Министерство науки и образования РФ
Федеральное государственное автономное образовательное
учреждение высшего профессионального образования
«Санкт-Петербургский государственный электротехнический
университет «ЛЭТИ» им. В. И. Ульянова (Ленина)»
(СПбГЭТУ «ЛЭТИ»)

Кафедра вычислительной техники

Отчёт
по лабораторной работе № 6
на тему:
“Межпроцессное взаимодействие в Windows”

по дисциплине “Операционные системы”

Выполнил студент гр. 4306:
Табаков А.В.
Принял: Тимофеев А.В.

Санкт-Петербург
2016

Цель работы: исследовать управление файловой системой с помощью Win32 API.

Задание 6.1. Реализация решения задачи о читателях-писателях.

Процессы (по 8 читателей и писателей)

```
PROCESS ffffa8006c02060
    SessionId: 1  Cid: 2538    Peb: 7efdf000  ParentCid: 0364
    DirBase: 172d0000  ObjectTable: fffff8a00a8a57d0  HandleCount:  60.
    Image: Manager.exe
----------------------------------------------------------------
PROCESS fffffa8009c45b10
    SessionId: 1  Cid: 2234    Peb: 7fffffdc000  ParentCid: 2538
    DirBase: 36c62000  ObjectTable: 00000000  HandleCount:   0.
    Image: Writer.exe

PROCESS fffffa8009bc3b10
    SessionId: 1  Cid: 1704    Peb: 7fffffd7000  ParentCid: 2538
    DirBase: 19b8ac000  ObjectTable: 00000000  HandleCount:   0.
    Image: Writer.exe

PROCESS fffffa800ac18410
    SessionId: 1  Cid: 20d0    Peb: 7fffffd8000  ParentCid: 2538
    DirBase: 1bff74000  ObjectTable: 00000000  HandleCount:   0.
    Image: Writer.exe

PROCESS fffffa800a160990
    SessionId: 1  Cid: 1a9c    Peb: 7fffffd5000  ParentCid: 2538
    DirBase: 14a399000  ObjectTable: 00000000  HandleCount:   0.
    Image: Writer.exe

PROCESS fffffa8008a03720
    SessionId: 1  Cid: 1088    Peb: 7fffffd4000  ParentCid: 2538
    DirBase: 2ee5e000  ObjectTable: 00000000  HandleCount:   0.
    Image: Writer.exe

PROCESS fffffa800aa64060
    SessionId: 1  Cid: 0954    Peb: 7fffffd5000  ParentCid: 2538
    DirBase: 43b23000  ObjectTable: 00000000  HandleCount:   0.
    Image: Writer.exe

PROCESS fffffa800a0de7c0
    SessionId: 1  Cid: 24a0    Peb: 7fffffdc000  ParentCid: 2538
    DirBase: 1f0fe8000  ObjectTable: 00000000  HandleCount:   0.
    Image: Writer.exe

PROCESS fffffa8009613400
    SessionId: 1  Cid: 27cc    Peb: 7fffffdf000  ParentCid: 2538
    DirBase: 2f2ed000  ObjectTable: 00000000  HandleCount:   0.
    Image: Writer.exe
--------------------------------------------------------------------------------
PROCESS fffffa8009ce31f0
    SessionId: 1  Cid: 1f8c    Peb: 7fffffdf000  ParentCid: 2538
    DirBase: 106852000  ObjectTable: 00000000  HandleCount:   0.
    Image: Reader.exe

PROCESS fffffa800a6229c0
    SessionId: 1  Cid: 1a08    Peb: 7fffffd3000  ParentCid: 2538
    DirBase: 3a737000  ObjectTable: 00000000  HandleCount:   0.
    Image: Reader.exe

PROCESS fffffa800a32a420
    SessionId: 1  Cid: 19fc    Peb: 7fffffdd000  ParentCid: 2538
    DirBase: 4b0bc000  ObjectTable: 00000000  HandleCount:   0.
    Image: Reader.exe

PROCESS fffffa8006db3b10
    SessionId: 1  Cid: 0c0c    Peb: 7fffffdf000  ParentCid: 2538
    DirBase: 1438e1000  ObjectTable: 00000000  HandleCount:   0.
    Image: Reader.exe

PROCESS fffffa800a78b060
```

```
   SessionId: 1  Cid: 2410    Peb: 7fffffdc000  ParentCid: 2538
   DirBase: 33146000  ObjectTable: 00000000  HandleCount:   0.
   Image: Reader.exe

PROCESS ffffa8009422b10
   SessionId: 1  Cid: 1d94    Peb: 7fffffda000  ParentCid: 2538
   DirBase: 1f0b6b000  ObjectTable: 00000000  HandleCount:   0.
   Image: Reader.exe

PROCESS ffffa800a5aa060
   SessionId: 1  Cid: 2480    Peb: 7fffffdf000  ParentCid: 2538
   DirBase: 1c6370000  ObjectTable: 00000000  HandleCount:   0.
   Image: Reader.exe

PROCESS ffffa8009be11c0
   SessionId: 1  Cid: 2678    Peb: 7fffffdf000  ParentCid: 2538
   DirBase: 1f0415000  ObjectTable: 00000000  HandleCount:   0.
   Image: Reader.exe
```

Все процессы читателей и писателей являются дочерними процесса Manager.exe

```
0034: Object: fffffa80095f1750  GrantedAccess: 001f0003 Entry: fffff8a009d9a0d0
Object: fffffa80095f1750  Type: (fffffa8006108080) Semaphore
   ObjectHeader: fffffa80095f1720 (new version)
      HandleCount: 1  PointerCount: 2
      Directory Object: fffff8a00106dae0  Name: Writer semaphore

0038: Object: fffffa8006e5afe0  GrantedAccess: 001f0003 Entry: fffff8a009d9a0e0
Object: fffffa8006e5afe0  Type: (fffffa8006108080) Semaphore
   ObjectHeader: fffffa8006e5afb0 (new version)
      HandleCount: 1  PointerCount: 2
      Directory Object: fffff8a00106dae0  Name: Reader semaphore

003c: Object: fffffa80064525f0  GrantedAccess: 001f0001 Entry: fffff8a009d9a0f0
Object: fffffa80064525f0  Type: (fffffa800614a8c0) Mutant
   ObjectHeader: fffffa80064525c0 (new version)
      HandleCount: 1  PointerCount: 2
      Directory Object: fffff8a00106dae0  Name: mutexNum0

0040: Object: fffffa8007f2afc0  GrantedAccess: 001f0001 Entry: fffff8a009d9a100
Object: fffffa8007f2afc0  Type: (fffffa800614a8c0) Mutant
   ObjectHeader: fffffa8007f2af90 (new version)
      HandleCount: 1  PointerCount: 2
      Directory Object: fffff8a00106dae0  Name: mutexNum1
…..
0058: Object: fffff8a002749080  GrantedAccess: 000f0007 Entry: fffff8a009d9a160
Object: fffff8a002749080  Type: (fffffa80061178f0) Section
   ObjectHeader: fffff8a002749050 (new version)
      HandleCount: 1  PointerCount: 2
      Directory Object: fffff8a00106dae0  Name: myMap6Work

005c: Object: fffff8a017f5e970  GrantedAccess: 00000001 Entry: fffff8a009d9a170
Object: fffff8a017f5e970  Type: (fffffa8006121500) Key
   ObjectHeader: fffff8a017f5e940 (new version)
      HandleCount: 1  PointerCount: 1
      Directory Object: 00000000  Name: \REGISTRY\MACHINE\SYSTEM\CONTROLSET00
\CONTROL\SESSION MANAGER

0060: Object: fffffa8008847060  GrantedAccess: 001fffff Entry: fffff8a009d9a180
Object: fffffa8008847060  Type: (fffffa80060f8c90) Thread
   ObjectHeader: fffffa8008847030 (new version)
      HandleCount: 1  PointerCount: 1

0064: Object: fffffa8009c45b10  GrantedAccess: 001fffff Entry: fffff8a009d9a190
Object: fffffa8009c45b10  Type: (fffffa80060f8de0) Process
   ObjectHeader: fffffa8009c45ae0 (new version)
      HandleCount: 1  PointerCount: 2
…..
```

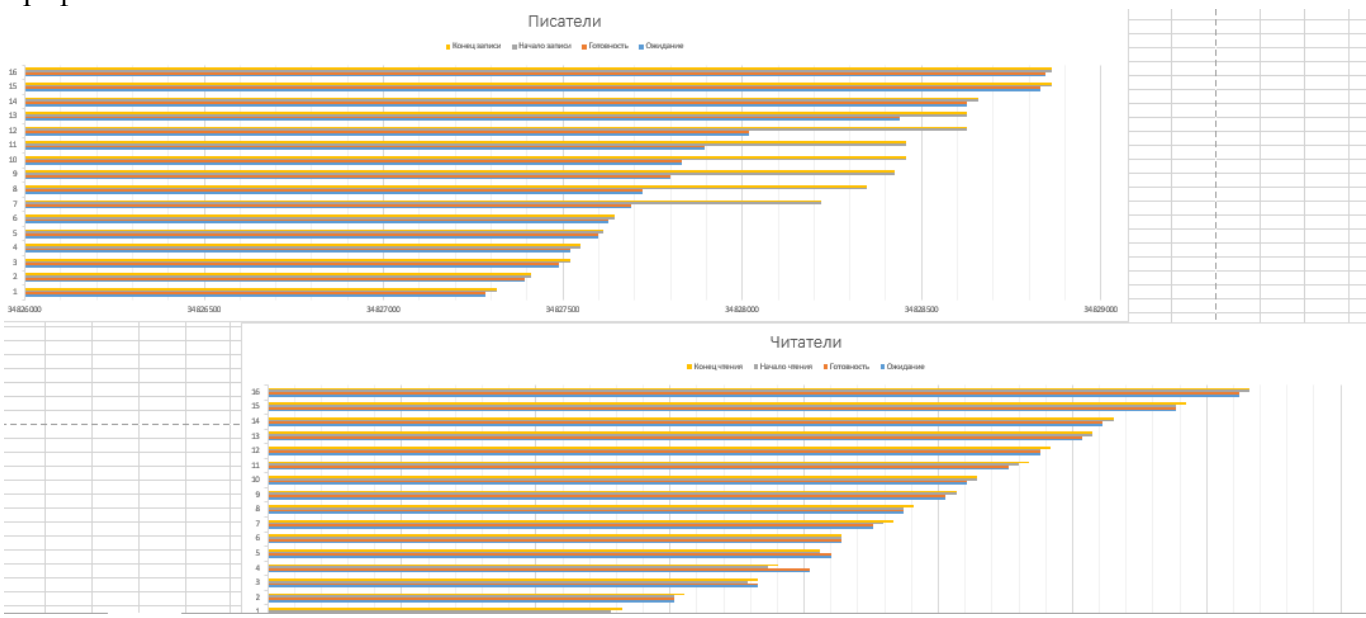
Log:

```
34827285 writer: wait semaphore
34827285 writer: ready
34827316 writer: start writing
34827316 writer: writing completed. page 4. waiting
34827519 writer: wait semaphore
34827519 writer: ready
34827550 writer: start writing
34827550 writer: writing completed. page 0. waiting
34827394 writer: wait semaphore
34827394 writer: ready
34827410 writer: start writing
34827410 writer: writing completed. page 2. waiting
34827628 writer: wait semaphore
34827628 writer: ready
34827644 writer: start writing
34827644 writer: writing completed. page 3. waiting
34828206 reader: wait semaphore
34828206 reader: ready
34828206 reader: start reading
34828221 reader: reading completed. page 0 : this is page #0. writing time = 34827535. waiting
34828440 reader: wait semaphore
34828440 reader: ready
34828455 reader: start reading
34828455 reader: reading completed. page 1 : this is page #1. writing time = 34827613. waiting
34828299 reader: wait semaphore
34828299 reader: ready
34828315 reader: start reading
34828330 reader: reading completed. page 4 : this is page #4. writing time = 34828206. waiting
34828549 reader: wait semaphore
34828549 reader: ready
34828549 reader: start reading
34828564 reader: reading completed. page 5 : this is page #5. writing time = 34828440. waiting
34827488 writer: wait semaphore
34827488 writer: ready
34827519 writer: start writing
34827519 writer: writing completed. page 5. waiting
34827722 writer: wait semaphore
34827722 writer: ready
34827738 writer: empty page not found, waiting.
34828034 writer: empty page not found, waiting.
34828346 writer: start writing
34828346 writer: writing completed. page 0. waiting
34828096 reader: wait semaphore
34828096 reader: ready
34828112 reader: start reading
34828128 reader: reading completed. page 4 : this is page #4. writing time = 34827301. waiting
34828330 reader: wait semaphore
34828330 reader: ready
34828346 reader: start reading
34828362 reader: reading completed. page 5 : this is page #5. writing time = 34827504. waiting
34828408 reader: wait semaphore
34828408 reader: ready
34828424 reader: start reading
34828424 reader: reading completed. page 0 : this is page #0. writing time = 34828346. waiting
34828642 reader: wait semaphore
34828642 reader: ready
34828658 reader: start reading
34828658 reader: reading completed. page 0 : this is page #0. writing time = 34828440. waiting
34827597 writer: wait semaphore
34827597 writer: ready
34827613 writer: start writing
34827613 writer: writing completed. page 1. waiting
```

34827831 writer: wait semaphore
34827831 writer: ready
34827847 writer: empty page not found, waiting.
34828143 writer: empty page not found, waiting.
34828455 writer: start writing
34828455 writer: writing completed. page 5. waiting
34827894 writer: wait semaphore
34827894 writer: ready
34827909 writer: empty page not found, waiting.
34828221 writer: start writing
34828221 writer: writing completed. page 4. waiting
34828440 writer: wait semaphore
34828440 writer: ready
34828455 writer: start writing
34828455 writer: writing completed. page 0. waiting
34828502 reader: wait semaphore
34828502 reader: ready
34828518 reader: start reading
34828533 reader: reading completed. page 4 : this is page #4. writing time = 34828408. waiting
34828752 reader: wait semaphore
34828752 reader: ready
34828752 reader: start reading
34828767 reader: reading completed. page 4 : this is page #4. writing time = 34828627. waiting
34828611 reader: wait semaphore
34828611 reader: ready
34828627 reader: start reading
34828627 reader: reading completed. page 2 : this is page #2. writing time = 34827394. waiting
34828845 reader: wait semaphore
34828845 reader: ready
34828861 reader: start reading
34828861 reader: reading completed. page 3 : this is page #3. writing time = 34827644. waiting
34827800 writer: wait semaphore
34827800 writer: ready
34827800 writer: empty page not found, waiting.
34828112 writer: empty page not found, waiting.
34828424 writer: start writing
34828424 writer: writing completed. page 4. waiting
34828627 writer: wait semaphore
34828627 writer: ready
34828658 writer: start writing
34828658 writer: writing completed. page 2. waiting
34828705 reader: wait semaphore
34828705 reader: ready
34828720 reader: start reading
34828736 reader: reading completed. page 1 : this is page #1. writing time = 34828611. waiting
34828954 reader: wait semaphore
34828954 reader: ready
34828954 reader: start reading
34828970 reader: reading completed. page 0 : this is page #0. writing time = 34828845. waiting
34828814 reader: wait semaphore
34828814 reader: ready
34828830 reader: start reading
34828830 reader: reading completed. page 2 : this is page #2. writing time = 34828642. waiting
34829048 reader: wait semaphore
34829048 reader: ready
34829064 reader: start reading
34829064 reader: reading completed. page 1 : this is page #1. writing time = 34828861. waiting
34827691 writer: wait semaphore
34827691 writer: ready
34827706 writer: empty page not found, waiting.
34828003 writer: empty page not found, waiting.
34828315 writer: empty page not found, waiting.
34828627 writer: start writing
34828627 writer: writing completed. page 1. waiting
34828830 writer: wait semaphore

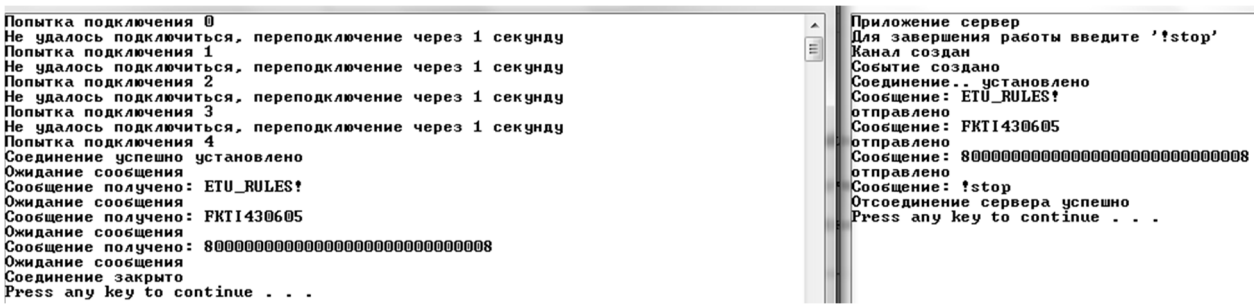
34828830 writer: ready
34828861 writer: start writing
34828861 writer: writing completed. page 0. waiting
34828018 writer: wait semaphore
34828018 writer: ready
34828018 writer: empty page not found, waiting.
34828330 writer: empty page not found, waiting.
34828627 writer: start writing
34828627 writer: writing completed. page 4. waiting
34828845 writer: wait semaphore
34828845 writer: ready
34828861 writer: start writing
34828861 writer: writing completed. page 1. waiting

График смены событий



Вывод: в один момент времени выполняется только один процесс, когда один процесс заканчивает читать или писать, то процессорное время занимается другим процессом.

Задание 6.2. Использование именованных каналов для реализации сетевого межпроцессного взаимодействия.



Процессы нашего клиент-серверного приложения в livekd

```
Searching for Process with Cid == 25c8
PROCESS fffffa8009b9ab10
    SessionId: 1  Cid: 25c8    Peb: 7efdf000  ParentCid: 12b8
    DirBase: 10e514000  ObjectTable: fffff8a017e672e0  HandleCount:  14.
    Image: Server.exe
```

```
0: kd> !process 29e0 0
Searching for Process with Cid == 29e0
PROCESS fffffa800646db10
    SessionId: 1  Cid: 29e0    Peb: 7efdf000  ParentCid: 19bc
    DirBase: 1c9846000  ObjectTable: fffff8a0074d69b0  HandleCount:  12.
    Image: Client.exe
```

Оба процесса имеют доступ к объектам

```
0030: Object: fffffa8009677820  GrantedAccess: 00160116 Entry: fffff8a018e7d0c0
Object: fffffa8009677820  Type: (fffffa8006107080) File
    ObjectHeader: fffffa80096777f0 (new version)
        HandleCount: 1  PointerCount: 1
        Directory Object: 00000000  Name: \pipeapp {NamedPipe}
```

```
0038: Object: fffffa8006cbffe0  GrantedAccess: 001f0003 Entry: fffff8a018e7d0e0
Object: fffffa8006cbffe0  Type: (fffffa8006106720) Event
    ObjectHeader: fffffa8006cbffb0 (new version)
        HandleCount: 1  PointerCount: 2
        Directory Object: fffff8a00106c080  Name: pipeEvent
```

Вывод: Win32 API позволяют создавать именованные каналы для межпроцессорного взаимодействия.

Приложение 1
Текст программ «Реализация решения о читателях-писателях»
Manager.cpp

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

using namespace std;

int main()
{
	const int READER_COUNT = 8;
	const int WRITER_COUNT = 8;

	const int SEMAPHORE_MAX_VALUE = 6;
	const int MEMORY_PAGE_COUNT = 6;
	const int MEMORY_PAGE_SIZE = 4096;

	const char MemoryName[] = "myMap6Work";

	const char WriterSemaphoreName[] = "Writer semaphore";
	const char ReaderSemaphoreName[] = "Reader semaphore";
	const char* MutexName[] = { "mutexNum0", "mutexNum1", "mutexNum2", "mutexNum3", "mutexNum4",
"mutexNum5" };

	LPCSTR writerProgramPath = "C:/Users/Komdosh/Documents/Visual Studio 2015/Projects/6
OC/Writer/x64/Release/Writer.exe";
	LPCSTR readerProgramPath = "C:/Users/Komdosh/Documents/Visual Studio 2015/Projects/6
OC/Reader/x64/Release/Reader.exe";

	PROCESS_INFORMATION piWriterProcessInfo[WRITER_COUNT];
	PROCESS_INFORMATION piReaderProcessInfo[READER_COUNT];

	HANDLE phMemoryPageMutex[MEMORY_PAGE_COUNT];

	HANDLE hWriterSemaphore = CreateSemaphoreA(NULL, SEMAPHORE_MAX_VALUE,
SEMAPHORE_MAX_VALUE, WriterSemaphoreName);
	if (!hWriterSemaphore)
		cout << "Couldn't create writer semaphore\n";

	HANDLE hReaderSemaphore = CreateSemaphoreA(NULL, SEMAPHORE_MAX_VALUE,
SEMAPHORE_MAX_VALUE, ReaderSemaphoreName);
	if (!hReaderSemaphore)
		cout << "Couldn't create reader semaphore\n";

	for (int i = 0; i < MEMORY_PAGE_COUNT; i++)
	{
		phMemoryPageMutex[i] = CreateMutexA(NULL, false, MutexName[i]);
		if (!phMemoryPageMutex[i])
			cout << "Couldn't create " << i << " mutex\n";
	}

	HANDLE hFile = CreateFileA("C:\\sharefile.txt",
		GENERIC_READ | GENERIC_WRITE,
		FILE_SHARE_READ | FILE_SHARE_WRITE,
		NULL,
		CREATE_NEW,
		FILE_ATTRIBUTE_NORMAL,
		NULL);
	if (hFile)
		cout << "Manager: C:/sharefile.txt created\n";

	HANDLE hMemory = CreateFileMappingA(hFile, NULL, PAGE_READWRITE, 0,
		MEMORY_PAGE_COUNT*MEMORY_PAGE_SIZE + MEMORY_PAGE_COUNT * sizeof(char),
MemoryName);
```

```
	if (!hMemory)
		cout << "Manager: couldn't create file mapping. error code: " << GetLastError() << endl;

	char* memory = (char*)MapViewOfFile(hMemory, FILE_MAP_ALL_ACCESS, 0, 0, 0);
	if (!memory)
		cout << "Manager: couldn't map view of file\n";
	ZeroMemory(memory, MEMORY_PAGE_COUNT*MEMORY_PAGE_SIZE + MEMORY_PAGE_COUNT *
sizeof(char));

	for (int i = 0; i < WRITER_COUNT; i++) {
		STARTUPINFOA startinfo = { sizeof(startinfo) };
		if (CreateProcessA(writerProgramPath, NULL, NULL, NULL, TRUE, 0, NULL, NULL, &startinfo,
&(piWriterProcessInfo[i])) == NULL)
			cout << "Manager: couldn't create writer process #" << i << " error code: " << GetLastError() <<
endl;
		else
			cout << "Writer " << i << " created\n";
		Sleep(100);
	}
	for (int i = 0; i < READER_COUNT; i++) {
		STARTUPINFOA startinfo = { sizeof(startinfo) };
		if (CreateProcessA(readerProgramPath, NULL, NULL, NULL, TRUE, 0, NULL, NULL, &startinfo,
&(piWriterProcessInfo[i])) == NULL)
			cout << "Manager: couldn't create reader process #" << i << " error code: " << GetLastError() <<
"\n";
		else
			cout << "Reader " << i << " created\n";
		Sleep(100);
	}

	for (int i = 0; i < WRITER_COUNT; i++) {
		cout << "Manager: waiting writer #" << i << "\n";
		WaitForSingleObject(piWriterProcessInfo[i].hProcess, 500);
	}
	for (int i = 0; i < READER_COUNT; i++) {
		cout << "Manager: waiting reader #" << i << "\n";
		WaitForSingleObject(piReaderProcessInfo[i].hProcess, 500);
	}
	cout << "Program finished\n";
	system("pause");
	return 0;
}
```

Reader.cpp

```
#include <windows.h>
#include <fstream>
#include <time.h>
#include <stdlib.h>

using namespace std;

int main()
{
	const int MEMORY_PAGE_COUNT = 6;
	const int MEMORY_PAGE_SIZE = 4096;
	const char MemoryName[] = "myMap6Work";
	const char WriterSemaphoreName[] = "Writer Semaphore";
	const char ReaderSemaphoreName[] = "Reader Semaphore";
	const char* mutexName[] = { "mutexNum0", "mutexNum1", "mutexNum2", "mutexNum3", "mutexNum4",
"mutexNum5" };

	srand(time(NULL));
	fstream LogFile;
	LogFile.open("C:\\log.txt", fstream::out | fstream::app);
```

```
	HANDLE hReaderSemaphore = OpenSemaphoreA(SEMAPHORE_ALL_ACCESS, FALSE,
ReaderSemaphoreName);
	if (!hReaderSemaphore)
		LogFile << "Reader: couldn't open reader semaphore\n";
	HANDLE hWriterSemaphore = OpenSemaphoreA(SEMAPHORE_ALL_ACCESS, FALSE, WriterSemaphoreName);
	if (!hWriterSemaphore)
		LogFile << "Writer: couldn't open writer semaphore\n";


	HANDLE hMemory = OpenFileMappingA(FILE_MAP_ALL_ACCESS, FALSE, MemoryName);
	if (!hMemory)
		LogFile << "Reader: couldn't open file mapping\n";

	char* memory = (char*)MapViewOfFile(hMemory, FILE_MAP_ALL_ACCESS, 0, 0, 0);
	if (!memory)
		LogFile << "Reader: couldn't map view of file\n";
	VirtualLock((PVOID)memory, MEMORY_PAGE_COUNT*MEMORY_PAGE_SIZE + MEMORY_PAGE_COUNT
* sizeof(char));

	char data[MEMORY_PAGE_SIZE];
	int page = 0;

	for (int i = 0; i<2;i++) {
		LogFile << GetTickCount() << " reader: wait semaphore \n";
		WaitForSingleObject(hWriterSemaphore, INFINITE);
		LogFile << GetTickCount() << " reader: ready \n";
		Sleep(10);
		for (int j = 0; j < MEMORY_PAGE_COUNT; j++)
		{
			page = (GetCurrentThreadId() + j) % MEMORY_PAGE_COUNT;
			if (memory[page] == 1)
				break;
			if (j == MEMORY_PAGE_COUNT - 1) {
				j = 0;
				LogFile << GetTickCount() << " reader: non-empty page not found, waiting. \n";
				Sleep(300);
			}
		}

		int offset = page * MEMORY_PAGE_SIZE + MEMORY_PAGE_COUNT * sizeof(char);
		HANDLE mutex = OpenMutexA(MUTEX_ALL_ACCESS, TRUE, mutexName[page]);
		WaitForSingleObject(mutex, INFINITE);
		if (mutex != NULL)
		{
			LogFile << GetTickCount() << " reader: start reading\n";
			Sleep(10);
			if (memory[page] == 0) {
				LogFile << GetTickCount() << " reader: page collision! page = " << page << "\n";
				i++;
				ReleaseMutex(mutex);
				ReleaseSemaphore(hWriterSemaphore, 1, NULL);
			}
			else
			{
				memory[page] = 0;
				memcpy(data, memory + offset, MEMORY_PAGE_SIZE);
				LogFile << GetTickCount() << " reader: reading completed. page " << page << " : " <<
data << ". waiting\n";
				ReleaseMutex(mutex);
				ReleaseSemaphore(hReaderSemaphore, 1, NULL);
			}

		}
		Sleep(200 + rand() % 500);
	}
```

```
	return 0;
}
```

Writer.cpp

```
#include <windows.h>
#include <fstream>
#include <time.h>
#include <stdlib.h>

using namespace std;

int main()
{
	const int MEMORY_PAGE_COUNT = 6;
	const int MEMORY_PAGE_SIZE = 4096;
	const char memoryName[] = "myMap6Work";
	const char WriterSemaphoreName[] = "Writer Semaphore";
	const char ReaderSemaphoreName[] = "Reader Semaphore";
	const char* mutexName[] = { "mutexNum0", "mutexNum1", "mutexNum2", "mutexNum3", "mutexNum4",
"mutexNum5" };

	srand(time(NULL));
	fstream LogFile;
	LogFile.open("C:\\log.txt", fstream::out | fstream::app);

	HANDLE hReaderSemaphore = OpenSemaphoreA(SEMAPHORE_ALL_ACCESS, FALSE,
ReaderSemaphoreName);
	if (!hReaderSemaphore)
		LogFile << "Reader: couldn't open reader semaphore\n";
	HANDLE hWriterSemaphore = OpenSemaphoreA(SEMAPHORE_ALL_ACCESS, FALSE, WriterSemaphoreName);
	if (!hWriterSemaphore)
		LogFile << "Writer: couldn't open writer semaphore\n";

	HANDLE hMemory = OpenFileMappingA(FILE_MAP_ALL_ACCESS, FALSE, memoryName);
	if (!hMemory)
		LogFile << "Writer: couldn't open file mapping\n";

	char* memory = (char*)MapViewOfFile(hMemory, FILE_MAP_ALL_ACCESS, 0, 0, 0);
	if (memory)
		LogFile << "Writer: couldn't write data to memory\n";
	VirtualLock((PVOID)memory, MEMORY_PAGE_COUNT*MEMORY_PAGE_SIZE + MEMORY_PAGE_COUNT
* sizeof(char));

	char data[MEMORY_PAGE_SIZE];
	int page = 0;

	for (int i = 0; i<2;i++) {
		LogFile << GetTickCount() << " writer: wait semaphore \n";
		WaitForSingleObject(hReaderSemaphore, INFINITE); // подождать и захватить семафор читателя =
уменьшить число чистых страниц
		LogFile << GetTickCount() << " writer: ready \n";
		Sleep(10);
		for (int j = 0; j < MEMORY_PAGE_COUNT; j++) {
			page = (GetCurrentThreadId() + j) % MEMORY_PAGE_COUNT;
			if (memory[page] == 0)
				break;
			if (j == MEMORY_PAGE_COUNT - 1) {
				j = 0;
				LogFile << GetTickCount() << " writer: empty page not found, waiting. \n";
				Sleep(300);
			}
		}
		int offset = page * MEMORY_PAGE_SIZE + MEMORY_PAGE_COUNT * sizeof(char);
		sprintf(data, "this is page #%d. writing time = %d", page, GetTickCount());
		HANDLE mutex = OpenMutexA(MUTEX_ALL_ACCESS, TRUE, mutexName[page]);
```

```
                WaitForSingleObject(mutex, INFINITE);
                if (mutex){
                        Sleep(10);
                        if (memory[page] == 1) {
                                LogFile << GetTickCount() << " writer: page collision! page = " << page << "\n";
                                i++;
                                ReleaseMutex(mutex);
                                ReleaseSemaphore(hReaderSemaphore, 1, NULL);
                        }
                        else {
                                memory[page] = 1;
                                LogFile << GetTickCount() << " writer: start writing\n";
                                memcpy(memory + offset, data, MEMORY_PAGE_SIZE);
                                LogFile << GetTickCount() << " writer: writing completed. page " << page << ".
waiting\n";
                                ReleaseMutex(mutex);
                                ReleaseSemaphore(hWriterSemaphore, 1, NULL);
                        }
                }
                Sleep(200 + rand() % 500);
        }
        LogFile.close();
        return 0;
}
```

Приложение 2
Текст программ клиент-серверного приложения
Server.cpp

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

using namespace std;


int main()
{
	setlocale(0, ".1251");

	HANDLE hPipe;
	HANDLE hWriteEvent;
	OVERLAPPED olpWrite;

	char buffer[255];

	cout << "Приложение сервер\nДля завершения работы введите '!stop'" << endl;
	hPipe = CreateNamedPipe("\\\\.\\pipe\\pipeapp", PIPE_ACCESS_OUTBOUND | WRITE_DAC |
FILE_FLAG_OVERLAPPED,
		PIPE_TYPE_MESSAGE | PIPE_WAIT, 1, 0, 0, NMPWAIT_USE_DEFAULT_WAIT, NULL);
	if (hPipe == INVALID_HANDLE_VALUE)
		cout << "Не удалось создать канал. код ошибки: " << GetLastError() << endl;
	else
		cout << "Канал создан" << endl;

	hWriteEvent = CreateEventA(NULL, TRUE, FALSE, "pipeEvent");
	if (hWriteEvent != INVALID_HANDLE_VALUE)
		cout << "Событие создано\n";
	else
		cout << "Не удалось создать событие. код ошибки: " << GetLastError() << endl;

	cout << "Соединение.. ";
	if (ConnectNamedPipe(hPipe, NULL))
		cout << "установлено" << endl;
	else
		cout << endl << "ошибка соединения" << endl;

	while (true)
	{
		cout << "Сообщение: ";
		cin >> buffer;
		ZeroMemory(&olpWrite, sizeof(olpWrite));
		olpWrite.hEvent = hWriteEvent;
		WriteFile(hPipe, buffer, 255, NULL, &olpWrite);
		if (!strcmp(buffer, "!stop")) {
			break;
		}
		if (WaitForSingleObject(hWriteEvent, INFINITE) == WAIT_OBJECT_0)
			cout << "отправлено\n";
		else
			cout << "ошибка \n";
	}


	cout << "Отсоединение сервера ";
```

```
	if (DisconnectNamedPipe(hPipe))
		cout << "успешно" << endl;
	else
		cout << "не удалось" << endl;

	CloseHandle(hPipe);
	CloseHandle(hWriteEvent);

	system("pause");
	return 0;
}
```

Client.cpp

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

using namespace std;
char msg[255];
bool exitBool = false;
VOID CALLBACK FileIOCompletionRoutine(DWORD, DWORD, LPOVERLAPPED) {
	if (!strcmp(msg, "!stop"))
		exitBool = true;
	else
		cout << "Message: " << msg << endl;
}

int main()
{
	setlocale(0, ".1251");

	HANDLE hOutPipe;
	char answer;
	int timeoutCur=0, timeoutMax = 10;
	while (timeoutCur<timeoutMax) {
		cout << "Попытка подключения " << timeoutCur << endl;
		hOutPipe = CreateFile("\\\\.\\pipe\\pipeapp", GENERIC_READ, 0, NULL, OPEN_ALWAYS,
FILE_FLAG_OVERLAPPED, NULL);
		if (hOutPipe == INVALID_HANDLE_VALUE) {
			cout << "Не удалось подключиться, переподключение через 1 секунду" << endl;
		}
		else {
			cout << "Соединение успешно установлено" << endl;
			break;
		}
		++timeoutCur;
		Sleep(1000);
	}
	if (timeoutCur == timeoutMax) {
		cout << "Время подключения истекло, подключение не удалось, завершение
программы"<<endl;
		system("pause");
		return -1;
	}

	while (true)
	{
		OVERLAPPED olpReadOverlapper;
		ZeroMemory(&olpReadOverlapper, sizeof(olpReadOverlapper));
```

```
		cout << "Ожидание сообщения\n";

		if (!ReadFileEx(hOutPipe, msg, 255, &olpReadOverlapper, FileIOCompletionRoutine)){
			cout << "Ошибка чтения" << endl;
			return 1;
		}
		SleepEx(INFINITE, 1);
		if (exitBool)
			break;
	}
	CloseHandle(hOutPipe);
	cout << "Соединение закрыто\n";

	system("pause");
	return 0;
}
```