**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Вычислительной техники**

**отчёт   
ЛАБОРАТОРНАЯ РАБОТА № 4**

**по дисциплине «Операционные Системы»**

**Тема: Межпроцессное взаимодействие**

|  |  |  |
| --- | --- | --- |
| Студент гр. 3311 |  | Сапронов К.Д. |
| Преподаватель |  | Тимофеев А.В. |

Санкт-Петербург

2025

**Цель работы:** исследовать инструменты и механизмы взаимодействия процессов в Windows.

**Задание 4.1. Реализация решения задачи о читателях-писателях.**

1. Выполнить решение задачи о читателях-писателях, для чего

необходимо разработать консольные приложения «Читатель» и

«Писатель»: одновременно запущенные экземпляры процессов-читателей и процессов-писателей должны совместно работать с буферной памятью в виде проецируемого файла.

2. Запустите приложения читателей и писателей, суммарное

количество одновременно работающих читателей и писателей должно

быть не менее числа страниц буферной памяти. Проверьте

функционирование приложений, проанализируйте журнальные файлы

процессов, постройте сводные графики смены «состояний» для не менее

5 процессов-читателей и 5 процессов-писателей.

3. Постройте графики в Excel/Python и дайте свои комментарии.

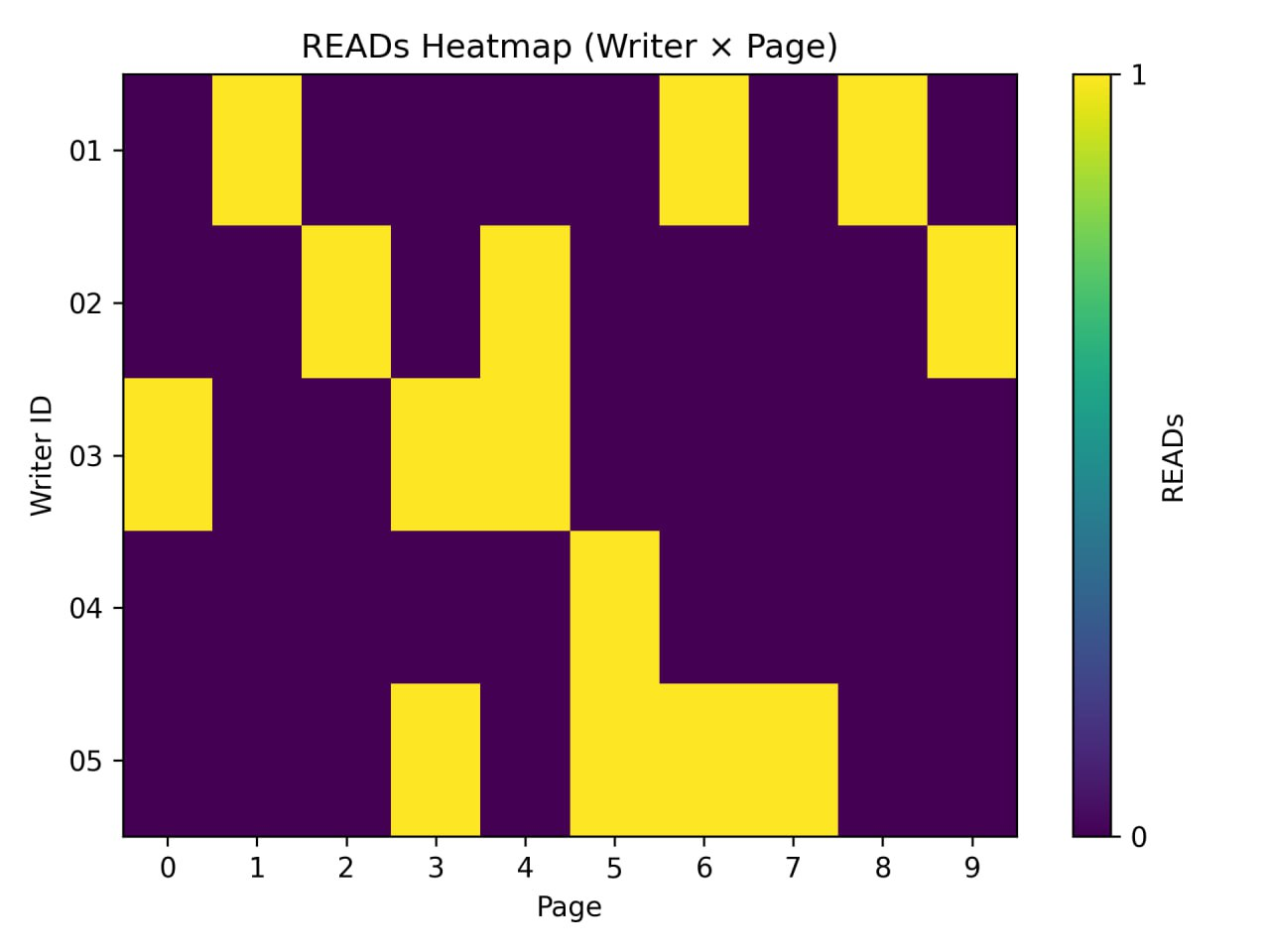
**Выполнение задания**

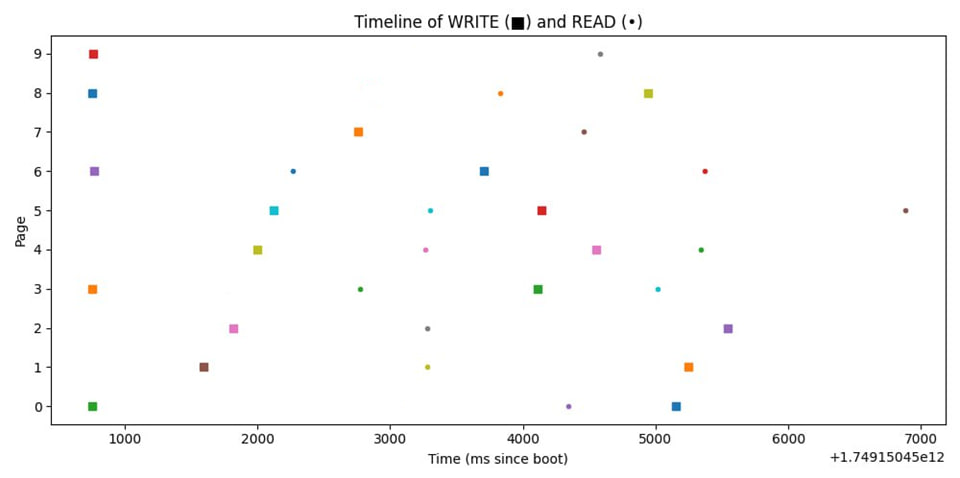
Для реализации модели писателей и читателей была разработана программа, использующая механизмы WinAPI для работы с разделяемой памятью и синхронизации процессов. В начале работы создается именованное отображение файла (SharedBuffer) размером 10 страниц по 4096 байт каждая. Для координации доступа к каждой странице используются отдельные именованные мьютексы (PageMutex\_0...PageMutex\_9), что позволяет разным процессам одновременно работать с разными страницами. Состояние страниц отслеживается через первый байт каждой страницы: значение '0' означает чистую страницу, доступную для записи, а '1' - грязную страницу, готовую для чтения.

Каждый из пяти писателей при запуске получает уникальный идентификатор и инициализирует подключение к разделяемой памяти и набору мьютексов страниц. При первом создании разделяемой памяти все страницы автоматически инициализируются как чистые. Писатель выполняет серию из пяти операций записи, предварительно находя чистую страницу путем проверки первого байта под захваченным мьютексом. После захвата мьютекса писатель помечает страницу как грязную, записывает в нее строку формата "hello\_ID" с указанием своего идентификатора, выдерживает случайную задержку от 0.5 до 1.5 секунд и освобождает мьютекс. Все операции записываются в индивидуальный лог-файл с точными временными метками, включая моменты ожидания записи, непосредственной записи и освобождения ресурсов.

Пять процессов-читателей подключаются к существующей разделяемой памяти и выполняют серию из пяти операций чтения. Читатель ищет грязную страницу, проверяя первый байт под захваченным мьютексом. После нахождения подходящей страницы читатель считывает ее содержимое, начиная со второго байта, помечает страницу как чистую, фиксирует прочитанные данные в лог-файле, выдерживает случайную задержку от 0.5 до 1.5 секунд и освобождает мьютекс. Лог-файлы читателей содержат записи о моментах ожидания чтения, фактах чтения, считанных данных и освобождения ресурсов.

Полученные рещультаты на графиках ниже:





По хитмепу видно, что все страницы были использованы писателями. По графику таймлайна работы процессов сначала работают писатели, после 2 секунд начинают работу читатели. По графику видно что процессы (и писатели и читатели) работают одновременно на разных страницах, но на каждой странице одновременно работает только один процесс. Также процессы записи и чтения чередуются – писатели пишут только на чистых страницах, а читатели читают с грязных.

**Задание 4.2. Использование именованных каналов для реализации сетевого межпроцессного взаимодействия.**

1. Создайте два консольных приложения с меню (каждая

выполняемая функция и/или операция должна быть доступна по

отдельному пункту меню), которые выполняют:

- приложение-сервер создает именованный канал, выполняет установление и отключение соединения, создает объект «событие», осуществляет ввод данных с клавиатуры и их асинхронную запись в именованный канал, выполняет ожидание завершения операции ввода-вывода;

- приложение-клиент подключается к именованному каналу, в асинхронном режиме считывает содержимое из именованного канала файла и отображает на экран.

2. Запустите приложения и проверьте обмен данных между

процессами. Запротоколируйте результаты в отчет. Дайте свои

комментарии в отчете относительно выполнения функций Win32 API.

**Выполнение задания**

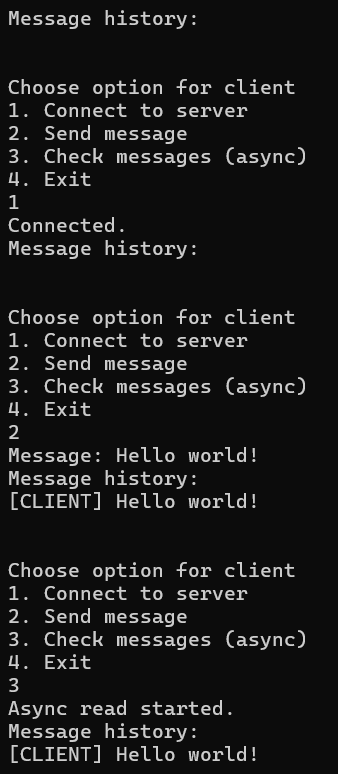
Программа представляет собой клиент-серверное приложение для обмена сообщениями через именованные каналы Windows. Сервер создает канал с помощью CreateNamedPipe и ожидает подключения клиента через ConnectNamedPipe. Клиент подключается к каналу через CreateFile с указанием FILE\_FLAG\_OVERLAPPED для асинхронных операций.

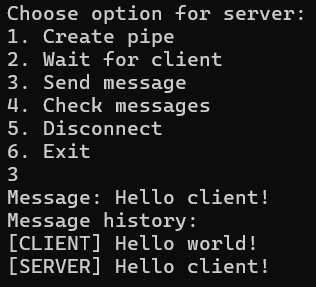
Основной цикл сервера предоставляет консольное меню для управления: создание канала, ожидание подключения, отправка и проверка сообщений. При отправке сообщения сервер использует WriteFile для записи в канал и сохраняет сообщение в истории. Для чтения используется синхронный ReadFile, который блокирует выполнение до получения данных от клиента.

Клиентская часть также реализует консольное меню с возможностью подключения, отправки сообщений и асинхронной проверки входящих сообщений через ReadFileEx с callback-функцией ReadCompletionCallback. Полученные сообщения автоматически добавляются в историю.

Обе стороны поддерживают историю сообщений (message\_history) с ограничением по размеру (MAX\_HISTORY). При передаче данных используется буфер фиксированного размера (BUFFER\_SIZE). Состояние подключения отслеживается через флаги client\_connected (сервер) и проверку pipe\_handle (клиент). Работа завершается по команде пользователя с корректным закрытием handles каналов.

**Пример выполнения программы.**

****

****

**Выводы:**

В ходе работы были изучены два подхода к выполнению межпроцессного взаимодействий – с помощью разделяемой памяти с “читальным залом” и именованных каналов, были написаны программы для каждого из подходов. Второй подход оказался заметно проще в реализации за счет встроенной системы синхронизации, в то время как читатели и писатели требовали ручного управления состояниями.