МИНОБРНАУКИ РОССИИ

Федеральное государственное автономное образовательное  
 учреждение высшего образования   
«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт компьютерных технологий и информационной безопасности

Кафедра математического обеспечения и применения ЭВМ

**Отчет**

По лабораторной работе №5

на тему: **«Обмен данными между процессами в Windows»**

по курсу

**«Операционные системы и системное программное обеспечение»**

Вариант №5

Выполнил:

студент гр. КТбо2-8

Жалнин Д.И.

Проверил:

Ассистент каф. МОП ЭВМ

Альминене Т. А.

# 1 Цель работы

Изучение механизмов межпроцессного обмена данными в Windows и приобретении навыков использования различных средств обмена данными при разработке программ.

2 Задание

Необходимо разработать программу, реализующую следующий алгоритм, используя Windows API:

Родительский процесс A порождает два процесса-потомка и создает безымянный канал. Далее в цикле процесс A просит пользователя ввести число и указать, какому из потомков (по номеру) следует это число послать. Каждый потомок вычисляет и выводит сумму полученных чисел. Процессы завершаются, когда пользователь вводит номер 0.

3 Ход работы

Для написания программы был выбран язык программирования С++. По ходу выполнения лабораторной работы был составлен следующий код:

**Код программы «родительского» процесса:**

#include <windows.h>

#include <iostream>

#include <io.h>

#include <stdio.h>

#include <fcntl.h>

STARTUPINFO startupInfoA;

STARTUPINFO startupInfoB;

PROCESS\_INFORMATION procInfoA;

PROCESS\_INFORMATION procInfoB;

TCHAR cLine[] = TEXT("Proc");

HANDLE hWritePipeA;

HANDLE hWritePipeB;

HANDLE hReadPipeA;

HANDLE hReadPipeB;

void closeHandles() {

TerminateProcess(procInfoA.hProcess, 9);

TerminateProcess(procInfoB.hProcess, 9);

CloseHandle(procInfoA.hProcess);

CloseHandle(procInfoA.hThread);

CloseHandle(procInfoB.hProcess);

CloseHandle(procInfoB.hThread);

}

int createProcess(TCHAR\* cl, STARTUPINFO\* sInfo, PROCESS\_INFORMATION\* pInfo) {

return CreateProcess(

NULL,

cl,

NULL,

NULL,

TRUE,

CREATE\_NEW\_CONSOLE,

NULL,

NULL,

sInfo,

pInfo

);

}

int main() {

setlocale(0, "Russian");

SECURITY\_ATTRIBUTES saAttrA = {

sizeof(SECURITY\_ATTRIBUTES),

NULL,

TRUE

};

SECURITY\_ATTRIBUTES saAttrB = {

sizeof(SECURITY\_ATTRIBUTES),

NULL,

TRUE

};

if (

CreatePipe(&hReadPipeA, &hWritePipeA, &saAttrA, 0) &&

CreatePipe(&hReadPipeB, &hWritePipeB, &saAttrB, 0)

) {

std::cout << "PIPES - OK\n";

}

else {

std::cout << "PIPES - ERROR\n";

return 1;

}

ZeroMemory(&procInfoA, sizeof(PROCESS\_INFORMATION));

ZeroMemory(&startupInfoA, sizeof(STARTUPINFO));

startupInfoA.cb = sizeof(STARTUPINFO);

ZeroMemory(&procInfoB, sizeof(PROCESS\_INFORMATION));

ZeroMemory(&startupInfoB, sizeof(STARTUPINFO));

startupInfoB.cb = sizeof(STARTUPINFO);

//переопределяем стандартный ввод в процессах-потомках

startupInfoA.dwFlags = STARTF\_USESTDHANDLES;

startupInfoB.dwFlags = STARTF\_USESTDHANDLES;

startupInfoA.hStdInput = hReadPipeA;

startupInfoB.hStdInput = hReadPipeB;

if (createProcess(cLine, &startupInfoA, &procInfoA) && createProcess(cLine, &startupInfoB, &procInfoB)) {

std::cout << "PROCESSES - OK\n";

}

else {

std::cout << "PROCESSES - ERROR\n";

closeHandles();

return 1;

}

CloseHandle(hReadPipeA);

CloseHandle(hReadPipeB);

int x;

char procLit;

FILE\* pipeA = \_fdopen((HFILE)\_open\_osfhandle((intptr\_t)hWritePipeA, \_O\_TEXT),"a+t");

FILE\* pipeB = \_fdopen((HFILE)\_open\_osfhandle((intptr\_t)hWritePipeB, \_O\_TEXT),"a+t");

while (true) {

std::cout << "Write a number -> ";

std::cin >> x;

if (x == 0) {

break;

}

std::cout << "Write the name of proccess (A/B) -> ";

std::cin >> procLit;

if (procLit == 'A' || procLit == 'a') {

fprintf(pipeA, "%d\n", x);

fflush(pipeA);

}

else if (procLit == 'B' || procLit == 'b') {

fprintf(pipeB, "%d\n", x);

fflush(pipeB);

}

else {

break;

}

}

fclose(pipeA);

fclose(pipeB);

closeHandles();

return 0;

}

**Код программы дочернего процесса:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <iostream>

#include <stdlib.h>

#include <windows.h>

#include <stdio.h>

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

int sum = 0;

int x = 0;

while (true) {

std::cin >> x;

printf("%s %d\n","The sum is",sum += x);

}

}

Исходный код был написан и протестирован в среде разработки Visual Studio 2019. Программа решает поставленную задачу и отрабатывает без ошибок.

4 Пояснения

Решение поставленного задания состоит из двух частей, которые располагаются в разных .cpp файлах, и соответственно компилируются в две различные программы.

В «родительском» исходном коде расположена логика создания двух «дочерних» процессов и замены в них стандартного входного потока данных. Сперва для процессов-потомков создаются все необходимые для операционной системы структуры:

1. Структуры с информацией, необходимой для запуска дочерних процессов:

STARTUPINFO startupInfoA;

STARTUPINFO startupInfoB;

1. Структуры, содержащие в себе информацию о процессах, их хэндлы и переопределенные потоки входных и выходных данных (если это необходимо):

PROCESS\_INFORMATION procInfoA;

PROCESS\_INFORMATION procInfoB;

1. Массив с командами, которые необходимо выполнить в командной строке для запуска дочернего процесса:

TCHAR cLine[] = TEXT("Proc");

1. Хэндлеры входных и выходных структур «pipeline», через которые дочерние процессы будут взаимодействовать с родительским:

HANDLE hWritePipeA;

HANDLE hWritePipeB;

HANDLE hReadPipeA;

HANDLE hReadPipeB;

Описание функций в исходном коде программ:

* closeHandles – закрывает все хэндлы процессов ии их потоков, которые использовались при работе программы
* int createProcess – запускает дочерний процесс и записывает его хэндл в передаваемую структуру типа HANDLE.
* int main:
* Заполняет информацию о дочерних потоках,
* Вызывает функцию для их запуска,
* Инициализирует анонимные «pipeline» для обмена сообщениями между процессами,
* Переопределяет стандартные потоки дочерних процессов, заменяя их на «pipeline»
* Циклически общается с пользователем, через интерфейс, который считывает информацию, для передачи в другой процесс (число), и литеру запущенного процесса (А или В). При вводе значения «0» основная программа и дочерние процессы заканчивают работу.

В коде дочерней программы содержится только зацикленный ввод одного целого числа и вывод суммы всех введенных из переопределенного потока ввода чисел.

5 Выводы

В ходе выполнения лабораторной работы я закрепил навыки работы со средствами управления потоками и процессами, а так познакомился с одним из возможных способов передачи сообщений (безымянный «pipeline») между процессами в операционной системе Windows с помощью «Windows API».