h>
#include <signal.h>
#include <sys/wait.h>

using namespace std;

int isNotPrime(int n)
{
   if (n < 2) {
      return 0;
   } else {
      for (int i = 2; i * i < n + 1; i++) {
         if (n % i == 0) {
            return 1;
         }
      }
   }
   return 0;
}
```

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

                                        string filename = argv[4];
                                        int fd_1[2];
                                        int fd_2[2];
                                        int flag_1 = 1;
                                        int flag_2 = 2;

                                        fd_1[0] = stoi(argv[0]);
                                        fd_1[1] = stoi(argv[1]);

                                        fd_2[0] = stoi(argv[2]);
                                        fd_2[1] = stoi(argv[3]);


                                        fstream f;
                                        f.open(filename, fstream::in | fstream::out | fstream::app);

                                        while(true) {
                                                int num_size;

                                                read(fd_1[0], &num_size, sizeof(int));
                                                char num_str[num_size];
                                                read(fd_1[0], &num_str, sizeof(char)*num_size);

                                                string result;
                                                for (int i = 0; i < num_size; i++) {
                                                        result.push_back(num_str[i]);
                                                }

                                                int number;
                                                int number_1;
                                                number =  stoi(result);
                                                number_1 = abs(number);
                                                if ( number > 0 && isNotPrime(number_1) > 0 ) {
                                                        f << result << endl;
                                                        cout << "A number " << result << " is added to file!"
<< endl;

                                                        write(fd_2[1], &flag_2, sizeof(int));

                                                } else {

                                                        write(fd_2[1], &flag_1, sizeof(int));
```

```
        }

    }

    return 0
}
```

Демонстрация работы программы



Выводы

Проделав данную лабораторную работу я приобрела практические навыки в управлении процессами в операционной системе и смогла произвести обмен данными между ними, использую каналы.