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 ДАННЫМИ**

|  |
| --- |
| Выполнил: студент гр. 253503  Котова К.А. |
| Проверил: ассистент кафедры информатики Гриценко Н.Ю. |

Минск 2024

**СОДЕРЖАНИЕ**

[1 Формулировка задачи 3](#_Toc181292096)

[2 Описание основных компонентов программы 4](#_Toc181292097)

[2.1 Создание разделяемой памяти 4](#_Toc181292098)

[2.2 Отображение разделяемой памяти в адресное пространство процесса 4](#_Toc181292099)

[2.3 Создание мьютекса для синхронизации доступа 4](#_Toc181292100)

[2.4 Запись данных в разделяемую память 4](#_Toc181292101)

[2.5 Чтение данных из разделяемой памяти 4](#_Toc181292102)

[3 Результаты выполнения работы программы 5](#_Toc181292103)

[3.1 Работа процесса-писателя 5](#_Toc181292104)

[3.2 Работа процесса читателя 5](#_Toc181292105)

[Вывод 6](#_Toc181292106)

[Список использованных источников 7](#_Toc181292107)

[Приложение А 8](#_Toc181292108)

# 

# 1 ФОРМУЛИРОВКА ЗАДАЧИ

Целью данной лабораторной работы является создание программы для организации обмена через разделяемую память.

Задачей является организовать совместное использование содержимого разделяемой памяти несколькими процессами (потоками) с предотвращением коллизий.

Пример архитектуры комплекса: пул буферов в памяти, передаваемых между процессами-«пользователями» с защитой от коллизий семафорами (мьютексами).

Процессы-«пользователи»: получение (захват) буфера; заполнение данными или обработка имеющегося содержимого; возврат в пул или адресная передача следующему процессу.

# 2 ОПИСАНИЕ ОСНОВНЫХ КОМПОНЕНТОВ ПРОГРАММЫ

## 2.1 Создание разделяемой памяти

Программа использует функцию CreateFileMappingW[1] для создания объекта разделяемой памяти.

Вызов CreateFileMappingW[1] с INVALID\_HANDLE\_VALUE и именем (L"Local\\MySharedMemory") создаёт в оперативной памяти объект, доступ к которому можно получить через это имя. Использование INVALID\_HANDLE\_VALUE указывает, что создается объект в памяти, а не на диске.

## 2.2 Отображение разделяемой памяти в адресное пространство процесса

После создания разделяемой памяти программа вызывает функцию MapViewOfFile[2], чтобы отобразить её в адресное пространство текущего процесса.

MapViewOfFile[2] возвращает указатель, с которым программа может работать как с обычным указателем на область памяти, записывая туда данные.

## 2.3 Создание мьютекса для синхронизации доступа

Чтобы обеспечить безопасный доступ нескольких процессов к разделяемой памяти, программа использует мьютекс.

Мьютекс гарантирует, что только один процесс может писать в разделяемую память в любой момент времени.

## 2.4 Запись данных в разделяемую память

В основном цикле программа ожидает захвата мьютекса. Как только мьютекс становится доступным, программа запрашивает у пользователя ввод. Введенное сообщение записывается в разделяемую память с помощью CopyMemory[3]. Если пользователь вводит "exit", программа записывает это значение в память и завершает цикл, что служит сигналом для других процессов о завершении работы.

## Чтение данных из разделяемой памяти

После захвата мьютекса программа проверяет содержимое разделяемой памяти. Если в памяти содержится строка "exit", программа завершает цикл и прекращает работу, поскольку это сигнал от writer о завершении. В противном случае данные выводятся на экран.

# 3 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

## 3.1 Работа процесса-писателя

На рисунке 3.1 программа writer записывает данные в разделяемую память для передачи другому процессу (reader), синхронизируя доступ с помощью мьютекса. На скриншоте последовательно вводятся сообщения, завершает ввод сообщение exit, после чего программа подтверждает запись каждого сообщения и завершает работу. Ввод команды exit служит сигналом завершения для читателя, информируя его, что больше данных не будет записано.

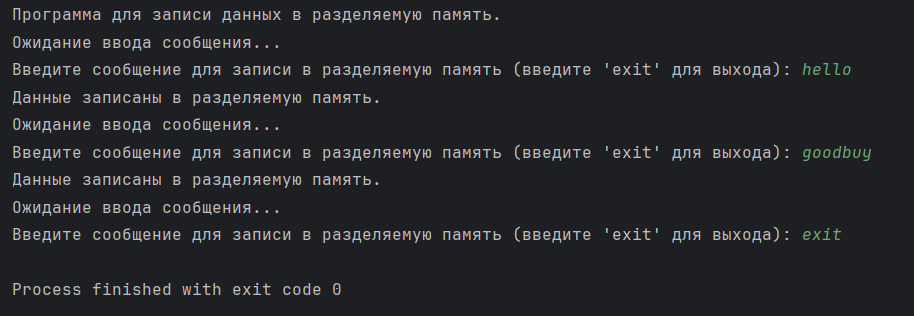


Рисунок 3.1 – Работа процесса-писателя

## 3.2 Работа процесса читателя

На рисунке 3.2 программа-читатель читает сообщения из разделяемой памяти, которые были записаны программой-писателем, после чего обнаруживает сообщение exit, сигнализирующее о завершении работы. При получении этого сигнала читатель выводит сообщение о завершении и корректно завершает свою работу, освобождая ресурсы.

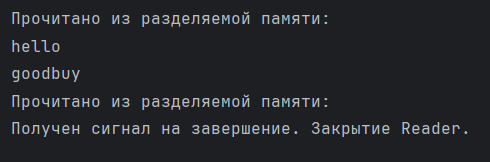


Рисунок 3.2 – Результат работы процесса-читателя

# ВЫВОД

В ходе выполнения лабораторной работы была реализована система межпроцессного взаимодействия с использованием разделяемой памяти и механизмов синхронизации. Разработаны две программы: writer, записывающая данные в разделяемую память, и reader, считывающая эти данные. Для управления доступом к разделяемой памяти использовался мьютекс, предотвращающий одновременный доступ к ресурсу несколькими процессами.

Тестирование показало, что синхронизация с помощью мьютекса позволяет организовать безопасный доступ к разделяемой памяти, гарантируя целостность и последовательность данных при обмене между процессами.

Программа корректно обрабатывает записи и завершает работу при получении команды exit, что подтверждает стабильность разработанного решения. Данный подход полезен в сценариях межпроцессного взаимодействия, где требуется надежный обмен данными между независимыми процессами.

Таким образом, лабораторная работа продемонстрировала практическое применение технологии разделяемой памяти и механизмов синхронизации для организации межпроцессного взаимодействия в операционной системе Windows.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Документация Microsoft для функции CreateFileMapping [Электронный ресурс]. – Электронные данные. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/api/winbase/nf-winbase-createfilemappinga – Дата доступа: 28.10.2024.

[2] Документация Microsoft для функции MapViewOfFile [Электронный ресурс]. – Электронные данные. – Режим доступа: https://learn.microsoft.com/en-us/windows/win32/api/memoryapi/nf-memoryapi-mapviewoffile –Дата доступа: 28.10.2024.

[3] Документация CopyMemory [Электронный ресурс]. – Электронные данные. – Режим доступа <https://learn.microsoft.com/en-us/previous-versions/windows/desktop/legacy/aa366535(v=vs.85)>: 28.09.2024.

# ПРИЛОЖЕНИЕ А (обязательное) Код программы

Содержимое файла writer.cpp

#include <windows.h>

#include <iostream>

const int BUF\_SIZE = 256;

int main() {

SetConsoleOutputCP(CP\_UTF8);

std::cout << "Программа для записи данных в разделяемую память.\n";

HANDLE hMapFile = CreateFileMappingW(

INVALID\_HANDLE\_VALUE,

NULL,

PAGE\_READWRITE,

0,

BUF\_SIZE,

L"Local\\MySharedMemory1"

);

if (hMapFile == NULL) {

std::cerr << "Не удалось создать разделяемую память: " << GetLastError() << std::endl;

return 1;

}

LPCTSTR pBuf = (LPTSTR)MapViewOfFile(

hMapFile,

FILE\_MAP\_ALL\_ACCESS,

0,

0,

BUF\_SIZE

);

if (pBuf == NULL) {

std::cerr << "Не удалось отобразить разделяемую память: " << GetLastError() << std::endl;

CloseHandle(hMapFile);

return 1;

}

HANDLE hMutex = CreateMutexW(

NULL,

FALSE,

L"Local\\Mutex12"

);

if (hMutex == NULL) {

std::cerr << "Не удалось создать мьютекс: " << GetLastError() << std::endl;

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

return 1;

}

while (true) {

std::cout << "Ожидание ввода сообщения...\n";

WaitForSingleObject(hMutex, INFINITE);

std::cout << "Введите сообщение для записи в разделяемую память (введите 'exit' для выхода): ";

std::string input;

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

if (input == "exit") {

CopyMemory((PVOID)pBuf, "exit", 5 \* sizeof(char));

break;

}

std::string currentData = (char\*)pBuf;

if (!currentData.empty()) {

currentData += "\n";

}

currentData += input;

CopyMemory((PVOID)pBuf, currentData.c\_str(), (currentData.size() + 1) \* sizeof(char));

std::cout << "Данные записаны в разделяемую память.\n";

ReleaseMutex(hMutex);

}

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

CloseHandle(hMutex);

return 0;

}

Содержимое файла reader.cpp

#include <windows.h>

#include <iostream>

const int BUF\_SIZE = 256;

int main() {

SetConsoleOutputCP(CP\_UTF8);

HANDLE hMapFile = OpenFileMappingW(

FILE\_MAP\_ALL\_ACCESS,

FALSE,

L"Local\\MySharedMemory1"

);

if (hMapFile == NULL) {

std::cerr << "Не удалось открыть разделяемую память: " << GetLastError() << std::endl;

return 1;

}

LPCTSTR pBuf = (LPTSTR)MapViewOfFile(

hMapFile,

FILE\_MAP\_ALL\_ACCESS,

0,

0,

BUF\_SIZE

);

if (pBuf == NULL) {

std::cerr << "Не удалось отобразить разделяемую память: " << GetLastError() << std::endl;

CloseHandle(hMapFile);

return 1;

}

HANDLE hMutex = OpenMutexW(

MUTEX\_ALL\_ACCESS,

FALSE,

L"Local\\Mutex12"

);

if (hMutex == NULL) {

std::cerr << "Не удалось открыть мьютекс: " << GetLastError() << std::endl;

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

return 1;

}

while (true) {

WaitForSingleObject(hMutex, INFINITE);

std::cout << "Прочитано из разделяемой памяти:" << std::endl;

if (strcmp((const char\*)pBuf, "exit") == 0) {

std::cout << "Получен сигнал на завершение. Закрытие Reader." << std::endl;

ReleaseMutex(hMutex);

break;

}

std::cout << pBuf << std::endl;

ReleaseMutex(hMutex);

Sleep(100);

}

UnmapViewOfFile(pBuf);

CloseHandle(hMapFile);

CloseHandle(hMutex);

return 0;

}