Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Операционные среды и системное программирование

ОТЧЁТ

к лабораторной работе №3

на тему

**ВЗАИМОДЕЙСТВИЕ ПРОЦЕССОВ: ОБМЕН ДАННЫХ**

Выполнил: студент гр.253505 Шпаковский А.В

Проверил: ассистент кафедры информатики Гриценко Н.Ю.

Минск 2024

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc183561834)

[2 Теоритические сведения 4](#_Toc183561835)

[2.1 Механизмы межпроцессного взаимодействия 4](#_Toc183561836)

[2.2 Именованные каналы 4](#_Toc183561837)

[3 Инструментальная языковая среда 5](#_Toc183561838)

[4 Результат запуска программы 6](#_Toc183561839)

[4.1 Иллюстрация запуска и работы программы 6](#_Toc183561840)

[Заключение 8](#_Toc183561841)

[Список использованных источников 9](#_Toc183561842)

[Приложение А (обязательное) Исходный код программного продукта 10](#_Toc183561843)

# 1 ЦЕЛЬ РАБОТЫ

Приложение, разработанное в рамках данной лабораторной работы, представляет собой сервер логирования, реализованный с использованием механизма именованных каналов (Named Pipes) для межпроцессного взаимодействия (IPC) в операционной системе Windows. Основной функционал программы заключается в приеме сообщений от одного или нескольких клиентов, их обработке и записи в лог-файл с добавлением временных меток.

Сервер создает именованный канал, ожидает подключения клиентов, считывает их сообщения и записывает их в лог-файл.

Клиент подключается к серверу через именованный канал и передает сообщение для логирования.

Данная работа наглядно демонстрирует простоту и эффективность использования именованных каналов для реализации многопользовательского взаимодействия в приложении, обеспечивая синхронизацию потоков данных и корректное завершение операций.

Программа визуализирует процесс работы через консольные сообщения, информируя пользователя о подключении клиентов, получении и записи сообщений.

Для разработки приложения использовались язык программирования C++ и функции Windows API. Средой разработки была выбрана Microsoft Visual Studio 2022, предоставляющая мощные инструменты для создания и отладки многозадачных приложений.

# 2 ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

## 2.1 Механизмы межпроцессного взаимодействия

Операционная система Windows предоставляет механизмы для упрощения обмена данными и обмена данными между приложениями. В совокупности действия, включенные этими механизмами, называются межпроцессными коммуникациями (IPC).[1]

Одним из наиболее удобных механизмов IPC в Windows являются именованные каналы (Named Pipes). Этот инструмент позволяет передавать данные между процессами на одном устройстве или в сети.

## 2.2 Именованные каналы

Именованный канал представляет собой именованный односторонний или дуплексный канал для обмена данными между сервером канала и одним или несколькими клиентами.[2] Он позволяет клиентам и серверу взаимодействовать через общий интерфейс с использованием стандартных системных функций, таких как CreateNamedPipe, ConnectNamedPipe, ReadFile и WriteFile.

Основные характеристики именованных каналов включают поддержку множественных клиентов, что позволяет каналу одновременно обслуживать несколько подключений. Кроме того, сообщения передаются в порядке очереди, что исключает потерю данных. Канал также может работать как в синхронном, так и в асинхронном режимах, позволяя настраивать его на блокирующий или неблокирующий режим.

Пример использования именованных каналов в приложении заключается в создании сервера, который ожидает подключения клиентов. Клиенты передают сообщения для обработки, и после их приема сервер записывает данные в файл, что позволяет сохранять историю взаимодействий.

# 3 ИНСТРУМЕНТАЛЬНАЯ ЯЗЫКОВАЯ СРЕДА

Для реализации сервера логирования и клиента был выбран язык программирования C++, предоставляющий мощные инструменты для работы с системными ресурсами операционной системы. С его помощью можно эффективно использовать Windows API, включая функции для создания и работы с именованными каналами. Благодаря поддержке низкоуровневого программирования и детального управления памятью, C++ позволяет оптимизировать производительность приложений.

Windows API обеспечивает обширный функционал для реализации межпроцессного взаимодействия. В рамках лабораторной работы использовались функции CreateNamedPipe, ConnectNamedPipe, ReadFile, WriteFile и CloseHandle. Эти функции позволяют создавать именованные каналы, устанавливать соединение между процессами, передавать данные и корректно завершать операции. Например, сервер использует CreateNamedPipe для создания канала и ConnectNamedPipe для ожидания подключения клиента, а клиент применяет CreateFile для установления соединения с сервером.

Средой разработки было выбрано Microsoft Visual Studio 2022, предоставляющее удобные инструменты для создания, отладки и тестирования программ. Visual Studio позволяет быстро интегрировать проект с системными библиотеками Windows API, что упрощает разработку приложений с использованием низкоуровневых системных функций. Кроме того, встроенные средства анализа производительности и работы с памятью помогают находить узкие места в коде и оптимизировать его.

Приложение разрабатывалось и тестировалось в среде Windows 10, что позволило использовать современные возможности операционной системы для работы с именованными каналами. Операционная система Windows предоставляет надежную поддержку этого механизма, что делает его подходящим для создания многопользовательских приложений.

Таким образом, сочетание языка программирования C++, функционала Windows API и инструментов Visual Studio позволило реализовать простую и эффективную систему логирования, демонстрирующую основные принципы межпроцессного взаимодействия.

# 4 РЕЗУЛЬТАТ ЗАПУСКА ПРОГРАММЫ

## 4.1 Иллюстрация запуска и работы программы

На рисунке 4.1 мы наблюдаем окно консоли запущенного приложения-сервера, который ожидает подключения клиентов.

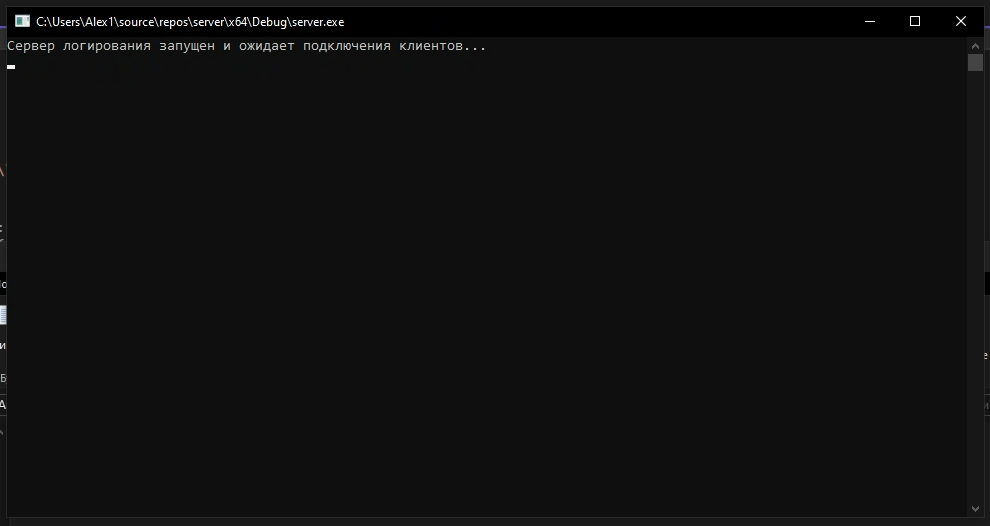


Рисунок 4.1 — Окно консоли после запуска приложения-сервера

На рисунке 4.2 изображено окно консоли приложения-сервера после отправки сообщения клиентом.

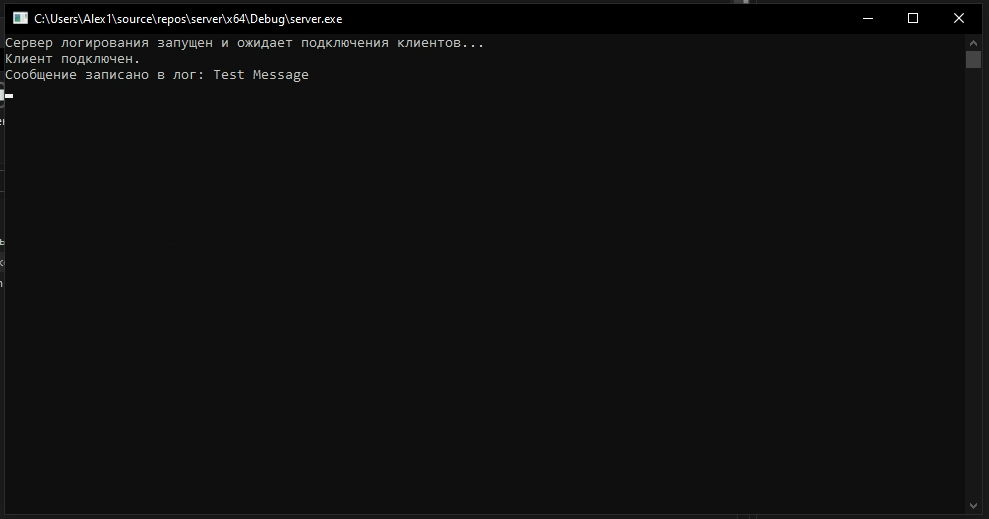


Рисунок 4.2 — Окно приложения-сервера после ввода сообщения

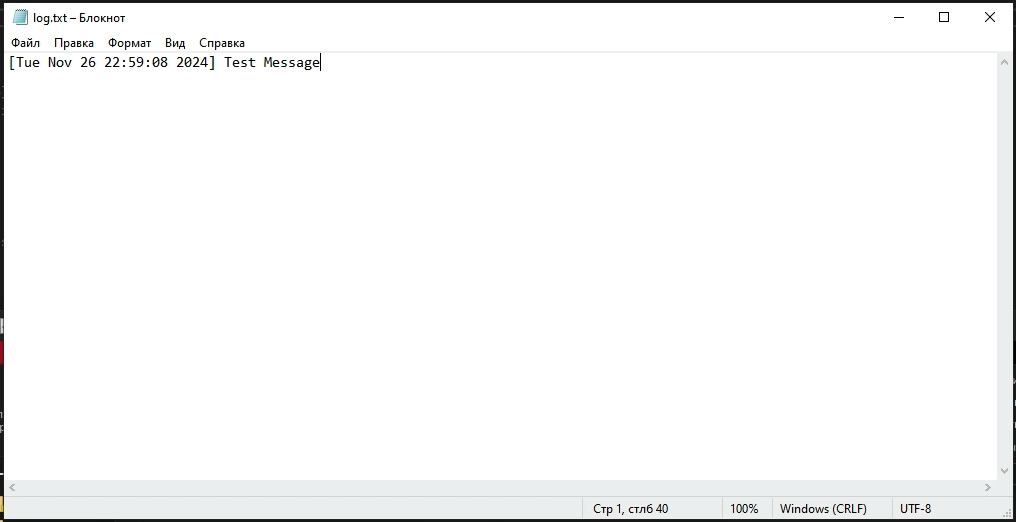


Рисунок 4.3 — Содержание лог-файла

Программа находится в режиме ожидания подключения клиентов и приема их сообщений. Этот этап является критически важным, поскольку он определяет, как сервер взаимодействует с клиентами и обрабатывает поступающую информацию. Когда клиент успешно подключается, сервер начинает процесс получения сообщения. После того как сообщение поступает, сервер не просто сохраняет его, а также обрабатывает, добавляя к нему временную метку. Это означает, что каждая запись о взаимодействии будет содержать не только текст сообщения, но и конкретное время его получения, что значительно улучшает возможности анализа.

Полученные данные записываются в лог-файл, который изображен на рисунке 4.3. Этот механизм логирования обеспечивает фиксацию событий в формате, удобном для последующего анализа. Каждое сообщение аккуратно записывается в отдельной строке файла, что упрощает чтение и обработку информации. В строке содержится как текст сообщения, так и временная метка, что создает четкую и понятную структуру данных.

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной лабораторной работы по курсу «Операционные системы и системное программирование» был реализован сервер логирования, который демонстрирует использование именованных каналов для межпроцессного взаимодействия. Основная задача программы заключалась в разработке приложения, которое может принимать сообщения от клиентов, добавлять к ним временные метки и записывать их в лог-файл. В процессе работы особое внимание было уделено корректному взаимодействию между процессами, а также синхронизации операций записи данных в файл, что обеспечило надежность и упорядоченность логирования.

В реализации использовались функции Windows API, такие как CreateNamedPipe, ConnectNamedPipe, ReadFile и WriteFile, которые обеспечивают создание именованного канала, передачу данных между процессами и их запись в файл. Эти функции позволяют организовать стабильный и безопасный обмен данными, что делает систему логирования эффективной и масштабируемой.

Программа продемонстрировала эффективность использования именованных каналов как механизма межпроцессного взаимодействия, а также дала практические навыки работы с Windows API. Лабораторная работа позволила углубить знания о межпроцессном взаимодействии и разработке многозадачных приложений, где важно учитывать синхронизацию данных и управление ресурсами операционной системы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Взаимодействие между процессами - Win32 apps | Microsoft Learn – Электронный ресурс. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/ipc/interprocess-communications

[2] Именованные каналы - Win32 apps - Microsoft Learn – Электронный ресурс. – Режим доступа: https://learn.microsoft.com/ru-ru/windows/win32/ipc/named-pipes

# ПРИЛОЖЕНИЕ А (обязательное) Исходный код программного продукта

#include <windows.h>

#include <iostream>

#include <fstream>

#include <string>

#include <ctime>

#define PIPE\_NAME L"\\\\.\\pipe\\LogPipe"

#define BUFFER\_SIZE 512

void LogMessage(const std::string& message) {

std::ofstream logfile("log.txt", std::ios::app);

if (logfile.is\_open()) {

time\_t now = time(0);

char dt[26];

ctime\_s(dt, sizeof(dt), &now);

logfile << "[" << dt << "] " << message << std::endl;

logfile.close();

}

else {

std::cerr << "Ошибка открытия файла log.txt" << std::endl;

}

}

int main() {

setlocale(0, "rus");

HANDLE hPipe;

char buffer[BUFFER\_SIZE];

DWORD bytesRead;

std::cout << "Сервер логирования запущен и ожидает подключения клиентов..." << std::endl;

hPipe = CreateNamedPipe(

PIPE\_NAME,

PIPE\_ACCESS\_INBOUND,

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

PIPE\_UNLIMITED\_INSTANCES,

BUFFER\_SIZE,

BUFFER\_SIZE,

0,

NULL

);

if (hPipe == INVALID\_HANDLE\_VALUE) {

DWORD error = GetLastError();

std::cerr << "Ошибка создания канала. Код ошибки: " << error << std::endl;

return 1;

}

while (true) {

if (ConnectNamedPipe(hPipe, NULL) || GetLastError() == ERROR\_PIPE\_CONNECTED) {

std::cout << "Клиент подключен." << std::endl;

if (ReadFile(hPipe, buffer, BUFFER\_SIZE - 1, &bytesRead, NULL)) {

buffer[bytesRead] = '\0'; // Завершаем строку

std::string message(buffer);

LogMessage(message); // Логируем сообщение

std::cout << "Сообщение записано в лог: " << message << std::endl;

}

else {

std::cerr << "Ошибка при чтении из канала." << std::endl;

}

DisconnectNamedPipe(hPipe);

}

}

CloseHandle(hPipe);

return 0;

}

#include <windows.h>

#include <iostream>

#include <string>

#define PIPE\_NAME L"\\\\.\\pipe\\LogPipe"

#define BUFFER\_SIZE 512

int main() {

setlocale(0, "rus");

HANDLE hPipe;

DWORD bytesWritten;

std::string message;

std::cout << "Введите сообщение для логирования: ";

std::getline(std::cin, message);

// Подключение к именованному каналу

hPipe = CreateFile(

PIPE\_NAME,

GENERIC\_WRITE,

0,

NULL,

OPEN\_EXISTING,

0,

NULL

);

if (hPipe == INVALID\_HANDLE\_VALUE) {

DWORD error = GetLastError();

if (error == ERROR\_PIPE\_BUSY) {

std::cerr << "Сервер занят. Попробуйте позже." << std::endl;

}

else {

std::cerr << "Не удалось подключиться к серверу. Код ошибки: " << error << std::endl;

}

return 1;

}

if (WriteFile(hPipe, message.c\_str(), message.length(), &bytesWritten, NULL)) {

std::cout << "Сообщение успешно отправлено серверу." << std::endl;

}

else {

std::cerr << "Ошибка при отправке сообщения." << std::endl;

}

CloseHandle(hPipe);

return 0;

}

"