**Московский авиационный институт**

**(Национальный исследовательский университет)**

Институт: «Информационные технологии и прикладная математика»

Кафедра: 806 «Вычислительная математика и программирование»

Дисциплина: «Операционные системы»

**Лабораторная работа № 2**

Тема: Управление процессами в ОС

Студент: Калугин К. А.

Группа: М8О-207Б-19

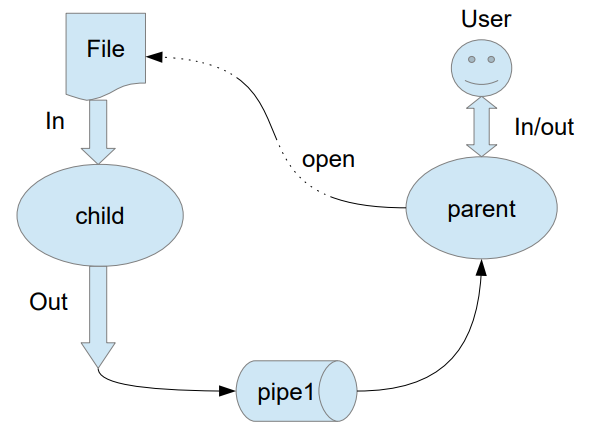
Преподаватель: Миронов Е.С.

Дата:

Оценка:

Москва, 2021

**Постановка задачи:** 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.



7 вариант) В файле записаны команды вида: «число число число». Дочерний процесс считает их сумму и выводит результат в стандартный поток вывода. Числа имеют тип float.

**Алгоритм решения задачи.**

Вначале я, при помощи другой программы, подготавливаю файл, в который записаны числа. Далее, уже в основной программе, я подготавливаю этот файл для передачи в дочерний процесс через pipe. После чего создаю процесс, передаю данные и дожидаюсь возврата результата. Дочерний процесс в это время проводит сложение чисел и возвращает код ошибки (0 – в случае успеха). После возврата кода я вывожу либо ответ, либо сообщение об ошибке.

**Листинг программы**

**1.с**

#include <stdio.h>

#include <windows.h>

#include <string.h>

#include <locale.h>

HANDLE parentinput = NULL;

HANDLE childoutput = NULL;

HANDLE file;

HANDLE errparentinput = NULL;

HANDLE errchildoutput = NULL;

SECURITY\_ATTRIBUTES sa;

PROCESS\_INFORMATION pi;

STARTUPINFO si;

DWORD D;

DWORD err = 0;

float x;

int error = 0;

int main ()

{

    HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

    sa.nLength = sizeof (SECURITY\_ATTRIBUTES);

    sa.bInheritHandle = TRUE;

    ZeroMemory (&si, sizeof(si));

    si.cb = sizeof (si);

    ZeroMemory (&pi, sizeof(pi));

    char name[] = "working";

    scanf("%s", name);

    file = CreateFileA (//открытие файла на чтение

        name,

        GENERIC\_READ,

        0,

        &sa,

        OPEN\_EXISTING,

        FILE\_ATTRIBUTE\_NORMAL,

        NULL

    );

    if ((err = GetLastError ()) != 0)

        {

            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

            printf ("ERROR\nIncorrect work of function \"CreateFileA\". Code of error: %d.", err);

            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

            return -1;

        }

    if (!CreatePipe (&parentinput, &childoutput, &sa, 0))

        {

            if ((err = GetLastError ()) != 0)

                {

                    SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

                    printf ("ERROR\nIncorrect work of function \"CreatePipe\".");

                    SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

                    return -1;

                }

            return -1;

        }

    if (!SetHandleInformation(parentinput, HANDLE\_FLAG\_INHERIT, 0))

        {

            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

            printf ("ERROR\nIncorrect work of function \"SetHandleInformation\".");

            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

            return -1;

        }

    if (!CreatePipe (&errparentinput, &errchildoutput, &sa, 0))

    {

        if ((err = GetLastError ()) != 0)

            {

                SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

                printf ("ERROR\nIncorrect work of function \"CreatePipe\".");

                SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

                return -1;

            }

        return -1;

    }

    if (!SetHandleInformation(errparentinput, HANDLE\_FLAG\_INHERIT, 0))

        {

            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

            printf ("ERROR\nIncorrect work of function \"SetHandleInformation\".");

            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

            return -1;

        }

    si.hStdError = errchildoutput;

    si.hStdOutput = childoutput;

    si.hStdInput = file;

    si.dwFlags |= STARTF\_USESTDHANDLES;

    if (!CreateProcess (NULL,

        "2.exe",

        NULL,

        NULL,

        TRUE,

        0,

        NULL,

        NULL,

        &si,

        &pi)

    )

        {

            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

            printf ("ERROR\nIncorrect work of function \"CreateProcess\".");

            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

            return -1;

        }

    else

        {

            if(!ReadFile (errparentinput, &error, sizeof(int), &D, NULL))

                {

                    err = GetLastError ();

                    SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

                    printf ("ERROR\nIncorrect work of function \"ReadFile\". Code of error: %d.", err);

                    SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

                    return -1;

                }

            else

                {

                    if (error == 0)

                        {

                            if(!ReadFile (parentinput, &x, sizeof(float), &D, NULL))

                                {

                                    err = GetLastError ();

                                    SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

                                    printf ("ERROR\nIncorrect work of function \"ReadFile\". Code of error: %d.", err);

                                    SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

                                    return -1;

                                }

                            else

                                {

                                    printf ("%f", x);

                                }

                        }

                    else if (error == 1)

                        {

                            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

                            printf ("ERROR\nInput file is incorrect");

                            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

                        }

                    else if (error == 2)

                        {

                            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

                            printf ("ERROR\nIncorrect work of child process");

                            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

                        }

                    else if (error == 3)

                        {

                            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 4));

                            printf ("ERROR\nFile is empty");

                            SetConsoleTextAttribute(hConsole, (WORD) ((0 << 4) | 7));

                        }

                }

        }

    CloseHandle (file);

    CloseHandle (parentinput);

    CloseHandle (childoutput);

    CloseHandle (errparentinput);

    CloseHandle (errchildoutput);

    CloseHandle (pi.hProcess);

    CloseHandle (pi.hThread);

    CloseHandle (hConsole);

    return 0;

}

**2.с**

#include <stdio.h>

#include <windows.h>

int main()

{

    HANDLE in = GetStdHandle(STD\_INPUT\_HANDLE);

    HANDLE out = GetStdHandle(STD\_OUTPUT\_HANDLE);

    HANDLE err = GetStdHandle(STD\_ERROR\_HANDLE);

    DWORD D;

    float x = 0, y = 0;

    int error = 0;

    char flag = '@';

    int errflag = 0;

    while (1)

        {

            if(!ReadFile (in, &y, sizeof(float), &D, NULL))

                {

                    error = 2;

                    WriteFile (err, &error, sizeof(int), &D, NULL);

                    return -1;

                }

            flag = '!';

            if(!ReadFile (in, &flag, sizeof(char), &D, NULL))

                {

                    error = 2;

                    WriteFile (err, &error, sizeof(int), &D, NULL);

                    return -1;

                }

            if (flag == '!')

                {

                    break;

                }

            else if (flag != ' ')

                {

                    error = 1;

                    if(!WriteFile (err, &error, sizeof(int), &D, NULL))

                        {

                            error = 2;

                            WriteFile (err, &error, sizeof(int), &D, NULL);

                            return -1;

                        }

                    return -1;

                }

            x += y;

            errflag ++;

        }

    if (errflag == 0)

        {

            error = 3;

            if(!WriteFile (err, &error, sizeof(int), &D, NULL))

                {

                    error = 2;

                    WriteFile (err, &error, sizeof(int), &D, NULL);

                    return -1;

                }

            return -1;

        }

    if(!WriteFile (err, &error, sizeof(int), &D, NULL))

        {

            error = 2;

            WriteFile (err, &error, sizeof(int), &D, NULL);

            return -1;

        }

    if(WriteFile (out, &x, sizeof(float), &D, NULL))

        {

            error = 2;

            WriteFile (err, &error, sizeof(int), &D, NULL);

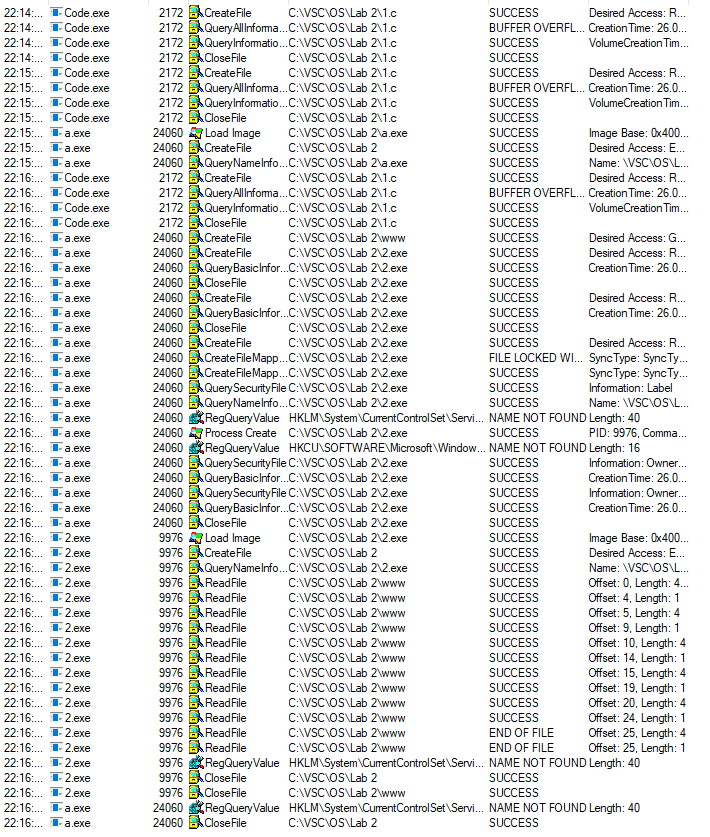
            return -1;

        }

    return 0;

}

**ProcMon**



**Выводы**

Я изучил работу с процессами и pipe`ами в Windows. Основная проблема, с которой я столкнулся – это малое количество документации, особенно русскоязычной, что вместе с общей запутанностью команд (создание процесса требует 10 различных параметров) создает определенные трудности.

**Список литературы**

1. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. —СПб.: Издательский дом «Питер», 2018. — С. 111 - 123