

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра Вычислительной техники

ОТЧЁТ
по лабораторной работе №4
по дисциплине «Операционные системы»
Тема: Межпроцессорное взаимодействие

Студент гр. 9308

Соболев М.С.

Преподаватель

Тимофеев А.В.

Санкт-Петербург,
2021

Оглавление

1. Введение	3
2. Реализация решения задачи о читателях-писателях.....	5
2.1. Консольное приложение «Диспетчер», запускающее приложения «Писатель» и «Читатель».....	5
2.2. Сводные графики смены состояний для приложений-писателей	7
2.3. Сводные графики смены состояний для приложений-читателей	10
2.4. График доступности страниц в оперативной памяти	13
2.5. Журнальный файл процессов-писателей и процессов-писателей	16
2.6. Исходный код программы «Диспетчер».....	28
2.7. Исходный код программы «Писатель».....	37
2.8. Исходный код программы «Читатель»	47
2.9. Вывод	58
3. Использование именованных каналов для реализации сетевого межпроцессорного взаимодействия	59
3.1. Приложения «Сервер» и «Клиент»	59
3.2. Исходный код программы «Сервер»	64
3.3. Исходный код программы «Клиент»	71
3.4. Вывод	75
4. Список использованных источников.....	76

1. Введение

Тема работы: Межпроцессорное взаимодействие.

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

Указания к выполнению

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: о размер страницы буферной памяти равен размеру физической страницы оперативной памяти; о число страниц буферной памяти равно сумме цифр в номере студенческого билета без учета первой цифры.
- страницы буферной памяти должны быть заблокированы в оперативной памяти (функция VirtualLock);
- длительность выполнения процессами операций «чтения» и «записи» задается случайным образом в диапазоне от 0,5 до 1,5 сек.;
- для синхронизации работы процессов необходимо использовать объекты синхронизации типа «семафор» и «мьютекс»;
- процессы-читатели и процессы-писатели ведут свои журнальные файлы, в которые регистрируют переходы из одного «состояния» в другое (начало ожидания, запись или чтение, переход к освобождению) с указанием кода времени (функция TimeGetTime). Для состояний «запись» и «чтение» необходимо также запротоколировать номер рабочей страницы.

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, дайте свои комментарии относительно переходов процессов из одного состояния в другое. Постройте графики занятости страниц буферной памяти (проецируемого файла) во времени, дайте свои комментарии.

3. Подготовьте итоговый отчет с развернутыми выводами по заданию.

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 (функция Win32 API – CreateNamedPipe), выполняет установление и отключение соединения (функции Win32 API – ConnectNamedPipe, DisconnectNamedPipe), создаёт объект «событие» (функция Win32 API – CreateEvent) осуществляет ввод данных с клавиатуры и их асинхронную запись в именованный канал (функция Win32 API – WriteFile), выполняет ожидание завершения операции ввода-вывода (функция Win32 API – WaitForSingleObject);

- приложение-клиент подключается к именованному каналу (функция Win32 API – CreateFile), в асинхронном режиме считывает содержимое из именованного канала файла (функция Win32 API – ReadFileEx) и отображает на экран.

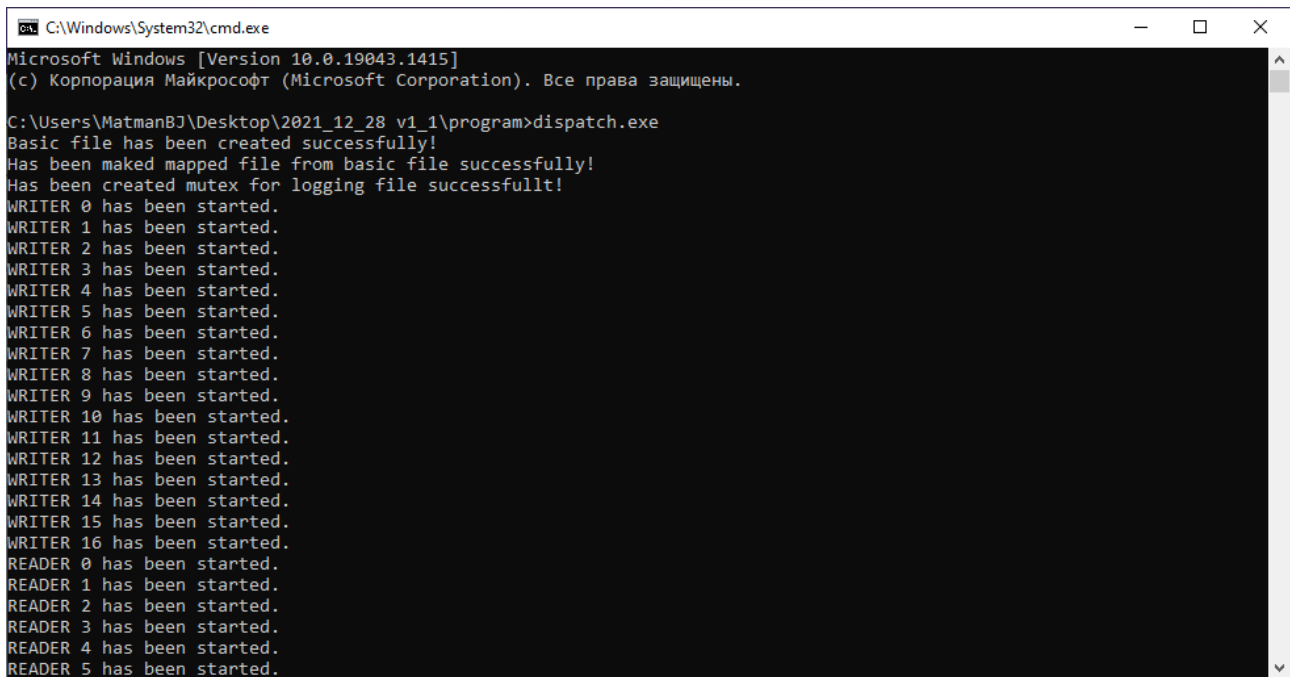
2. Запустите приложения и проверьте обмен данных между процессами. Запротоколируйте результаты в отчёт. Дайте свои комментарии в отчёте относительно выполнения функций Win32 API.

3. Подготовьте итоговый отчёт с развернутыми выводами по заданию.

2. Реализация решения задачи о читателях-писателях

2.1. Консольное приложение «Диспетчер», запускающее приложения «Писатель» и «Читатель»

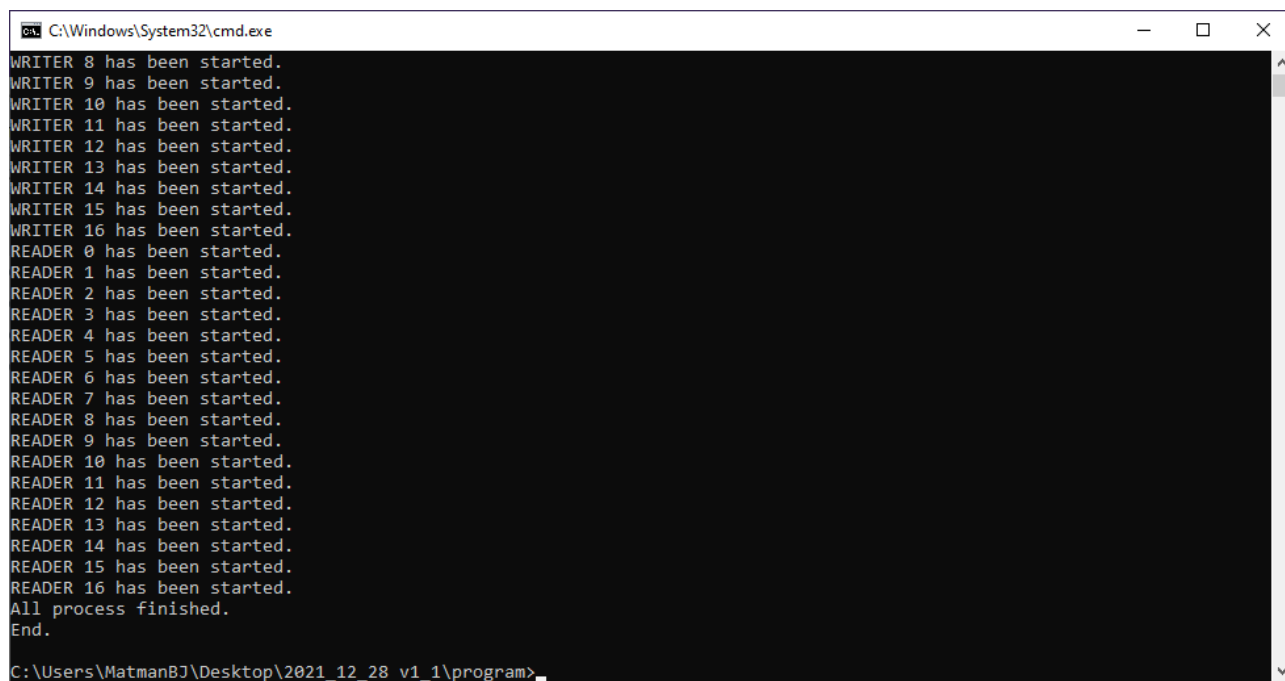
Вывод сообщения о запуске приложений «Писатель» и «Читатель» в консольном приложении «Диспетчер».



```
C:\Windows\System32\cmd.exe
Microsoft Windows [Version 10.0.19043.1415]
(c) Корпорация Майкрософт (Microsoft Corporation). Все права защищены.

C:\Users\MatmanBJ\Desktop\2021_12_28 v1_1\program>dispatch.exe
Basic file has been created successfully!
Has been maked mapped file from basic file successfully!
Has been created mutex for logging file successfullt!
WRITER 0 has been started.
WRITER 1 has been started.
WRITER 2 has been started.
WRITER 3 has been started.
WRITER 4 has been started.
WRITER 5 has been started.
WRITER 6 has been started.
WRITER 7 has been started.
WRITER 8 has been started.
WRITER 9 has been started.
WRITER 10 has been started.
WRITER 11 has been started.
WRITER 12 has been started.
WRITER 13 has been started.
WRITER 14 has been started.
WRITER 15 has been started.
WRITER 16 has been started.
READER 0 has been started.
READER 1 has been started.
READER 2 has been started.
READER 3 has been started.
READER 4 has been started.
READER 5 has been started.
```

Рисунок 1: Вывод сообщения о запуске приложений «Писатель» и «Читатель» в консольном приложении «Диспетчер»



```
C:\Windows\System32\cmd.exe
WRITER 8 has been started.
WRITER 9 has been started.
WRITER 10 has been started.
WRITER 11 has been started.
WRITER 12 has been started.
WRITER 13 has been started.
WRITER 14 has been started.
WRITER 15 has been started.
WRITER 16 has been started.
READER 0 has been started.
READER 1 has been started.
READER 2 has been started.
READER 3 has been started.
READER 4 has been started.
READER 5 has been started.
READER 6 has been started.
READER 7 has been started.
READER 8 has been started.
READER 9 has been started.
READER 10 has been started.
READER 11 has been started.
READER 12 has been started.
READER 13 has been started.
READER 14 has been started.
READER 15 has been started.
READER 16 has been started.
All process finished.
End.
C:\Users\MatmanBJ\Desktop\2021_12_28 v1_1\program>
```

Рисунок 2: Вывод сообщения о запуске приложений «Писатель» и «Читатель» в консольном приложении «Диспетчер»

2.2. Сводные графики смены состояний для приложений-писателей

Приведены графики смены состояний для 5 из 17 приложений-писателей.



Рисунок 3: График смены состояний для приложения-писателя 1



Рисунок 4: График смены состояний для приложения-писателя 2



Рисунок 5: График смены состояний для приложения-писателя 3

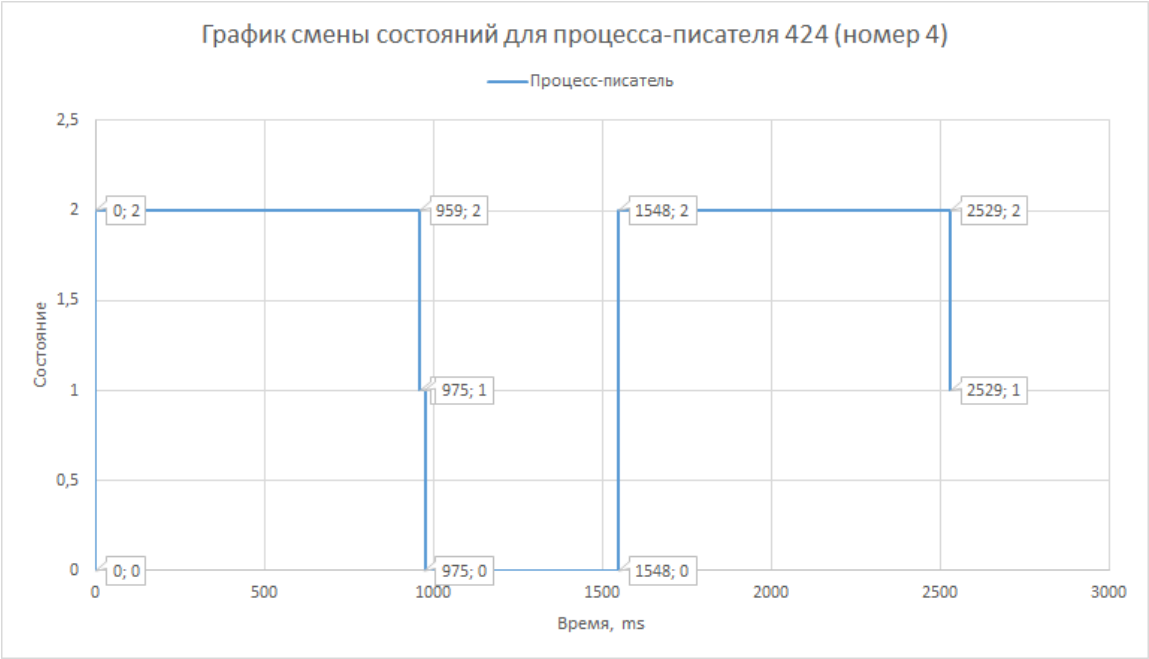


Рисунок 6: График смены состояний для приложения-писателя 4

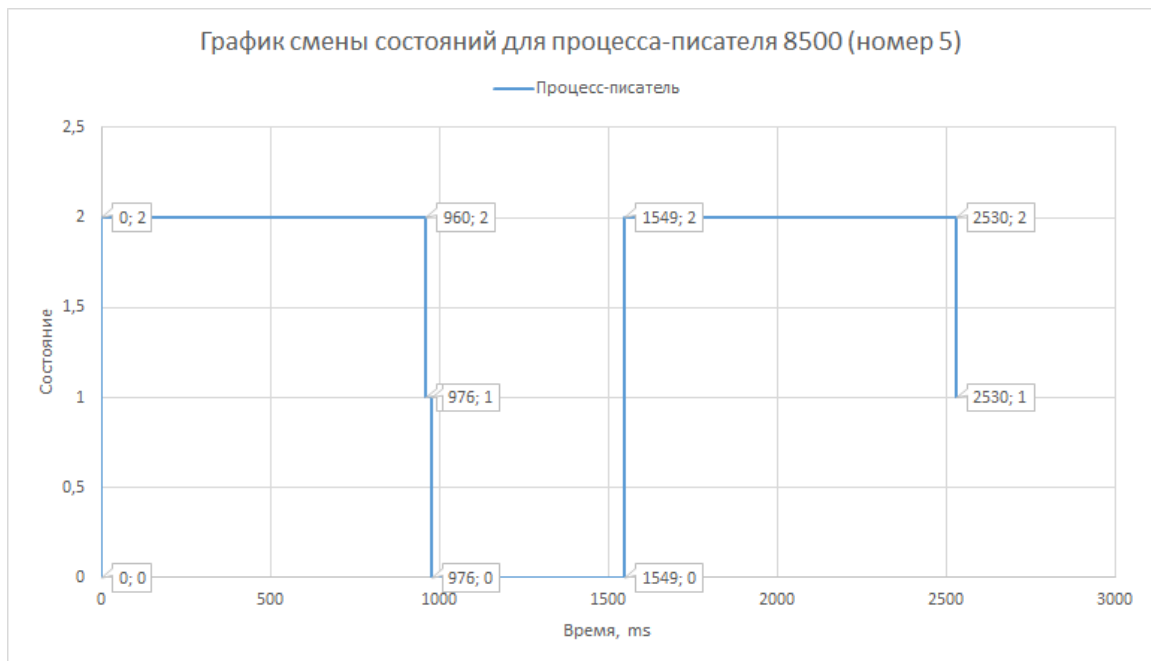


Рисунок 7: График смены состояний для приложения-писателя 5

2.3. Сводные графики смены состояний для приложений-читателей

Приведены графики смены состояний для 5 из 17 приложений-читателей.

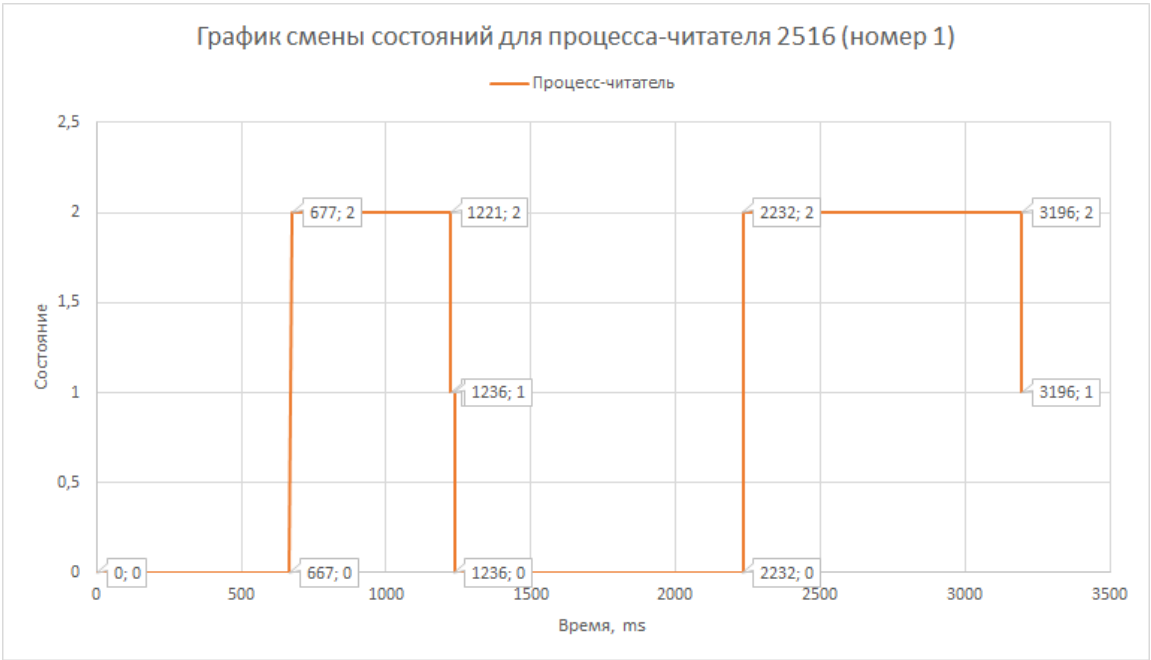


Рисунок 8: График смены состояний для приложения-читателя 1

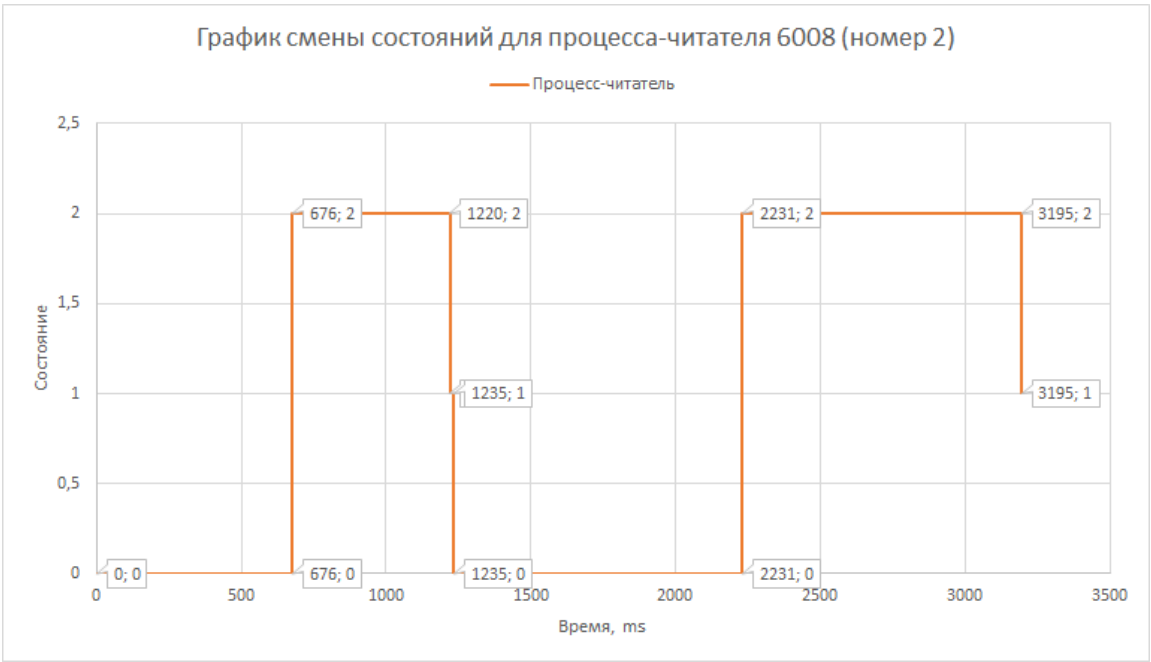


Рисунок 9: График смены состояний для приложения-читателя 2



Рисунок 10: График смены состояний для приложения-читателя 3

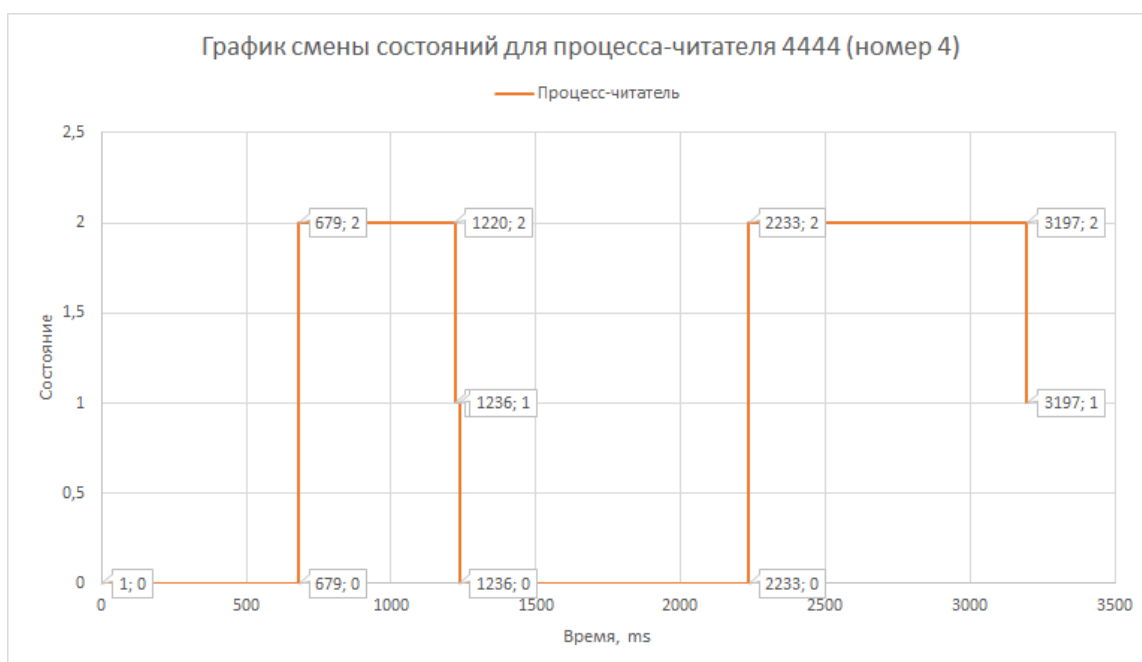


Рисунок 11: График смены состояний для приложения-читателя 4

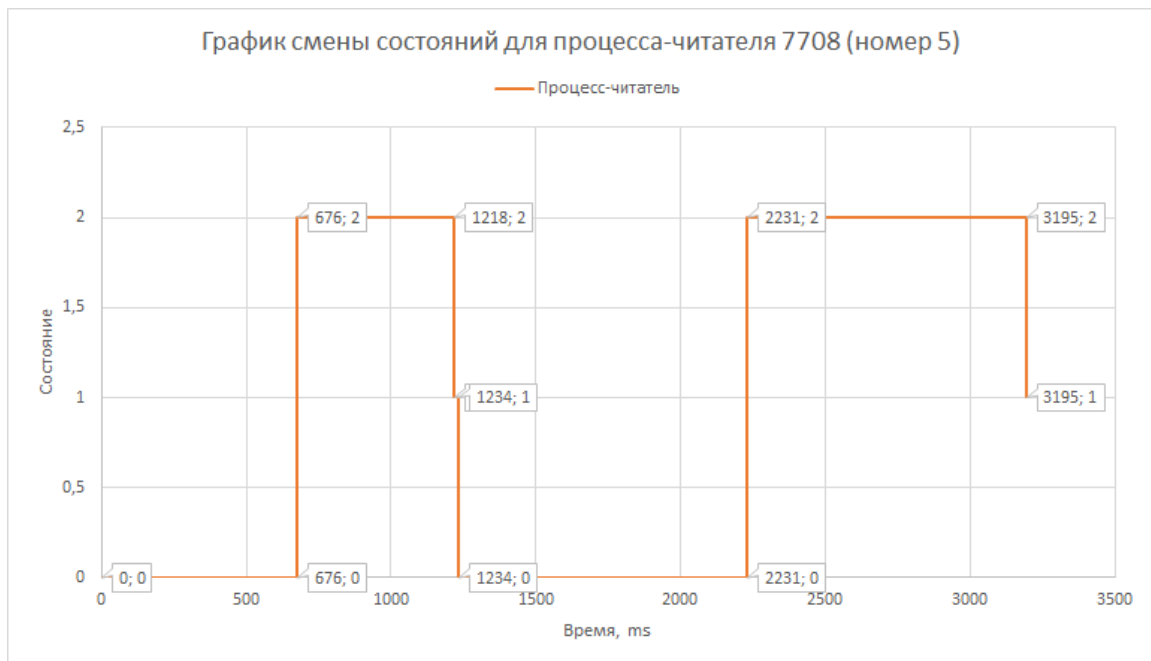


Рисунок 12: График смены состояний для приложения-читателя 5

2.4. График доступности страниц в оперативной памяти

Доступность 5 страниц из оперативной памяти.

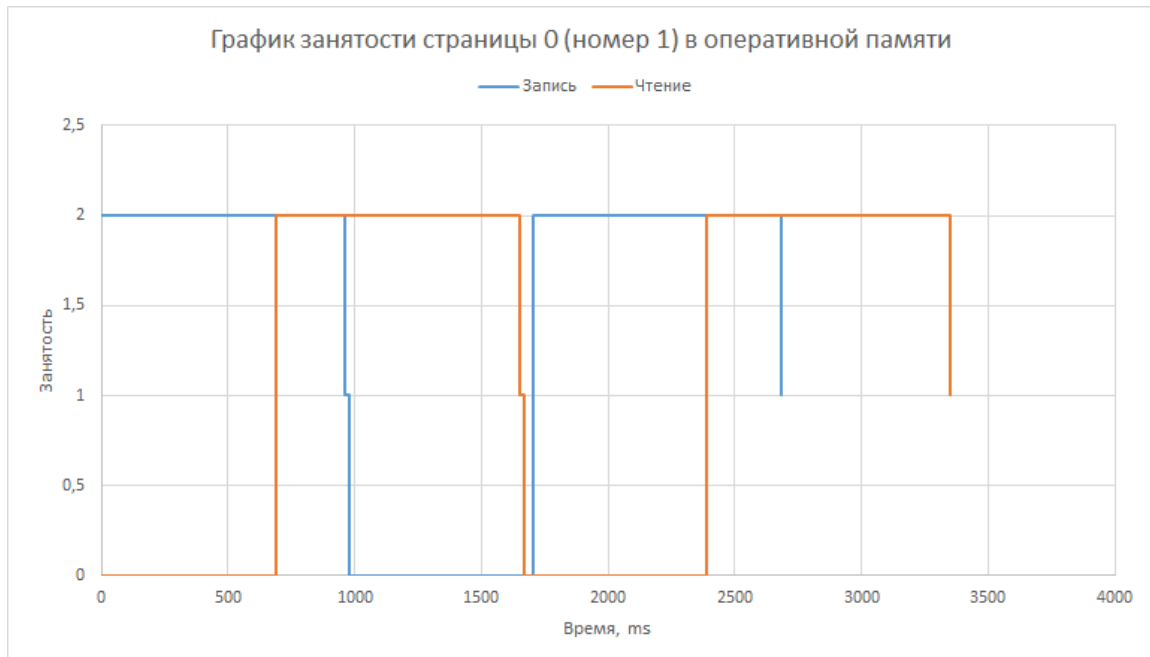


Рисунок 13: График доступности страницы 1 из оперативной памяти

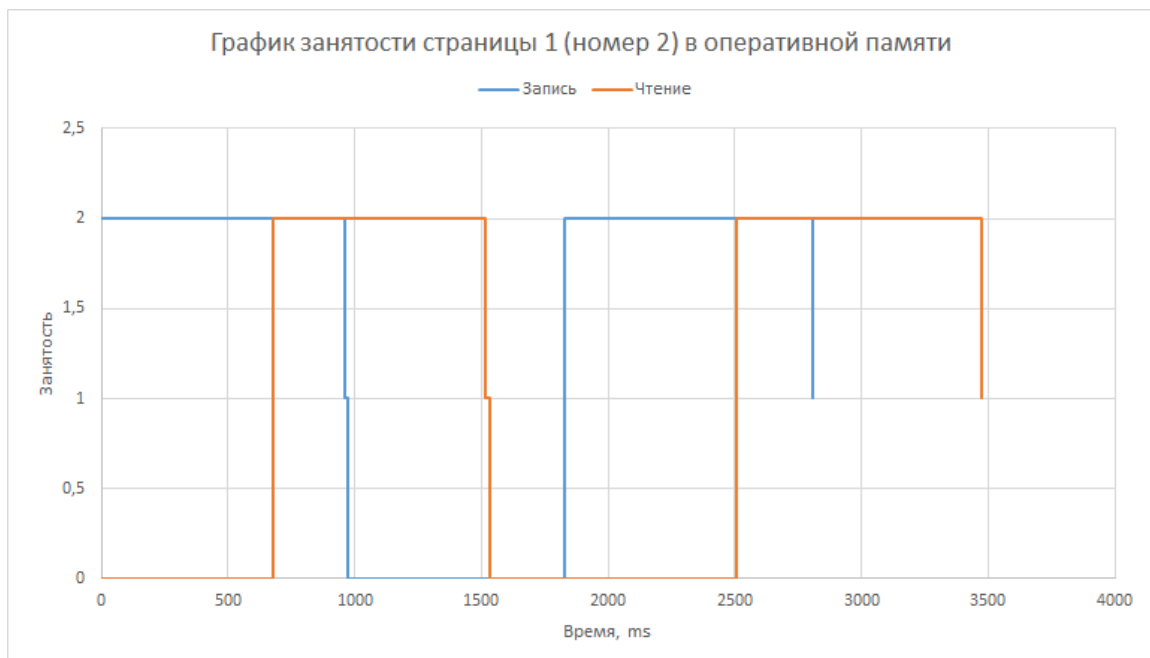


Рисунок 14: График доступности страницы 2 из оперативной памяти

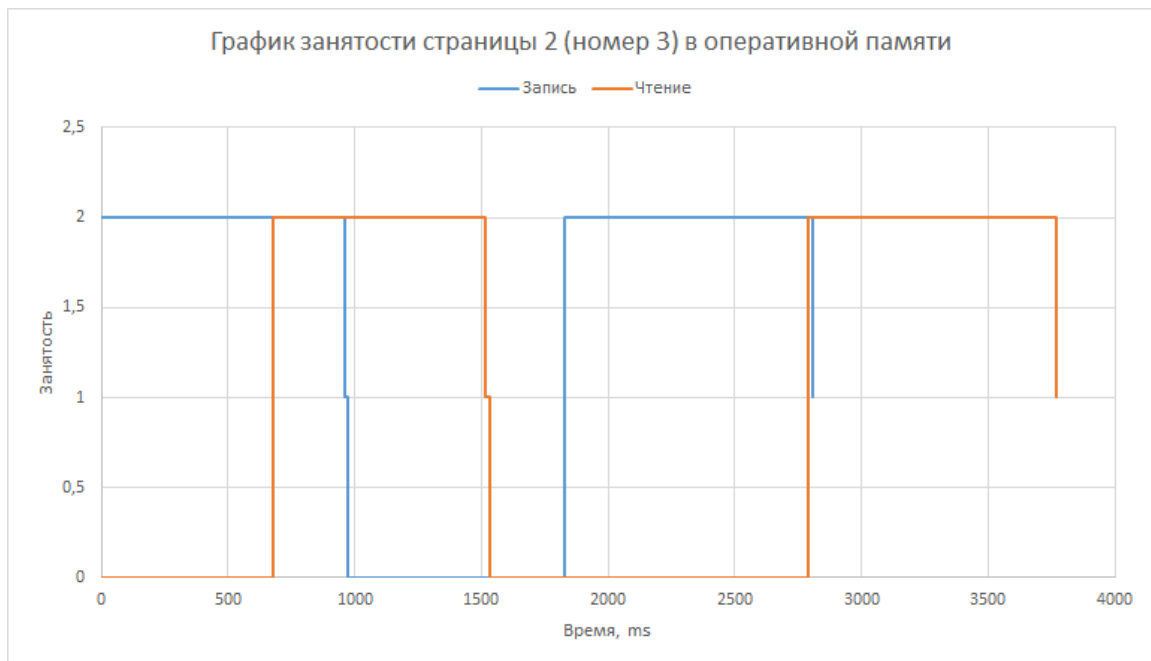


Рисунок 15: График доступности страницы 3 из оперативной памяти

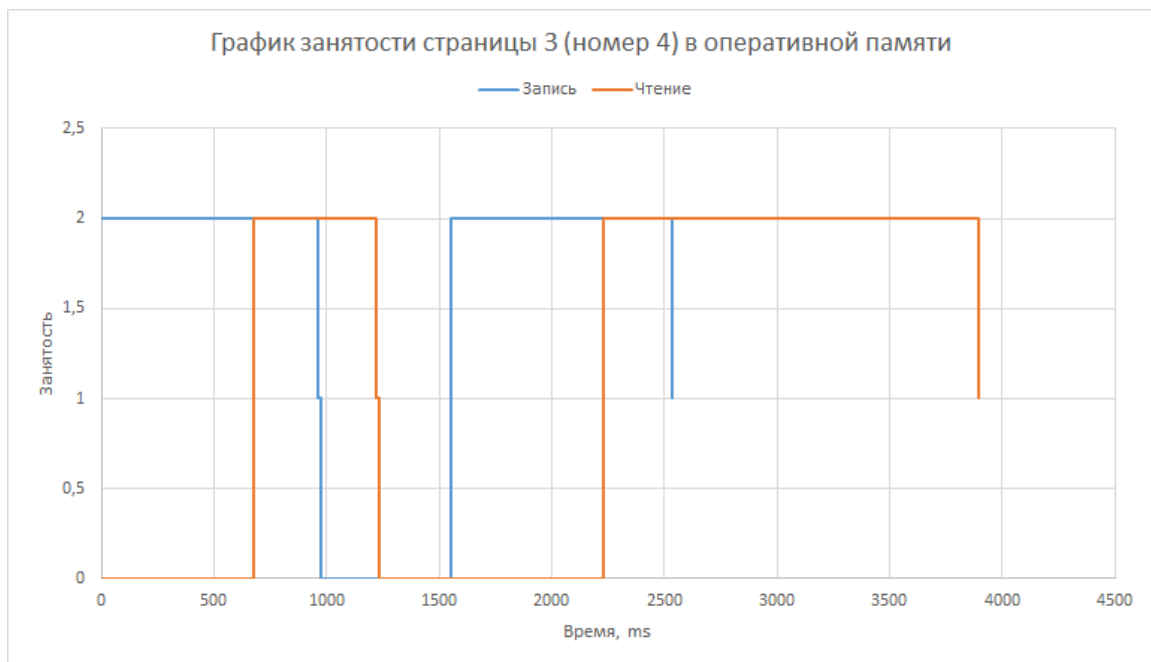


Рисунок 16: График доступности страницы 4 из оперативной памяти

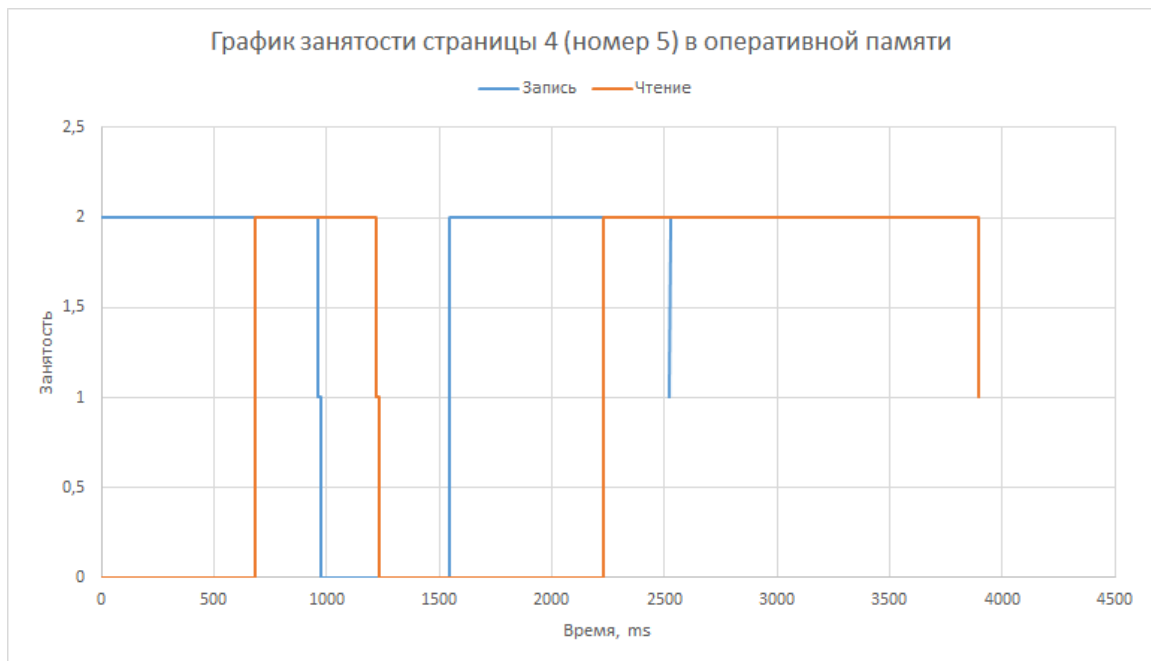


Рисунок 17: График доступности страницы 5 из оперативной памяти

2.5. Журнальный файл процессов-писателей и процессов-писателей

Журнальный файл представлен в виде записей
номер/тип/действие/страница/секунды.

WRITER 2924 has been started!

2924 W BW -1 0

2924 W WR 0 0

2924 W RL 0 961

2924 W BW -1 977

2924 W WR 3 1551

2924 W RL 3 2533

WRITER 2924 has been ended!

WRITER 7728 has been started!

7728 W BW -1 0

7728 W WR 1 0

7728 W RL 1 958

7728 W BW -1 974

7728 W WR 4 1547

7728 W RL 4 2529

WRITER 7728 has been ended!

WRITER 8804 has been started!

8804 W BW -1 0

8804 W WR 2 0

8804 W RL 2 959

8804 W BW -1 975

8804 W WR 5 1548

8804 W RL 5 2530

WRITER 8804 has been ended!

WRITER 424 has been started!

424 W BW -1 0

424 W WR 3 0

424 W RL 3 959

424 W BW -1 975

424 W WR 6 1548

424 W RL 6 2529

WRITER 424 has been ended!

WRITER 8500 has been started!

8500 W BW -1 0

8500 W WR 4 0

8500 W RL 4 960

8500 W BW -1 976

8500 W WR 7 1549

8500 W RL 7 2530

WRITER 8500 has been ended!

WRITER 4280 has been started!

4280 W BW -1 0

4280 W WR 5 0

4280 W RL 5 959

4280 W BW -1 974

4280 W WR 8 1547

4280 W RL 8 2529

WRITER 4280 has been ended!

WRITER 1864 has been started!

1864 W BW -1 0

1864 W WR 6 0

1864 W RL 6 951

1864 W BW -1 967

1864 W WR 9 1540

1864 W RL 9 2522

WRITER 1864 has been ended!

WRITER 1332 has been started!

1332 W BW -1 0

1332 W WR 7 0

1332 W RL 7 960

1332 W BW -1 975

1332 W WR 10 1549

1332 W RL 10 2531

WRITER 1332 has been ended!

WRITER 10992 has been started!

10992 W BW -1 0

10992 W WR 8 0

10992 W RL 8 960

10992 W BW -1 976

10992 W WR 11 1549

10992 W RL 11 2530

WRITER 10992 has been ended!

WRITER 5660 has been started!

5660 W BW -1 0

5660 W WR 9 0

5660 W RL 9 960

5660 W BW -1 975

5660 W WR 12 1548

5660 W RL 12 2530

WRITER 5660 has been ended!

WRITER 9436 has been started!

9436 W BW -1 0

9436 W WR 10 0

9436 W RL 10 960

9436 W BW -1 975

9436 W WR 13 1549

9436 W RL 13 2529

WRITER 9436 has been ended!

WRITER 10580 has been started!

10580 W BW -1 0

10580 W WR 11 0

10580 W RL 11 961

10580 W BW -1 977

10580 W WR 14 1550

10580 W RL 14 2530

WRITER 10580 has been ended!

WRITER 2416 has been started!

2416 W BW -1 1

2416 W WR 12 1

2416 W RL 12 962

2416 W BW -1 977

2416 W WR 15 1550

2416 W RL 15 2531

WRITER 2416 has been ended!

WRITER 8956 has been started!

8956 W BW -1 0

8956 W WR 13 0

8956 W RL 13 961

8956 W BW -1 977

8956 W WR 16 1550

8956 W RL 16 2531

WRITER 8956 has been ended!

WRITER 1932 has been started!

1932 W BW -1 0

1932 W WR 14 0

1932 W RL 14 959

1932 W BW -1 974

1932 W WR 0 1701

1932 W RL 0 2684

WRITER 1932 has been ended!

WRITER 10272 has been started!

10272 W BW -1 0

10272 W WR 15 0

10272 W RL 15 960

10272 W BW -1 976

10272 W WR 1 1828

10272 W RL 1 2808

WRITER 10272 has been ended!

WRITER 6992 has been started!

6992 W BW -1 0

6992 W WR 16 0

6992 W RL 16 959

6992 W BW -1 974

6992 W WR 2 1826

6992 W RL 2 2806

WRITER 6992 has been ended!

reader 2516 started.

2516 R BW -1 0

2516 R RD 1 677

2516 R RL 1 1221

2516 R BW -1 1236

2516 R RD 3 2232

2516 R RL 3 3196

READER 2516 has been ended.

reader 6008 started.

6008 R BW -1 0

6008 R RD 2 676

6008 R RL 2 1220

6008 R BW -1 1235

6008 R RD 4 2231

6008 R RL 4 3195

READER 6008 has been ended.

reader 6156 started.

6156 R BW -1 2

6156 R RD 3 677

6156 R RL 3 1220

6156 R BW -1 1235

6156 R RD 5 2232

6156 R RL 5 3196

READER 6156 has been ended.

reader 4444 started.

4444 R BW -1 1

4444 R RD 4 679

4444 R RL 4 1220

4444 R BW -1 1236

4444 R RD 6 2233

4444 R RL 6 3197

READER 4444 has been ended.

reader 7708 started.

7708 R BW -1 0

7708 R RD 5 676

7708 R RL 5 1218

7708 R BW -1 1234

7708 R RD 7 2231

7708 R RL 7 3195

READER 7708 has been ended.

reader 2496 started.

2496 R BW -1 0

2496 R RD 6 678

2496 R RL 6 1220

2496 R BW -1 1236

2496 R RD 8 2233

2496 R RL 8 3196

READER 2496 has been ended.

reader 10848 started.

10848 R BW -1 0

10848 R RD 7 676

10848 R RL 7 1218

10848 R BW -1 1233

10848 R RD 9 2231

10848 R RL 9 3194

READER 10848 has been ended.

reader 4212 started.

4212 R BW -1 0

4212 R RD 8 677

4212 R RL 8 1219

4212 R BW -1 1234

4212 R RD 10 2232

4212 R RL 10 3195

READER 4212 has been ended.

reader 8464 started.

8464 R BW -1 0

8464 R RD 9 676

8464 R RL 9 1217

8464 R BW -1 1233

8464 R RD 11 2230

8464 R RL 11 3194

READER 8464 has been ended.

reader 8480 started.

8480 R BW -1 0

8480 R RD 10 675

8480 R RL 10 1217

8480 R BW -1 1233

8480 R RD 12 2230

8480 R RL 12 3193

READER 8480 has been ended.

reader 10564 started.

10564 R BW -1 1

10564 R RD 11 677

10564 R RL 11 1218

10564 R BW -1 1234

10564 R RD 13 2231

10564 R RL 13 3194

READER 10564 has been ended.

reader 11080 started.

11080 R BW -1 0

11080 R RD 12 677

11080 R RL 12 1218

11080 R BW -1 1234

11080 R RD 14 2230

11080 R RL 14 3193

READER 11080 has been ended.

reader 10332 started.

10332 R BW -1 0

10332 R RD 13 677

10332 R RL 13 1219

10332 R BW -1 1235

10332 R RD 15 2231

10332 R RL 15 3194

READER 10332 has been ended.

reader 7704 started.

7704 R BW -1 0

7704 R RD 14 678

7704 R RL 14 1220

7704 R BW -1 1235

7704 R RD 16 2232

7704 R RL 16 3195

READER 7704 has been ended.

reader 8744 started.

8744 R BW -1 0

8744 R RD 15 677

8744 R RL 15 1219

8744 R BW -1 1234

8744 R RD 0 2386

8744 R RL 0 3350

READER 8744 has been ended.

reader 11040 started.

11040 R BW -1 0

11040 R RD 16 677

11040 R RL 16 1219

11040 R BW -1 1235

11040 R RD 1 2509

11040 R RL 1 3473

READER 11040 has been ended.

WRITER 11128 has been started!

11128 W BW -1 0

11128 W WR 0 690

11128 W RL 0 1653

11128 W BW -1 1669

11128 W WR 2 2789

11128 W RL 2 3769

WRITER 11128 has been ended!

reader 6272 started.

6272 R BW -1 0

6272 R RD 1 971

6272 R RL 1 1514

6272 R BW -1 1530

6272 R RD 3 2946

6272 R RL 3 3897

READER 6272 has been ended.

reader 9016 started.

9016 R BW -1 1

9016 R RD 2 970

9016 R RL 2 1514

9016 R BW -1 1530

9016 R RD 4 2946

9016 R RL 4 3897

READER 9016 has been ended.

2.6. Исходный код программы «Диспетчер»

/*

Program:

Saint-Petersburg ETU OS laboratory work 4 part 1

Author:

Matvey Sobolev, 2021

Compiler:

g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0

Copyright (C) 2018 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

*/

```
#include <windows.h>
```

```
#include <string>
```

```
#include <iostream>
```

```
#include <malloc.h>
```

```
using namespace std;
```

```
// ----- CONSTANTS DECLARATION -----
```

```
const size_t PAGE_NUMBER = 9 + 3 + 0 + 8 + 2 + 4 - 9;
```

```
const size_t READER_NUMBER = 18;
```

```
const size_t WRITER_NUMBER = 18;
```

```

const string FILE_NAME("basicfile");
const string MAP_NAME("mappingfile");
//const string FILE_LOG_NAME("basiclogfile");
//const string MAP_LOG_NAME("mappinglogfile");
const string LOG_MUTEX_NAME("logmutexfile");
const string IO_MUTEX_NAME("iomutexfile");

// ----- MAIN -----

int main()
{
    size_t localCounter = 0;
    string localBuffer;
    HANDLE hProcesses[READER_NUMBER + WRITER_NUMBER];
    HANDLE hThreads[READER_NUMBER + WRITER_NUMBER];

    // Getting page size

    SYSTEM_INFO temporarySystemInfo; // temporary item
    GetSystemInfo(&temporarySystemInfo); // getting system info
    const DWORD PAGE_SIZE = temporarySystemInfo.dwPageSize; // page size getting from
system info

    // Creating base file, which will be mapped

    HANDLE hBasicFile = CreateFileA(FILE_NAME.c_str(), GENERIC_WRITE |
GENERIC_READ, 0, NULL, CREATE_ALWAYS, 0, NULL);
    if (hBasicFile == INVALID_HANDLE_VALUE) // check, if it's wrong value
    {
        cout << "Basic file hasn't been created (something wrong). Last error code: " << GetLastError()
<< ".\n";
        return GetLastError();
    }

```

```

}
else // successful creation
{
    cout << "Basic file has been created successfully!\n";
}

SetFilePointer(hBasicFile, PAGE_SIZE * PAGE_NUMBER, 0, FILE_BEGIN);
SetEndOfFile(hBasicFile);

// Making mapped file from basic file

HANDLE hMappedFile = CreateFileMappingA(hBasicFile, NULL, PAGE_READWRITE, 0, 0,
MAP_NAME.c_str());
if (hMappedFile == NULL) // check, if it's wrong value
{
    cout << "Hasn't been made mapped file from basic file (something wrong). Last error code: "
<< GetLastError() << ".\n";
    CloseHandle(hBasicFile); // closing successfully created file handle (see previous if/else)
    return GetLastError();
}
else if (GetLastError() == ERROR_ALREADY_EXISTS) // check, if it's "existing" value
{
    cout << "Hasn't been made mapped file from basic file (already exists). Last error code: " <<
GetLastError() << ".\n";
    CloseHandle(hBasicFile); // closing successfully created file handle (see previous if/else)
    return GetLastError();
}
else // successful making
{
    cout << "Has been made mapped file from basic file successfully!\n";
}

```

```

/*
// [MATLAB] Creating base file, which will be mapped

HANDLE hBasicLogFile = CreateFileA(FILE_LOG_NAME.c_str(), GENERIC_WRITE |
GENERIC_READ, 0, NULL, CREATE_ALWAYS, 0, NULL);
if (hBasicLogFile == INVALID_HANDLE_VALUE) // check, if it's wrong value
{
    cout << "Basic file hasn't been created (something wrong). Last error code: " << GetLastError()
<< ".\n";
    return GetLastError();
}
else // successful creation
{
    cout << "Basic file has been created successfully!\n";
}

SetFilePointer(hBasicLogFile, PAGE_SIZE * PAGE_NUMBER, 0, FILE_BEGIN);
SetEndOfFile(hBasicLogFile);

// [MATLAB] Making mapped file from basic file

HANDLE hMappedLogFile = CreateFileMappingA(hBasicLogFile, NULL,
PAGE_READWRITE, 0, 0, MAP_LOG_NAME.c_str());
if (hMappedLogFile == NULL) // check, if it's wrong value
{
    cout << "Hasn't been made mapped file from basic file (something wrong). Last error code: "
<< GetLastError() << ".\n";
    CloseHandle(hBasicLogFile); // closing successfully created file handle (see previous if/else)
    return GetLastError();
}
else if (GetLastError() == ERROR_ALREADY_EXISTS) // check, if it's "existing" value
{

```

```

        cout << "Hasn't been maked mapped file from basic file (already exists). Last error code: " <<
GetLastError() << ".\n";
        CloseHandle(hBasicLogFile); // closing successfully created file handle (see previous if/else)
        return GetLastError();
    }
    else // successful making
    {
        cout << "Has been maked mapped file from basic file successfully!\n";
    }
    */

// Creating mutex for logging

HANDLE mLogFile = CreateMutexA(NULL, FALSE, LOG_MUTEX_NAME.c_str());
if (mLogFile == NULL) // check, if it's wrong value
{
    cout << "Hasn't been created mutex for logging file (something wrong). Last error code: " <<
GetLastError() << ".\n";
    return GetLastError();
}
else if (GetLastError() == ERROR_ALREADY_EXISTS) // check, if it's "existing" value
{
    cout << "Hasn't been created mutex for logging file (already exists). Last error code: " <<
GetLastError() << ".\n";
    return GetLastError();
}
else
{
    cout << "Has been created mutex for logging file successfullt!\n";
}

// Creating mutexes for all of the pages

```



```

HANDLE mInputOutput[PAGE_NUMBER];
for (int i = 0; i < PAGE_NUMBER; i++)
{
    localBuffer = IO_MUTEX_NAME + std::to_string(i);
    mInputOutput[i] = CreateMutexA(NULL, FALSE, localBuffer.c_str());
    if (mInputOutput[i] == NULL)
    {
        cout << "Hasn't been created mutex for input/output (something wrong). Last error code: "
<< GetLastError() << ".\n";
        return GetLastError();
    }
    else if (GetLastError() == ERROR_ALREADY_EXISTS) // check, if it's "existing" value
    {
        cout << "Hasn't been created mutex for input/output (already exists). Last error code: " <<
GetLastError() << ".\n";
        return GetLastError();
    }
}

// Starting writer processes

for (int i = 0; i < READER_NUMBER; i++, localCounter++)
{
    STARTUPINFOA localDestination;
    SecureZeroMemory(&localDestination, sizeof(STARTUPINFOA)); // "ZeroMemory" macro
is unsecured!!!
    localDestination.cb = sizeof(STARTUPINFOA);
    PROCESS_INFORMATION localInformation;

    WINBOOL localResult = CreateProcessA("writer.exe", NULL, NULL, NULL, FALSE, 0,
NULL, NULL, &localDestination, &localInformation);

```

```

    if (localResult == true)
    {
        cout << "WRITER " << i << " has been started.\n";
    }
    else
    {
        cout << "Problem with creating writer number " << i << " process. Last error number: " <<
GetLastError() << ".\n";
        return GetLastError();
    }

    hProcesses[localCounter] = localInformation.hProcess;
    hThreads[localCounter] = localInformation.hThread;
    Sleep(2);
}

// Starting reader processes

for (int i = 0; i < WRITER_NUMBER; i++, localCounter++)
{
    STARTUPINFOA localDestination;
    SecureZeroMemory(&localDestination, sizeof(STARTUPINFOA)); // "ZeroMemory" macro
is unsecured!!!
    localDestination.cb = sizeof(STARTUPINFOA);
    PROCESS_INFORMATION localInformation;

    WINBOOL localResult = CreateProcessA("reader.exe", NULL, NULL, NULL, FALSE, 0,
NULL, NULL, &localDestination, &localInformation);
    if (localResult == true)
    {
        cout << "READER " << i << " has been started.\n";
    }
}

```

```

else
{
    cout << "Problem with creating reader number " << i << " process. Last error number: " <<
GetLastError() << ".\n";
    return GetLastError();
}

hProcesses[localCounter] = localInformation.hProcess;
hThreads[localCounter] = localInformation.hThread;
Sleep(2);
}

// Waitig for all process finish

WaitForMultipleObjects(READER_NUMBER + WRITER_NUMBER, hProcesses, TRUE,
INFINITE);

cout << "All process finished.\n";

// Cleaning and freeing

for (int i = 0; i < READER_NUMBER + WRITER_NUMBER; i++)
{
    CloseHandle(hProcesses[i]);
    CloseHandle(hThreads[i]);
}
for (int i = 0; i < PAGE_NUMBER; i++)
{
    CloseHandle(mInputOutput[i]);
}
CloseHandle(mLogFile);
CloseHandle(hMappedFile);

```

```
CloseHandle(hBasicFile);

// End

cout << "End.\n";

return 0;
}
```

2.7. Исходный код программы «Писатель»

/*

Program:

Saint-Petersburg ETU OS laboratory work 4 part 1

Author:

Matvey Sobolev, 2021

Compiler:

g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0

Copyright (C) 2018 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

*/

```
#include <windows.h>
```

```
#include <string>
```

```
#include <iostream>
```

```
#include <ctime>
```

```
#include <cstdlib>
```

```
#include <fstream>
```

```
#include <sstream>
```

```
#include <time.h>
```

```
using namespace std;
```

```
const size_t PAGE_NUMBER = 9 + 3 + 0 + 8 + 2 + 4 - 9;
```

```

const size_t WRITE_TIMES_NUMBER = 2;
const string LOGFILENAME("logfile.txt");
const string FILE_NAME("basicfile");
const string MAP_NAME("mappingfile");
const string LOG_MUTEX_NAME("logmutexfile");
const string IO_MUTEX_NAME("iomutexfile");

// The class for the logging all the time

class LogFile
{
private:
    ostringstream *localStream;
    fstream file; // file itself
    //time_t startTime; // iocnt1
    clock_t startTime; // starting time
    string fileName; // filename
public:
    LogFile(string fileName);
    ~LogFile();
    size_t getTime();
    void log(string localMessage);
    void log(int type, size_t id, long long pageNum, bool isRead, int what);
    void flush();
};

int main()
{
    // Initializing

    const size_t maxCharSize = 256;
    size_t processID = GetCurrentProcessId();

```

```

size_t processPage;
size_t pause;
string localBuffer;
LogFile logFile(LOGFILENAME);

// Getting page size

SYSTEM_INFO temporarySystemInfo; // temporary item
GetSystemInfo(&temporarySystemInfo); // getting system info
const DWORD PAGE_SIZE = temporarySystemInfo.dwPageSize; // page size getting from
system info

// Opening mapped file

HANDLE    hMappedFile    =    OpenFileMappingA(FILE_MAP_ALL_ACCESS,    false,
MAP_NAME.c_str());
if (hMappedFile == NULL) // error check
{
    logFile.log("WRITER: Something wrong with opening mapped file. Last error code: " +
std::to_string(GetLastError()));
    logFile.flush();
    return GetLastError();
}

// Opening mutex for the log file

HANDLE    mLogFile    =    OpenMutexA(MUTEX_ALL_ACCESS,    false,
LOG_MUTEX_NAME.c_str());
if (mLogFile == NULL) // error check
{
    logFile.log("WRITER: Something wrong with opening mutex for logging file. Last error code:
" + std::to_string(GetLastError()));

```

```

    logFile.flush();
    return GetLastError();
}

// Getting all mutexes for the input and output

HANDLE mInputOutput[PAGE_NUMBER];
for (size_t i = 0; i < PAGE_NUMBER; i++)
{
    localBuffer = IO_MUTEX_NAME + std::to_string(i);

    mInputOutput[i] = OpenMutexA(MUTEX_ALL_ACCESS, false, localBuffer.c_str());
    if(mInputOutput[i] == NULL) // error check
    {
        logFile.log("WRITER: Something wrong with opening mutex for input/output number " +
std::to_string(i) + ". Last error code: " + std::to_string(GetLastError()));
        logFile.flush();
        return GetLastError();
    }
}

// View mapping file

void* aMappedFile = MapViewOfFile(hMappedFile, FILE_MAP_WRITE, 0, 0, 0); // address of
map view of file
if (aMappedFile == NULL) // error check
{
    logFile.log("WRITER: Something wrong with viewing mapped file. Last error code: " +
std::to_string(GetLastError()));
    logFile.flush();
    return GetLastError();
}

```



```

// Logging

logFile.log("WRITER " + std::to_string(processID) + " has been started!");

// Blocking pages in RAM with VirtualLock

VirtualLock(aMappedFile, PAGE_SIZE * PAGE_NUMBER);
for(size_t gi = 0; gi < WRITE_TIMES_NUMBER; gi++)
{
    // Page choosing

    //processPage = rand() % PAGE_NUMBER; // rand
    processPage = -1; // 1st free

    // LogFile starting

    logFile.log(1, processID, processPage, false, -1);

    // Catching mutex

    //WaitForSingleObject(mInputOutput[processPage], INFINITE); // rand
    processPage = WaitForMultipleObjects(PAGE_NUMBER, mInputOutput, FALSE, INFINITE);
// 1st free

    // Choosing random place

    unsigned mem_src = rand() % 256;
    *((unsigned*)((char*)aMappedFile + PAGE_SIZE*processPage)) = mem_src;

    // LogFile accessing or changing

```

```

logFile.log(2, processID, processPage, false, mem_src);
pause = (rand() % 1001) + 500;
Sleep((DWORD)pause);

// LogFile releasing

logFile.log(3, processID, processPage, false, -1);

// Releasing mutex

ReleaseMutex(mInputOutput[processPage]);

// Waiting

Sleep(10);
}

// All logs commonly writing in the file (AND start synchronizing)

WaitForSingleObject(mLogFile, INFINITE);
logFile.log("WRITER " + std::to_string(processID) + " has been ended!");
logFile.flush(); // synchronizing the buffer
ReleaseMutex(mLogFile);

// ... AND end synchronizing

// Unlocking page sizes

VirtualUnlock(aMappedFile, PAGE_SIZE * PAGE_NUMBER);

// Cleaning and freeing

```

```

for (int i = 0; i < PAGE_NUMBER; i++)
{
    CloseHandle(mInputOutput[i]);
}
CloseHandle(mLogFile);
CloseHandle(hMappedFile);
UnmapViewOfFile(aMappedFile);

return 0;
}

size_t LogFile::getTime() // getting time method
{
    size_t milisecFromStart = 0;
    //time_t endTime; // iocnt1
    //time(&endTime); // iocnt1
    //milisecFromStart = (size_t)(difftime(endTime, startTime) * 1000 + 0.5); // iocnt1
    clock_t endTime = clock();
    milisecFromStart = (size_t)((double)(endTime - startTime) / CLOCKS_PER_SEC) * 1000 +
0.5);
    return milisecFromStart;
}

void LogFile::log(int type, size_t id, long long pageNum, bool isRead, int what = -1) // isRead true -
- reader, false -- writer
{
    string localState;
    string localType;
    string ID = std::to_string(id);
    switch (type)
    {
        case 1:

```

```

        localState = "BW";
        break;
case 2:
    if (isRead == true)
    {
        localState = "RD";
    }
    else
    {
        localState = "WR";
    }
    break;
case 3:
    localState = "RL";
    break;
default:
    localState = "FL";
    break;
}

if (isRead == true)
{
    localType = "R";
}
else
{
    localType = "W";
}

size_t milisecFromStart = getTime(); // checking the time
string time = std::to_string(milisecFromStart); // translating the time to the string
//string page = pageNum == -1 ? string("the first one released") : std::to_string (pageNum);

```

```

string page = pageNum == -1 ? string("-1") : std::to_string(pageNum);
string swhat = what == -1 ? "" : " byte " + std::to_string(what) + (isRead == true ? " from" : " to");
//log(localType + " " + ID + " " + localState + swhat + " page " + page + " (time = " + time + "
ms). ");
log(ID + " " + localType + " " + localState + " " + page + " " + time);
}

```

```

void LogFile::log (string localMessage) // logging the message
{
    (*localStream) << localMessage << "\n";
}

```

```

void LogFile::flush () // flushing the message
{
    file << (*localStream).str() << "\n"; // write the message
    file.flush(); // flushing
    (*localStream).str(""); // setting the buffer to "null"
}

```

```

LogFile::LogFile (string fileName) // constructor
{
    this->fileName = fileName;
    localStream = new ostringstream();
    file.open(fileName, std::fstream::app | std::fstream::out);
    //time(&startTime); // iocnt1
    startTime = clock();
}

```

```

LogFile::~~LogFile () // destructor
{
    delete localStream;
    file.close();
}

```

}

2.8. Исходный код программы «Читатель»

/*

Program:

Saint-Petersburg ETU OS laboratory work 4 part 1

Author:

Matvey Sobolev, 2021

Compiler:

g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0

Copyright (C) 2018 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

*/

```
#include <windows.h>
```

```
#include <string>
```

```
#include <iostream>
```

```
#include <ctime>
```

```
#include <cstdlib>
```

```
#include <fstream>
```

```
#include <sstream>
```

```
#include <time.h>
```

```
#include <list>
```

```
using namespace std;
```

```

const size_t PAGE_NUMBER = 9 + 3 + 0 + 8 + 2 + 4 - 9;
const size_t READ_TIMES_NUMBER = 2;
const string LOGFILENAME("logfile.txt");
const string FILE_NAME("basicfile");
const string MAP_NAME("mappingfile");
const string LOG_MUTEX_NAME("logmutexfile");
const string IO_MUTEX_NAME("iomutexfile");

// The class for the logging all the time

class LogFile
{
private:
    ostringstream *localStream;
    fstream file; // file itself
    //ostringstream *localStream2;
    //fstream file2;
    //time_t startTime; // iocnt1
    clock_t startTime; // starting time
    string fileName; // filename
    //string fileName2;
public:
    LogFile(string fileName/*, string fileName2*/);
    ~LogFile();
    size_t getTime();
    void log(string localMessage);
    //void log2(string localMessage2);
    void log(int type, size_t id, long long pageNum, bool isRead, int what);
    void flush();
    //void flush2();
};

```



```

int main()
{
    // Initializing

    const size_t maxCharSize = 256;
    size_t processID = GetCurrentProcessId();
    size_t processPage;
    size_t pause;
    string localBuffer;
    LogFile logFile(LOGFILENAME);

    // Getting page size

    SYSTEM_INFO temporarySystemInfo; // temporary item
    GetSystemInfo(&temporarySystemInfo); // getting system info
    const DWORD PAGE_SIZE = temporarySystemInfo.dwPageSize; // page size getting from
system info

    // Opening mapped file

    HANDLE    hMappedFile    =    OpenFileMappingA(FILE_MAP_ALL_ACCESS,    false,
MAP_NAME.c_str());
    if (hMappedFile == NULL) // error check
    {
        logFile.log("READER: Something wrong with opening mapped file. Last error code: " +
std::to_string(GetLastError()));
        logFile.flush();
        return GetLastError();
    }

    // Opening mutex for the log file

```

```

HANDLE mLogFile = OpenMutexA(MUTEX_ALL_ACCESS, false,
LOG_MUTEX_NAME.c_str());
if (mLogFile == NULL) // error check
{
    logFile.log("READER: Something wrong with opening mutex for logging file. Last error code:
" + std::to_string(GetLastError()));
    logFile.flush();
    return GetLastError();
}

// Getting all mutexes for the input and output

HANDLE mInputOutput[PAGE_NUMBER];
for (size_t i = 0; i < PAGE_NUMBER; ++i)
{
    localBuffer = IO_MUTEX_NAME + std::to_string(i);

    mInputOutput[i] = OpenMutexA(MUTEX_ALL_ACCESS, false, localBuffer.c_str());
    if (mInputOutput[i] == NULL) // error check
    {
        logFile.log("READER: Something wrong with opening mutex for input/output number " +
std::to_string(i) + ". Error: " + std::to_string(GetLastError()));
        logFile.flush();
        return GetLastError();
    }
}

// View mapping file

void* aMappedFile = MapViewOfFile(hMappedFile, FILE_MAP_READ, 0, 0, 0); // address of
map view of file
if (aMappedFile == NULL) // error check

```

```

{
    logFile.log("READER: Something wrong with viewing mapped file. Last error code: " +
std::to_string(GetLastError()));
    logFile.flush();
    return GetLastError();
}

/*
// [MATLAB] Opening mapped file

HANDLE hMappedLogFile = OpenFileMappingA(FILE_MAP_ALL_ACCESS, false,
MAP_NAME.c_str());
if (hMappedLogFile == NULL) // error check
{
    logFile.log2("READER: Something wrong with opening mapped file. Last error code: " +
std::to_string(GetLastError()));
    logFile.flush2();
    return GetLastError();
}

// [MATLAB] View mapping file

void* aMappedLogFile = MapViewOfFile(hMappedLogFile, FILE_MAP_READ, 0, 0, 0); //
address of map view of file
if (aMappedLogFile == NULL) // error check
{
    logFile.log2("READER: Something wrong with viewing mapped file. Last error code: " +
std::to_string(GetLastError()));
    logFile.flush2();
    return GetLastError();
}
*/

```

```

// Logging

logFile.log("reader " + std::to_string(processID) + " started. ");

// Blocking pages in RAM with VirtualLock

VirtualLock(aMappedFile, PAGE_SIZE * PAGE_NUMBER);
for(size_t gi = 0; gi < READ_TIMES_NUMBER; gi++)
{
    // Page choosing

    //processPage = rand() % PAGE_NUMBER; // rand
    processPage = -1; // 1st free

    //LogFile starting

    logFile.log(1, processID, processPage, true, -1);

    // Catching mutex

    //WaitForSingleObject(mInputOutput[processPage], INFINITE); // rand
    processPage = WaitForMultipleObjects(PAGE_NUMBER, mInputOutput, FALSE, INFINITE);
// 1st free

    // Choosing random place

    unsigned mem_src = *((unsigned*)((char*)aMappedFile + PAGE_SIZE*processPage));

    // LogFile accessing or changing

    logFile.log(2, processID, processPage, true, mem_src);

```

```

    pause = (rand() % 1001) + 500;
    Sleep((DWORD)pause);

    logFile.log(3, processID, processPage, true, -1);

    // Releasing mutex

    ReleaseMutex(mInputOutput[processPage]);

    // Waiting

    Sleep(10);
}

// All logs commonly writing in the file (AND start synchronizing)

WaitForSingleObject(mLogFile, INFINITE);
logFile.log("READER " + std::to_string(processID) + " has been ended.");
logFile.flush(); // synchronizing the buffer
ReleaseMutex(mLogFile);

// ... AND end synchronizing

// Unlocking page sizes

VirtualUnlock(aMappedFile, PAGE_SIZE * PAGE_NUMBER);

// Cleaning and freeing

for (int i = 0; i < PAGE_NUMBER; ++i)
{
    CloseHandle(mInputOutput[i]);
}

```

```

    }
    CloseHandle(mLogFile);
    CloseHandle(hMappedFile);
    UnmapViewOfFile(aMappedFile);

    return 0;
}

size_t LogFile::getTime() // getting time method
{
    size_t milisecFromStart = 0;
    //time_t endTime; // iocnt1
    //time(&endTime); // iocnt1
    //milisecFromStart = (size_t)(difftime(endTime, startTime) * 1000 + 0.5); // iocnt1
    clock_t endTime = clock();
    milisecFromStart = (size_t)((((double)(endTime - startTime) / CLOCKS_PER_SEC) * 1000 +
0.5);
    return milisecFromStart;
}

void LogFile::log(int type, size_t id, long long pageNum, bool isRead, int what = -1) // isRead true -
- reader, false -- writer
{
    string localState;
    string localType;
    string ID = std::to_string(id);
    switch (type)
    {
        case 1:
            localState = "BW";
            break;
        case 2:

```

```

    if (isRead == true)
    {
        localState = "RD";
    }
    else
    {
        localState = "WR";
    }
    break;
case 3:
    localState = "RL";
    break;
default:
    localState = "FL";
    break;
}

```

```

if (isRead == true)
{
    localType = "R";
}
else
{
    localType = "W";
}

```

```

size_t milisecFromStart = getTime(); // checking the time
string time = std::to_string(milisecFromStart); // translating the time to the string
//string page = pageNum == -1 ? string("the first one released") : std::to_string(pageNum);
string page = pageNum == -1 ? string("-1") : std::to_string(pageNum);
string swhat = what == -1 ? "" : " byte " + std::to_string(what) + (isRead == true ? " from" : " to");

```

```

//log(localType + " " + ID + " " + localState + swhat + " page " + page + " (time = " + time + "
ms). ");
log(ID + " " + localType + " " + localState + " " + page + " " + time);
//log2(time);
//flush2();
}

```

```

void LogFile::log (string localMessage) // logging the message
{
    (*localStream) << localMessage << "\n";
}

```

```

/*void LogFile::log2 (string localMessage2) // logging the message
{
    (*localStream2) << localMessage2;
}*/

```

```

void LogFile::flush () // flushing the message
{
    file << (*localStream).str() << "\n"; // write the message
    file.flush(); // flushing
    (*localStream).str(""); // setting the buffer to "null"
}

```

```

/*void LogFile::flush2 ()
{
    file2 << (*localStream2).str() << ", "; // write the message
    file2.flush(); // flushing
    (*localStream2).str(""); // setting the buffer to "null"
}*/

```

```

LogFile::LogFile (string fileName/*, string fileName2*/) // constructor

```



```

{
    this->fileName = fileName;
    localStream = new ostream();
    file.open(fileName, std::fstream::app | std::fstream::out);
    //time(&startTime); // iocnt 1
    //this->fileName2 = fileName2;
    //localStream2 = new ostream();
    //file2.open(fileName2, std::fstream::app | std::fstream::out);
    startTime = clock();
}

```

LogFile::~LogFile () // destructor

```

{
    delete localStream;
    file.close();
    //delete localStream2;
    //file2.close();
}

```

2.9. Вывод

В ходе выполнения первой части («Реализация решения задачи о читателях и писателях») лабораторной работы 4 «Межпроцессорное взаимодействие» была освоена одновременная работа с буферной памятью приложений-читателей и приложений-писателей. Была освоена блокировка страниц в оперативной памяти, изучена технология мьютексов, рассчитано время работы приложений-писателей и приложений-читателей.

Исходя из полученных данных о доступности 5 страниц в оперативной памяти (по аналогии с занятостью читателей и писателей им были выделены 3 состояния и показаны обращения к ним), можно сделать несколько заключений. Во-первых, наложенные обращения к страницам в оперативной памяти от писателей и читателей обусловлены тем, что каждое приложение начинает свою работу и проходит первую итерацию, поэтому происходит такая ситуация. Во-вторых, с момента освобождения мьютекса до записи в журнал может пройти некоторое время, поэтому страница может получить такое «наложение на графиках». В-третьих, в момент смены итераций можно заметить, что к страницам не было обращения (что также подтверждается файлами журнала).

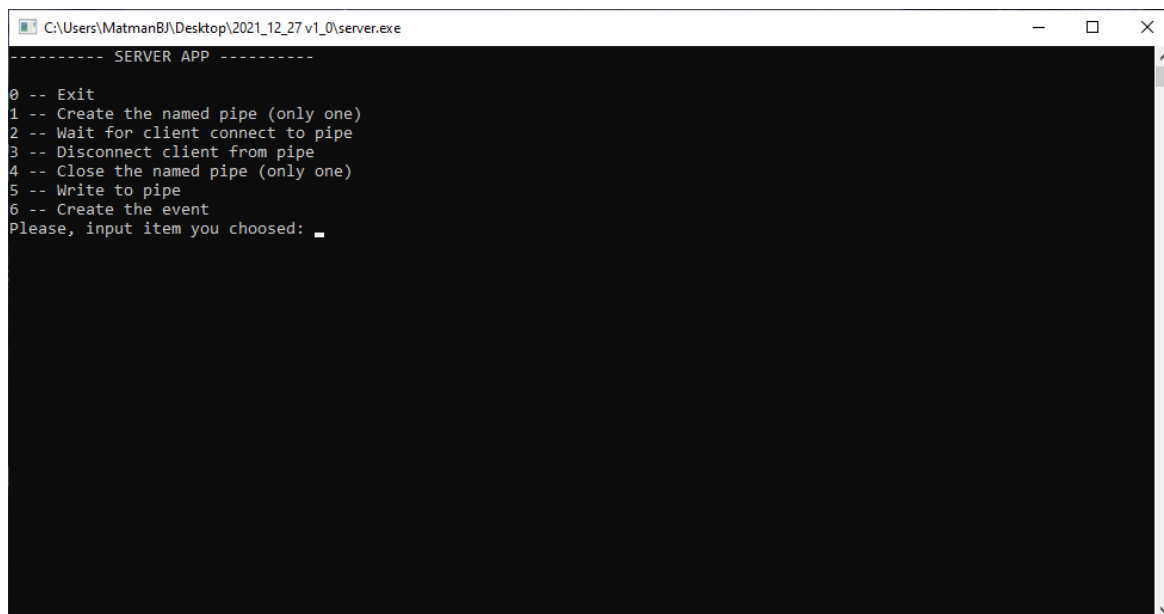
Также, исходя из смены состояний на приложений-читателей и приложений-писателей был сделан вывод о том, что время работы, требовавшееся для тех или иных действий было примерно одинаково, так как приложения занимали первую освободившуюся страницу.

Таким образом и была реализована задача о читателях и писателях.

3. Использование именованных каналов для реализации сетевого межпроцессорного взаимодействия

3.1. Приложения «Сервер» и «Клиент»

Консольные приложения «Сервер» и «Клиент» для работы с именованным каналом.



```
C:\Users\MatmanBJ\Desktop\2021_12_27 v1_0\server.exe
----- SERVER APP -----
0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: _
```

Рисунок 18: Меню сервера

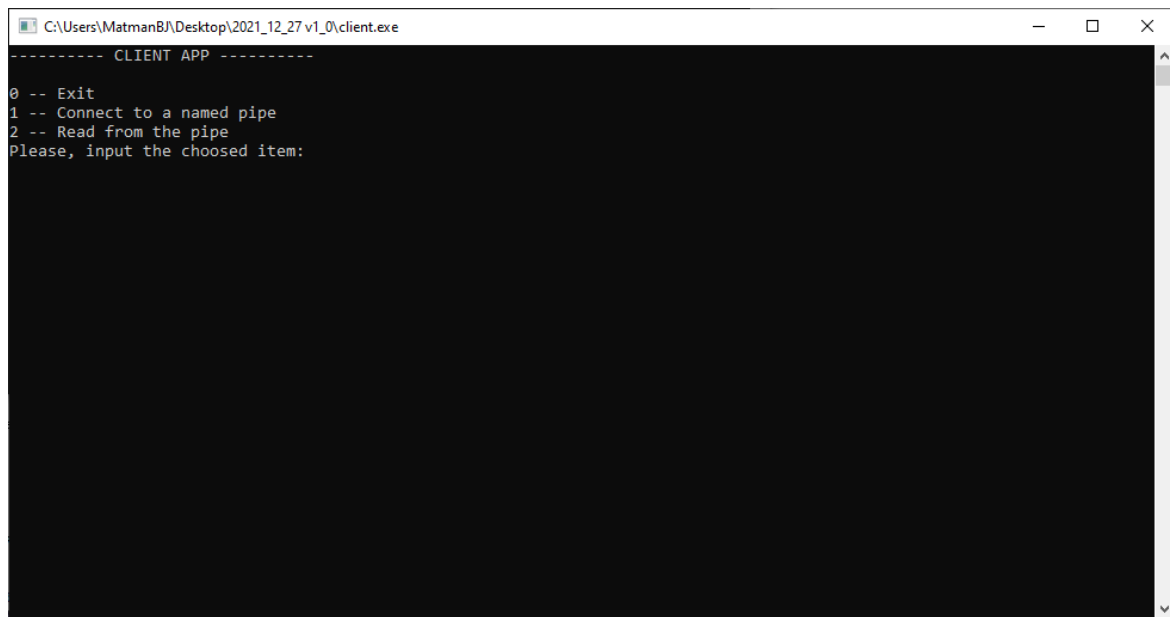


Рисунок 19: Меню клиента

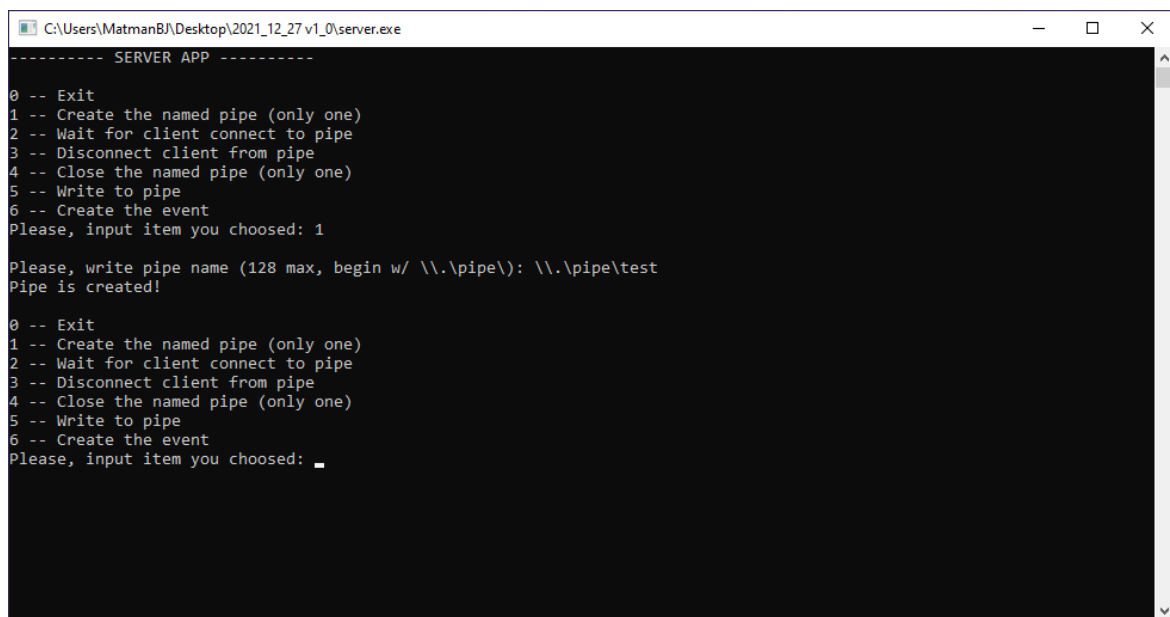


Рисунок 20: Создание именованного канала

```
C:\Users\MatmanBJ\Desktop\2021_12_27 v1_0\server.exe
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 1

Please, write pipe name (128 max, begin w/ \\.\pipe\): \\.\pipe\test
Pipe is created!

0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 6

Event created!

0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed:
```

Рисунок 21: Создание события

```
C:\Users\MatmanBJ\Desktop\2021_12_27 v1_0\server.exe
0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 6

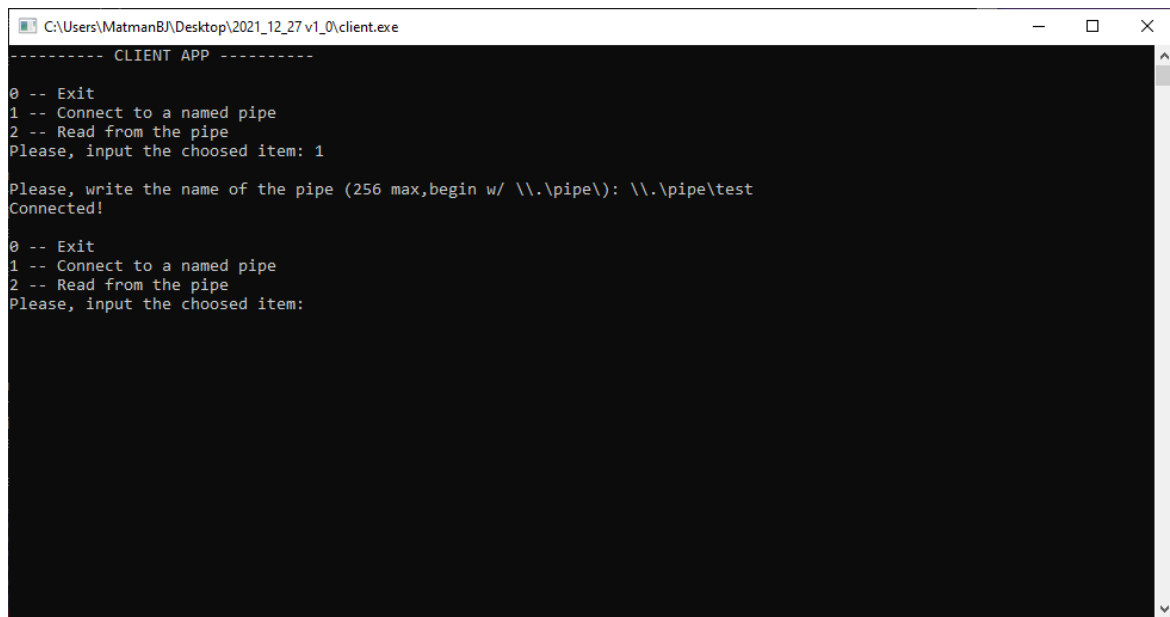
Event created!

0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 2

Connected!

0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed:
```

Рисунок 22: Подключение к именованному каналу со стороны сервера



```
----- CLIENT APP -----
0 -- Exit
1 -- Connect to a named pipe
2 -- Read from the pipe
Please, input the choosed item: 1

Please, write the name of the pipe (256 max,begin w/ \\.\pipe\): \\.\pipe\test
Connected!
```

Рисунок 23: Подключение к именованному каналу со стороны клиента



```
----- CLIENT APP -----
0 -- Exit
1 -- Connect to a named pipe
2 -- Read from the pipe
Please, input the choosed item: 1

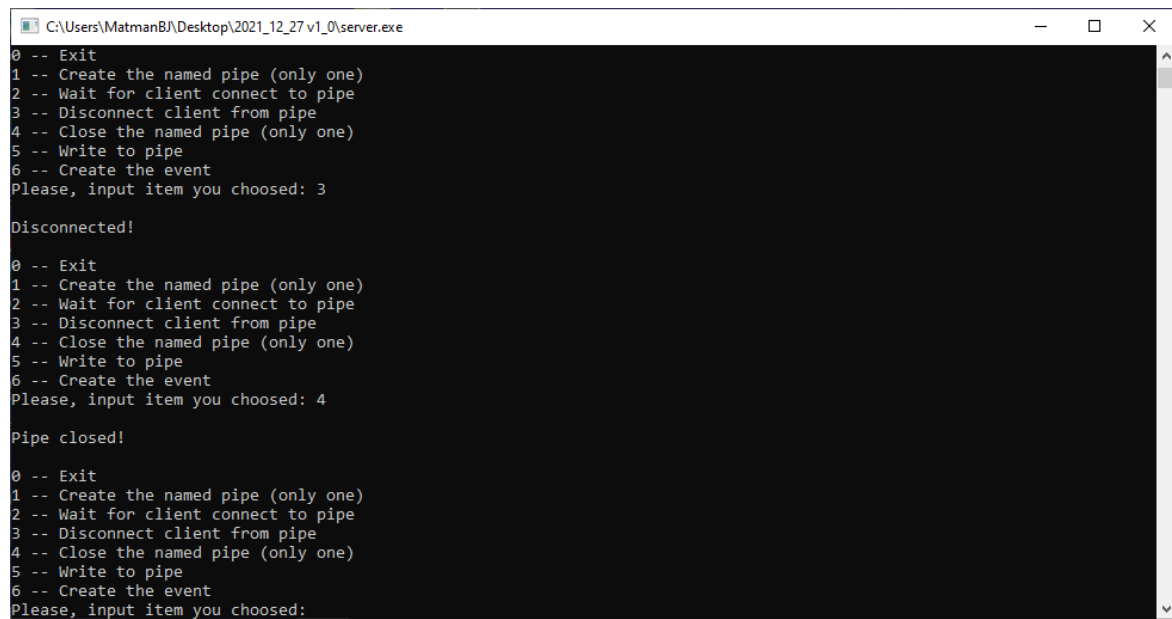
Please, write the name of the pipe (256 max,begin w/ \\.\pipe\): \\.\pipe\test
Connected!

0 -- Exit
1 -- Connect to a named pipe
2 -- Read from the pipe
Please, input the choosed item: 2

Received message:
test

0 -- Exit
1 -- Connect to a named pipe
2 -- Read from the pipe
Please, input the choosed item:
```

Рисунок 24: Отправка сообщения через именованный канал



```
C:\Users\MatmanBJ\Desktop\2021_12_27 v1_0\server.exe
0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 3
Disconnected!
0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 4
Pipe closed!
0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 3
Disconnected!
0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 4
Pipe closed!
0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 3
Disconnected!
0 -- Exit
1 -- Create the named pipe (only one)
2 -- Wait for client connect to pipe
3 -- Disconnect client from pipe
4 -- Close the named pipe (only one)
5 -- Write to pipe
6 -- Create the event
Please, input item you choosed: 4
Pipe closed!
```

Рисунок 25: Отключение именованного канала от сервера и его закрытие

3.2. Исходный код программы «Сервер»

/*

Program:

Saint-Petersburg ETU OS laboratory work 4 part 2

Author:

Matvey Sobolev, 2021

Compiler:

g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0

Copyright (C) 2018 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

*/

```
#include <iostream>
```

```
#include <string>
```

```
#include <windows.h>
```

```
using namespace std;
```

```
const size_t maxBufferSize = 256;
```

```
int main()
```

```
{
```

```
    int menuChoose = 1; // choosed value
```

```
    HANDLE hPipe = INVALID_HANDLE_VALUE; // default value
```



```

HANDLE hEvent = NULL; // default value
cout << "----- SERVER APP -----\\n";

while (menuChoose != 0)
{
    menuChoose = -1;
    while (menuChoose < 0 || menuChoose > 7)
    {
        cout << "\\n";
        cout << "0 -- Exit\\n";
        cout << "1 -- Create the named pipe (only one)\\n";
        cout << "2 -- Wait for client connect to pipe\\n";
        cout << "3 -- Disconnect client from pipe\\n";
        cout << "4 -- Close the named pipe (only one)\\n";
        cout << "5 -- Write to pipe\\n";
        cout << "6 -- Create the event\\n";
        cout << "Please, input item you choosed: ";
        cin >> menuChoose;
        if (menuChoose < 0 || menuChoose > 7)
        {
            cout << "Something wrong! Please, try again!\\n";
        }
        else
        {
            cout << "\\n";
        }
    }
    switch (menuChoose)
    {
        case 1: // Create the named pipe (only one)
            if (hPipe == INVALID_HANDLE_VALUE) // error check
            {

```

```

char localPipeName[maxBufferSize];
cout << "Please, write pipe name (128 max, begin w/ \\\\.\\pipe\\): ";
cin >> localPipeName;
hPipe    =    CreateNamedPipe(localPipeName,    PIPE_ACCESS_OUTBOUND,
PIPE_TYPE_BYTE | PIPE_WAIT,
    PIPE_UNLIMITED_INSTANCES, maxBufferSize, 0UL, 0UL, NULL);
if (hPipe != INVALID_HANDLE_VALUE)
{
    cout << "Pipe is created!\n";
}
else
{
    cout << "Pipe is not created! Last error code: " << GetLastError() << ".\n";
}
}
else
{
    cout << "Pipe is created already!\n";
}
break;
case 2: // Wait for client connect to pipe
    if (hPipe == INVALID_HANDLE_VALUE)
    {
        cout << "No pipe.\n";
    }
    else if (ConnectNamedPipe(hPipe, NULL))
    {
        cout << "Connected!\n";
    }
    else
    {
        cout << "No connection! Last error code is " << GetLastError() << "\n";
    }
}

```

```

    }
    break;
case 3: // Disconnect client from pipe
    if (hPipe == INVALID_HANDLE_VALUE)
    {
        cout << "No pipe.\n";
    }
    else if (DisconnectNamedPipe(hPipe))
    {
        cout << "Disconnected!\n";
    }
    else
    {
        cout << "No connection! Last error code is " << GetLastError() << "\n";
    }
    break;
case 4: // Close the named pipe (only one)
    if (hPipe != INVALID_HANDLE_VALUE)
    {
        DisconnectNamedPipe(hPipe);
        CloseHandle(hPipe);
        hPipe = INVALID_HANDLE_VALUE;
        cout << "Pipe closed!\n";
    }
    break;
case 5: // Write to pipe
    if (hPipe == INVALID_HANDLE_VALUE)
    {
        cout << "No pipe.\n";
    }
    else if (hEvent == NULL)
    {

```

```

        cout << "No event.\n";
    }
    else
    {
        bool localSuccess = false; // start -- false
        OVERLAPPED overlapped; // creating overlapped structure
        overlapped.Offset = 0UL; // offset
        overlapped.OffsetHigh = 0UL; // offset high
        overlapped.hEvent = hEvent; // event set
        string localMessage;
        cout << "Please, input message (" << maxBufferSize << " max):\n";
        fflush(stdin);
        cin >> localMessage;
        localSuccess = WriteFile(hPipe, localMessage.c_str(), maxBufferSize, NULL,
&overlapped);
        WaitForSingleObject(hEvent, INFINITE);

        if(localSuccess == true)
        {
            cout << "Message in pipe!\n";
        }
        else
        {
            cout << "Message is not in pipe! Last error code: " << GetLastError() << ".\n";
        }
    }
    break;
case 6: // Create the event
    if (hEvent == NULL)
    {
        hEvent = CreateEvent(NULL, false, false, NULL);
        if(hEvent != NULL)

```

```

        {
            cout << "Event created!\n";
        }
        else
        {
            cout << "Event isn't created! Last error code is " << GetLastError() << "\n";
        }
    }
    else
    {
        cout << "Pipe is already created.\n";
    }
    break;
}
}

// Cleaning and freeing

if (hPipe != INVALID_HANDLE_VALUE)
{
    CloseHandle(hPipe);
}
if (hEvent != NULL)
{
    CloseHandle(hEvent);
}

// Return

return 0;
}

```


3.3. Исходный код программы «Клиент»

/*

Program:

Saint-Petersburg ETU OS laboratory work 4 part 2

Author:

Matvey Sobolev, 2021

Compiler:

g++ (x86_64-posix-seh-rev0, Built by MinGW-W64 project) 8.1.0

Copyright (C) 2018 Free Software Foundation, Inc.

This is free software; see the source for copying conditions. There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

*/

```
#include <iostream>
```

```
#include <string>
```

```
#include <windows.h>
```

```
using namespace std;
```

```
const size_t maxBufferSize = 256;
```

```
char localMessage[maxBufferSize];
```

```
void endRoutine (DWORD dwE, DWORD dwBT, LPOVERLAPPED lpO);
```

```
int main()
```

```

{
    int menuChoose = 1;
    HANDLE hPipe = INVALID_HANDLE_VALUE;
    cout << "----- CLIENT APP -----\\n";

    while (menuChoose != 0)
    {
        menuChoose = -1;
        while (menuChoose < 0 || menuChoose > 2)
        {
            cout << "\\n";
            cout << "0 -- Exit\\n";
            cout << "1 -- Connect to a named pipe\\n";
            cout << "2 -- Read from the pipe\\n";
            cout << "Please, input the choosed item: ";
            cin >> menuChoose;
            if (menuChoose < 0 || menuChoose > 2)
            {
                cout << "Please, try again!\\n";
            }
            else
            {
                cout << "\\n";
            }
        }
        char localPipeName[maxBufferSize];
        //char localMessage[maxBufferSize];
        switch(menuChoose)
        {
            case 1:
                if(hPipe != INVALID_HANDLE_VALUE)
                {

```



```

        CloseHandle(hPipe);
    }
    cout << "Please, write the name of the pipe (" << maxBufferSize << " max,begin w/
\\\\.\\pipe\\): ";
    cin >> localPipeName;
    hPipe = CreateFile(localPipeName, GENERIC_READ, 0UL, NULL, OPEN_EXISTING,
FILE_ATTRIBUTE_READONLY | FILE_FLAG_OVERLAPPED, NULL);
    if(hPipe != INVALID_HANDLE_VALUE)
    {
        cout << "Connected!\n";
    }
    else
    {
        cout << "Not connected! Error code is " << GetLastError() << "\n";
    }
    break;
case 2:
    OVERLAPPED overlapped;
    overlapped.Offset = 0UL;
    overlapped.OffsetHigh = 0UL;
        cout << "Waiting for message ... [if no from server, the program will
wait]\n";
        ReadFileEx(hPipe, localMessage, sizeof(localMessage), &overlapped,
&endRoutine); // handle / place for message / size of the message / end function
        SleepEx(INFINITE, true);
        break;
    }
}

// Cleaning and freeing

if (hPipe != INVALID_HANDLE_VALUE)

```

```
{  
    CloseHandle(hPipe);  
}  
return 0;  
}
```

```
void endRoutine (DWORD dwE, DWORD dwBT, LPOVERLAPPED lpO)  
{  
    cout << localMessage << "\n";  
    cout << "Message received!\n";  
}
```

3.4. Вывод

В ходе выполнения второй части («Использование именованных каналов для реализации межпроцессорного взаимодействия») лабораторной работы 4 «Межпроцессорное взаимодействие» было освоено создание именованных каналов, создание объектов-событий, была изучена асинхронная запись в именованный канал. Также было реализовано подключение между приложением-сервером и приложением-клиентом.

В ходе работы было проверено установление подключения между приложением-сервером и приложением-клиентом (показано на скриншотах), а также была проверена связь между двумя приложениями, в результате чего сообщение, написанное в первом приложении (на сервере) было успешно передано второму приложению (клиенту) с использованием именованного канала (также показано на скриншотах). Это подтверждает асинхронный ввод/вывод по каналам связи.

Таким образом и были использованы именованные каналы для реализации сетевого межпроцессорного взаимодействия.

4. Список использованных источников

1. Операционные системы: электронные методические указания к лабораторным работам / Сост.: А. В. Тимофеев. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2016.
2. Таненбаум Э. Современные операционные системы. 2-е изд. – СПб.: Питер, 2002. – 1040 с.: ил.
3. Столлингс, Вильям. Операционные системы, 4-е издание. : Пер. с англ. – М. : Издательский дом «Вильямс», 2002. – 848 с. : ил. – Парал. Тит. Англ.
4. Документация «Microsoft» [сайт]. URL: <https://docs.microsoft.com/ru-ru/cpp/parallel/concrt/comparing-the-concurrency-runtime-to-other-concurrency-models?view=msvc-170>.
5. Курс «Операционные системы» в образовательной онлайн-системе Google Класс [сайт]. URL: <https://classroom.google.com/c/Mzg3ODc4NDE5MDU4>.