Министерство науки и образования Российской Федерации

Пензенский государственный университет

Кафедра «Системы автоматизированного проектирования»

**Отчет**

По лабораторной работе №4

По курсу «Операционные системы реального времени»

На тему «Работа с разделяемой памятью и файлами в асинхронном режиме»

Выполнил студент гр.20ВВС1

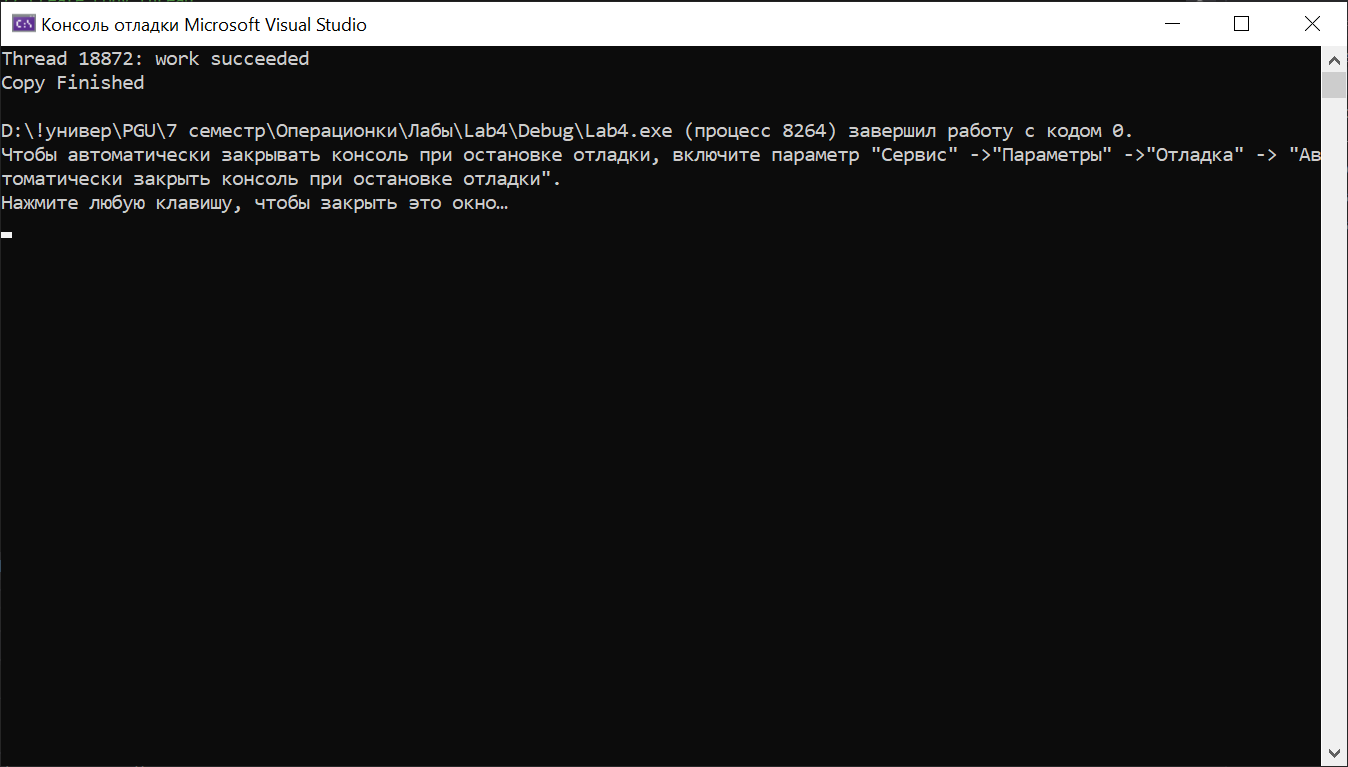
Горбунов Н.А.

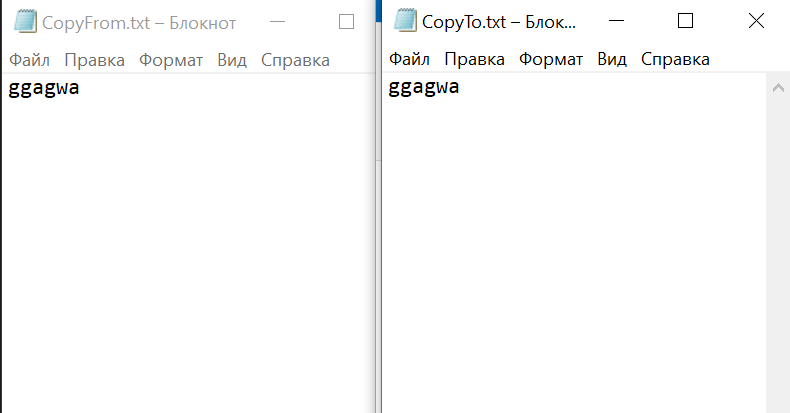
Проверил:

Гудков П.А.

Пенза, 2023

**Задание 7:** разработать менеджер копирования файлов, состоящий из двух нитей. Одна нить должна инициировать операцию копирования, а вторая нить уведомлять о завершении операции.





**Листинг:**

#include <windows.h>

#include <stdio.h>

#include <fstream>

#include <iostream>

#include <string>

using namespace std;

#define MAX\_SEM\_COUNT 10

HANDLE ghSemaphore;

DWORD WINAPI ThreadProc(LPVOID);

int main(void)

{

HANDLE aThread;

DWORD ThreadID;

int i;

// Create a semaphore with initial and max counts of MAX\_SEM\_COUNT

ghSemaphore = CreateSemaphore(

NULL, // default security attributes

MAX\_SEM\_COUNT, // initial count

MAX\_SEM\_COUNT, // maximum count

NULL); // unnamed semaphore

if (ghSemaphore == NULL)

{

printf("CreateSemaphore error: %d\n", GetLastError());

return 1;

}

// Create copy thread

aThread = CreateThread(

NULL, // default security attributes

0, // default stack size

(LPTHREAD\_START\_ROUTINE)ThreadProc,

NULL, // no thread function arguments

0, // default creation flags

&ThreadID); // receive thread identifier

if (aThread == NULL)

{

printf("CreateThread error: %d\n", GetLastError());

return 1;

}

// Wait for all threads to terminate

WaitForMultipleObjects(1, &aThread, TRUE, INFINITE);

cout << "Copy Finished \n";

// Close thread and semaphore handles

CloseHandle(aThread);

CloseHandle(ghSemaphore);

return 0;

}

int CopyMyFile()

{

string line;

//For writing text file

//Creating ofstream & ifstream class object

ifstream ini\_file{ "CopyFrom.txt" };

ofstream out\_file{ "CopyTo.txt" };

if (ini\_file && out\_file) {

while (getline(ini\_file, line)) {

out\_file << line << "\n";

}

}

else {

//Something went wrong

printf("Cannot read File");

}

//Closing file

ini\_file.close();

out\_file.close();

return 0;

}

DWORD WINAPI ThreadProc(LPVOID lpParam)

{

// lpParam not used in this example

UNREFERENCED\_PARAMETER(lpParam);

DWORD dwWaitResult;

BOOL bContinue = TRUE;

CopyMyFile();

while (bContinue)

{

// Try to enter the semaphore gate.

dwWaitResult = WaitForSingleObject(

ghSemaphore, // handle to semaphore

0L); // zero-second time-out interval

switch (dwWaitResult)

{

// The semaphore object was signaled.

case WAIT\_OBJECT\_0:

// TODO: Perform task

printf("Thread %d: work succeeded\n", GetCurrentThreadId());

bContinue = FALSE;

// Simulate thread spending time on task

Sleep(5);

// Release the semaphore when task is finished

if (!ReleaseSemaphore(

ghSemaphore, // handle to semaphore

1, // increase count by one

NULL)) // not interested in previous count

{

printf("ReleaseSemaphore error: %d\n", GetLastError());

}

break;

// The semaphore was nonsignaled, so a time-out occurred.

case WAIT\_TIMEOUT:

printf("Thread %d: wait timed out\n", GetCurrentThreadId());

break;

}

}

return TRUE;

}

**Вывод:** Научился работать с семафорами.