МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение

высшего образования «Южно-Уральский государственный университет»

(национальный исследовательский университет)

Высшая школа электроники и компьютерных наук

Кафедра системного программирования

ОТЧЕТ

по практической работе № 3

по дисциплине   
«Теория, методы и средства параллельной обработки информации»

Выполнил:

студент группы КЭ–301

/ Д.А. Иванов /

(подпись)

« » 2024 г.

**Задача:**

Организовать обмен между двумя процессами с помощью канала: 1й процесс 10 раз с интервалом в 2 секунды пишет в канал любое сообщение (например “Hello!”), 2й процесс читает и выводит на экран.

**Листинг программы (отправитель):**

#include <iostream>

#include <windows.h>

int main() {

HANDLE hPipe;

hPipe = CreateNamedPipe("\\\\.\\pipe\\MyPipe",

PIPE\_ACCESS\_OUTBOUND,

PIPE\_TYPE\_MESSAGE | PIPE\_READMODE\_MESSAGE | PIPE\_WAIT,

1,

0,

0,

0,

NULL);

if (hPipe == INVALID\_HANDLE\_VALUE)

{

std::cerr << "Error creating pipe." << std::endl;

return 1;

}

std::cout << "Waiting for the connection of the second process..." << std::endl;

if (!ConnectNamedPipe(hPipe, NULL))

{

std::cerr << "Error connecting to the pipe." << std::endl;

CloseHandle(hPipe);

return 1;

}

std::cout << "Connected." << std::endl;

for (int i = 0; i < 10; ++i)

{

std::string message = "Hello!";

DWORD bytesWritten;

if (!WriteFile(hPipe, message.c\_str(), message.size() + 1, &bytesWritten, NULL))

{

std::cerr << "Error writing to the pipe." << std::endl;

CloseHandle(hPipe);

return 1;

}

std::cout << "Sended: " << message << std::endl;

Sleep(2000);

}

CloseHandle(hPipe);

return 0;

}

**Листинг программы (получатель):**

#include <iostream>

#include <windows.h>

int main()

{

HANDLE hPipe;

hPipe = CreateFile("\\\\.\\pipe\\MyPipe",

GENERIC\_READ,

0,

NULL,

OPEN\_EXISTING,

0,

NULL);

if (hPipe == INVALID\_HANDLE\_VALUE)

{

std::cerr << "Error opening the channel." << std::endl;

return 1;

}

std::cout << "Reading..." << std::endl;

DWORD bytesRead;

char buffer[256];

while (true)

{

if (!ReadFile(hPipe, buffer, sizeof(buffer), &bytesRead, NULL))

{

std::cerr << "Error reading the message from the channel or there are no more messages." << std::endl;

CloseHandle(hPipe);

return 1;

}

if (bytesRead > 0)

{

std::cout << "Received: " << buffer << std::endl;

}

else

{

break;

}

}

}

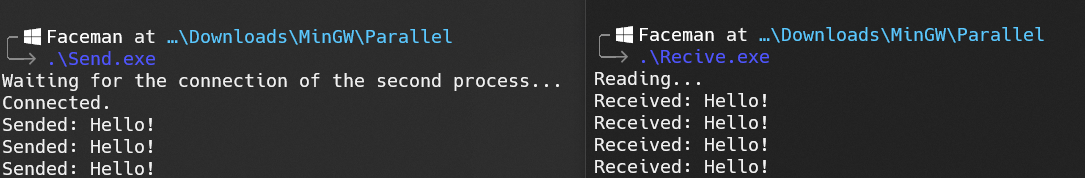


Рисунок 1 – Результат выполнения программы

**Ответы на вопросы:**

1. Что такое API?

‍ API (Application Programming Interface) - это набор компонентов, которые позволяют одной компьютерной программе использовать другую программу. API абстрагирует базовую реализацию и предоставляет только необходимые объекты или действия разработчику, упрощая процесс программирования при создании приложений. Например, Windows API предоставляет набор функций, структур данных и числовых констант, которые позволяют программам взаимодействовать с операционной системой Windows.

1. Как реализуется синхронизация операций чтения/записи и почему?

В операционных системах могут реализовываться следующие механизмы межпроцессного взаимодействия (IPC):

* Механизм именованных каналов (Named Pipes). Это механизм, который позволяет двум процессам обмениваться данными через специальные файлы, называемые именованными каналами. Процессы могут читать данные из канала или записывать их туда.
* Сокеты (Sockets). Это механизм, который позволяет процессам обмениваться данными через сетевые соединения. Сокеты могут использоваться для связи между процессами на одном компьютере или на разных компьютерах в сети.
* Доменные сокеты (Domain Sockets). Это механизм, который позволяет процессам обмениваться данными через специальные домены, которые создаются администратором системы. Доменные сокеты часто используются для создания высокопроизводительных систем межпроцессного взаимодействия.
* Механизм обмена сообщениями (Message Queues). Это механизм, который позволяет процессам отправлять и получать сообщения друг от друга через очередь сообщений. Этот механизм часто используется для синхронизации работы нескольких процессов.
* Механизм общей памяти (Shared Memory). Это механизм, который позволяет процессам обмениваться данными через общую область памяти. Процессы могут читать и записывать данные в эту область памяти напрямую.
* Механизм удаленного вызова процедур (Remote Procedure Call, RPC). Это механизм, который позволяет одному процессу вызывать процедуры другого процесса, как если бы они были локальными. RPC использует протоколы, такие как TCP/IP, для передачи вызовов процедур между процессами.

1. Привести определение канала, описать способ его использования.

Канал — это средство связи между двумя или более процессами. Он представляет собой логический канал, через который процессы могут передавать данные друг другу. Например, в UNIX-подобных системах канал может быть создан с помощью системного вызова pipe(). После создания канал представляет собой именованный файл, который имеет два дескриптора: read и write. Один процесс записывает данные в канал через дескриптор write, а другой читает данные из канала через дескриптор read.