Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №1 по курсу

«Операционные системы»

Студент: Бойцов Иван Алексеевич
Группа: М8О–212Б–22
Вариант: 5
Преподаватель: Миронов Евгений Сергеевич
Оценка: _____
Дата: _____
Подпись: _____

Москва, 2024

Цель работы

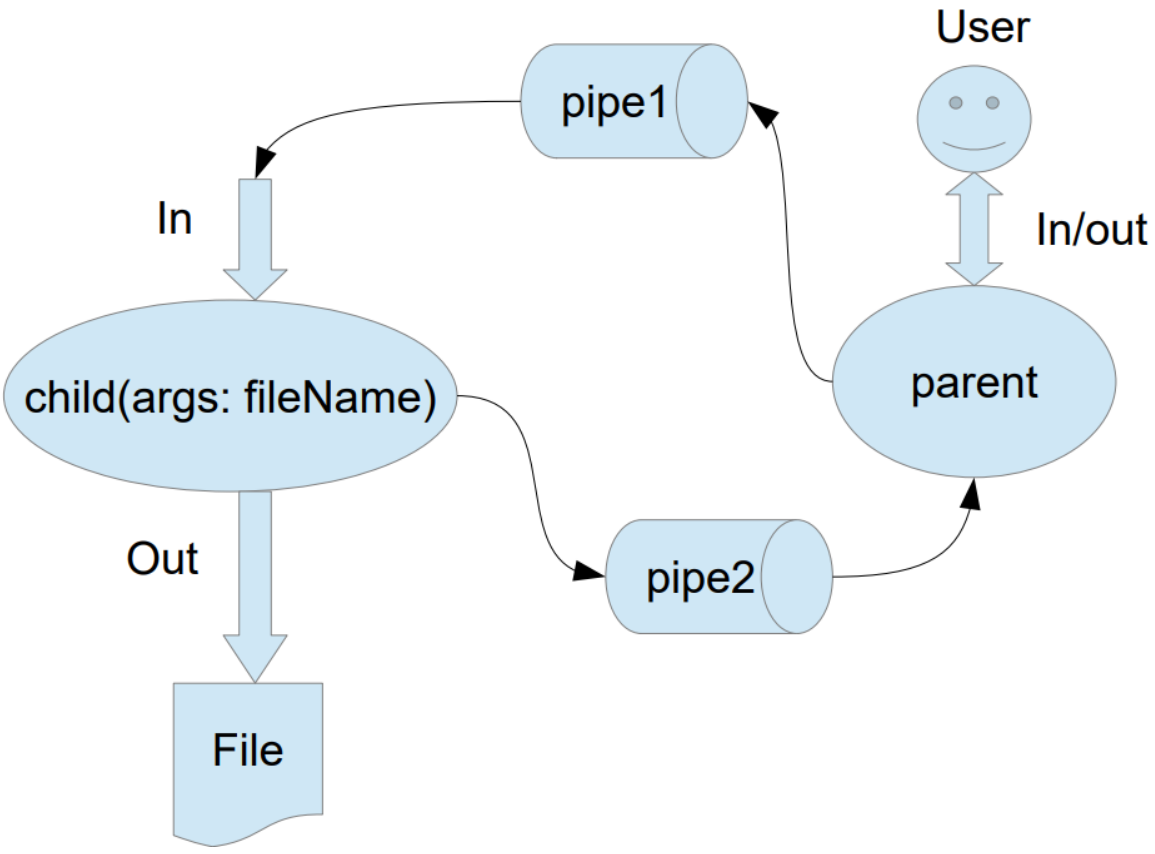
Приобретение практических навыков в:

1. Управление процессами в ОС
2. 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

принеобходимости 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.

Вариант 5

Пользователь вводит команды вида: «число». 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.

Решение

Программа создаёт два пайпа, которые будут связывать родительский процесс с дочерним, по первому пайпу числа из родительского процесса идут в дочерний, в котором они проходят "фильтрацию", от которой зависит останется ли число в файле или завершится программа. После этого сообщенние о завершнении проходит по второму пайпу и запись чисел от пользователя останавливается.

parent.cpp

```
#include <windows.h>
#include <iostream>
#include <string>

using namespace std;

wstring stringToWString(const string& str) {
    int len;
    int slength = (int)str.length() + 1;
    len = MultiByteToWideChar(CP_ACP, 0, str.c_str(), slength, 0, 0);
    wstring r(len, L'\0');
    MultiByteToWideChar(CP_ACP, 0, str.c_str(), slength, &r[0], len);
    return r;
}

int main() {
    HANDLE hPipe1Read, hPipe1Write, hPipe2Read, hPipe2Write;
    SECURITY_ATTRIBUTES sa = { sizeof(SECURITY_ATTRIBUTES), NULL,
TRUE };

    // Создание pipe для связи с дочерним процессом
```

```
    if (!CreatePipe(&hPipe1Read, &hPipe1Write, &sa, 0)) {
        cerr << "Failed to create pipe1" << endl;
        return 1;
    }
    if (!CreatePipe(&hPipe2Read, &hPipe2Write, &sa, 0)) {
        cerr << "Failed to create pipe2" << endl;
        return 1;
    }

    // Получение имени файла от пользователя
    string fileName;
    cout << "Enter the output file name: ";
    getline(cin, fileName);

    // Создание строки запуска дочернего процесса
    std::string commandLine =
"C:\\Users\\ivanb\\source\\repos\\OS_lab1_child\\x64\\Debug\\OS_lab1_child.exe
" + fileName;
    std::wstring wCommandLine = stringToWString(commandLine);

    STARTUPINFO si = { sizeof(STARTUPINFO) };
    PROCESS_INFORMATION pi;

    si.hStdInput = hPipe1Read;
    si.hStdOutput = hPipe2Write;
    si.hStdError = GetStdHandle(STD_ERROR_HANDLE);
    si.dwFlags |= STARTF_USESTDHANDLES;

    // Создание дочернего процесса
    if (!CreateProcess(NULL, &wCommandLine[0], NULL, NULL, TRUE, 0,
NULL, NULL, &si, &pi)) {
        cerr << "Failed to create process" << endl;
        return 1;
    }

    // Закрытие ненужных дескрипторов
    CloseHandle(pi.hProcess);
    CloseHandle(pi.hThread);
    CloseHandle(hPipe1Read);
    CloseHandle(hPipe2Write);

    // Ввод и отправка данных
    string input;
    while (true) {
```

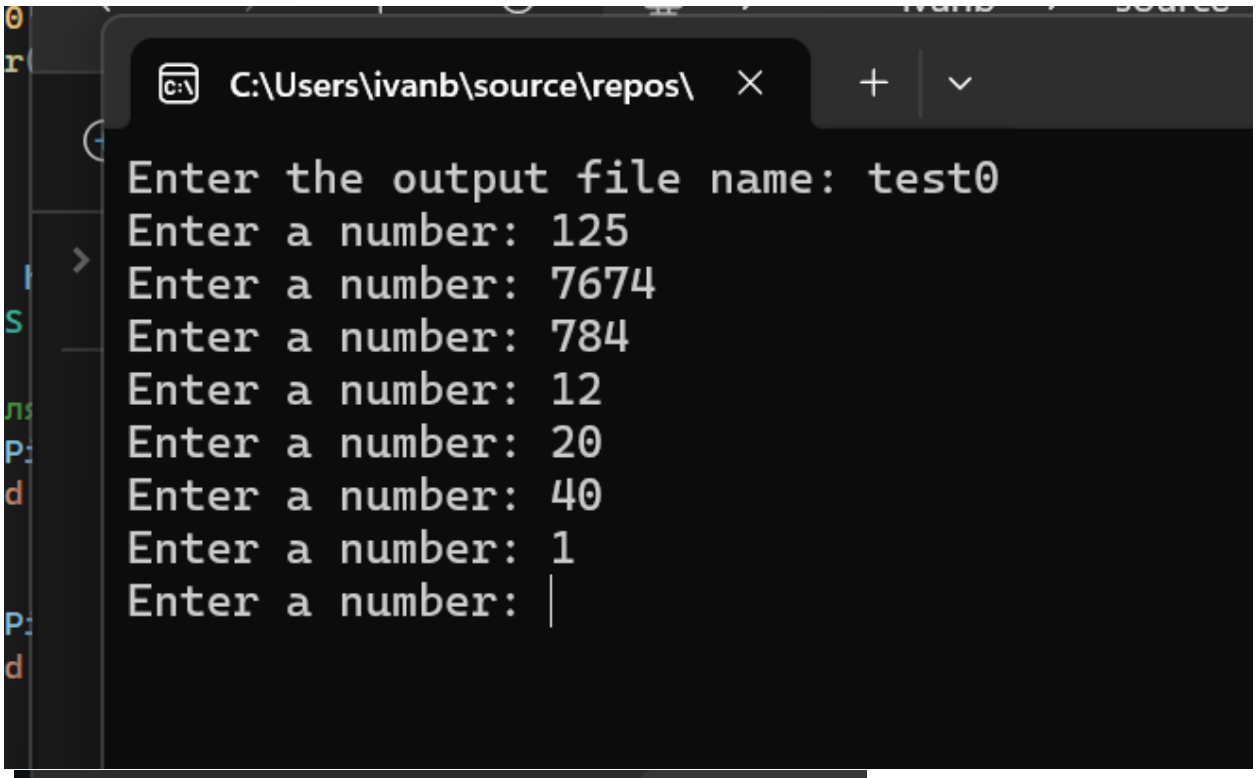
```
        cout << "Enter a number: ";
        getline(cin, input);

        if (input.empty()) break;

        DWORD written;
        if (!WriteFile(hPipe1Write, input.c_str(), input.size(), &written, NULL)) {
            cerr << "Failed to write to pipe" << endl;
            break;
        }

        // Если число отрицательное или простое, завершаем процесс
        int num = stoi(input);
        if (num < 0) break;
    }

    // Закрытие дескрипторов
    CloseHandle(hPipe1Write);
    CloseHandle(hPipe2Read);

    return 0;
}
```

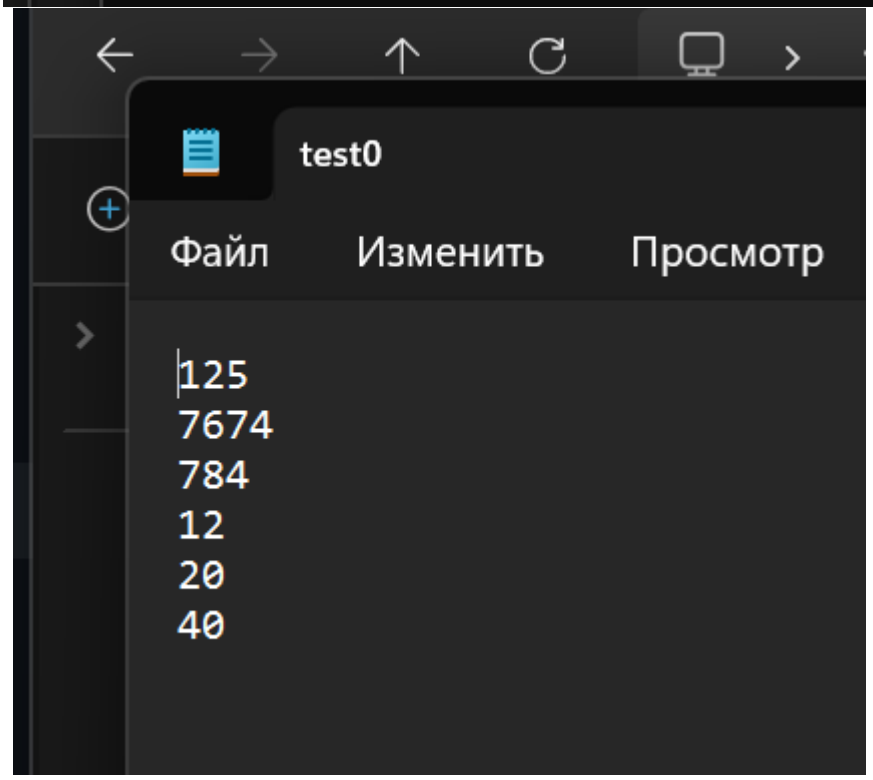
child.cpp

```
#include <windows.h>
#include <iostream>
#include <fstream>
#include <string>

bool isComposite(int num) {
    if (num <= 1) return false;
    for (int i = 2; i * i <= num; ++i) {
        if (num % i == 0) return true;
    }
    return false;
}

int main(int argc, char* argv[]) {
    if (argc < 2) {
        std::cerr << "No file name provided!" << std::endl;
        return 1;
    }
```

```
    std::ofstream outFile(argv[1], std::ios::app);
    if (!outFile.is_open()) {
        std::cerr << "Failed to open file: " << argv[1] << std::endl;
        return 1;
    }

    HANDLE hPipe1Read = GetStdHandle(STD_INPUT_HANDLE);
    char buffer[128];
    DWORD bytesRead;

    while (true) {
        if (!ReadFile(hPipe1Read, buffer, sizeof(buffer) - 1, &bytesRead, NULL) ||
bytesRead == 0) {
            break;
        }
        buffer[bytesRead] = '\0';

        int number = std::stoi(buffer);
        if (number < 0 || !isComposite(number)) {
            break;
        }

        outFile << number << std::endl;
    }

    outFile.close();
    return 0;
}
```

Доказательство работы





Вывод

В результате выполнения данной лабораторной работы была написана программа на языке C++, осуществляющая работу с процессами и взаимодействие между ними. Были приобретены практические навыки в управлении процессами в Windows и обеспечении обмена данных между процессами посредством каналов.