МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М. Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп'ютерних систем та інфокомунікацій

Кафедра комп'ютерних систем, мереж і кібербезпеки

**Лабораторна робота**

з Системного програмування

(назва дисципліни)

на тему: «Вивчення вбудованих об'єктів синхронізації в ОС Windows»

Виконав: студент 3-го курсу групи № 535а

напряму підготовки (спеціальності)

123-«Комп’ютерна інженерія»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(шифр і назва напряму підготовки (спеціальності))

Скринник Є. І.

(прізвище й ініціали студента)

Прийняв: асистент каф.503

Мозговий М.В.

(посада, науковий ступінь, прізвище й ініціали)

Національна шкала: \_\_\_\_\_\_\_\_\_\_

Кількість балів: \_\_\_\_\_

Оцінка: ECTS \_\_\_\_\_

Харків – 2020

**Цель работы:**

1. Изучение встроенных объектов синхронизации в ОС Windows.
2. Изучение системных вызовов Win32 API для реализации алгоритмов межпоточной и межпроцессной синхронизации.

**Постановка задачи:**

**Программа 1:**

Требуется разработать программу, которая контролирует наличие только одного экземпляра самой программы в памяти. Т.е. при попытке запустить программу при уже наличии одного запущенного экземпляра, программа выдает ошибку о невозможности старта. Сама программа просто должна вывести в консоль фразу “Is Running” в случае успешного запуска.

**Программа 2:**

Программа должна контролировать кол-во одновременно открытых указателей на файлы между всеми запущенными потоками. Приложение при старте создает заданное кол-во потоков, где каждый поток при старте переходит в спящий режим на период времени от 1 до 3 сек, потом пытается открыть файл для записи и записать в него время выполнения данной операции. После чего подождать от 1 до 3 сек. И закрыть файл. Программа в процессе работы не может открыть больше чем заданное кол-во файловых указателей. В случае когда уже новый поток не может превысить кол-во одновременно открытых файлов он ожидает пока хотя бы один файл не будет закрыт.

**Программа 3:**

Необходимо написать программу, которая реализует 3х поточную работу (любой алгоритм: например 1 поток считает сумму чисел в массиве, 2ой поток считает среднее значение в массиве, 3ий поток считает макс. и мин значение в массиве). Сам алгоритм вычисления с обращением к критических операторам (обращение к массиву) должен быть реализован в виде взаимоисключения одновременного обращения к источнику данных (массиву).

Задача: программа должна иметь 2 режима работы: с взаимоисключением и без. В каждом режиме должен производиться замер времени работы. Для получения более ощутимых интервалов работать с массивом от 50 тыс. элементов.

**Код первой программы:**

#include <stdio.h>

#include <Windows.h>

#include <locale.h>

HANDLE mutex = CreateMutexA(NULL, FALSE, "MyMutex");

int main()

{

setlocale(LC\_ALL, "russian");

HANDLE mutex = CreateMutex(NULL, FALSE, "SpLAB5Prog");

DWORD result = WaitForSingleObject(mutex, 0);

printf("Лабораторная работа № 5\nПо системному программированию\nВыполнил студент 535а группы Скрынник Егор\nЗапуск единственного экземпляра\n\n");

if (result == WAIT\_OBJECT\_0)

{

printf("Программа запущена!\n");

system("pause");

ReleaseMutex(mutex);

}

else

{

printf("Не удалось запустить программу!\n");

system("pause");

}

CloseHandle(mutex);

return 0;

}

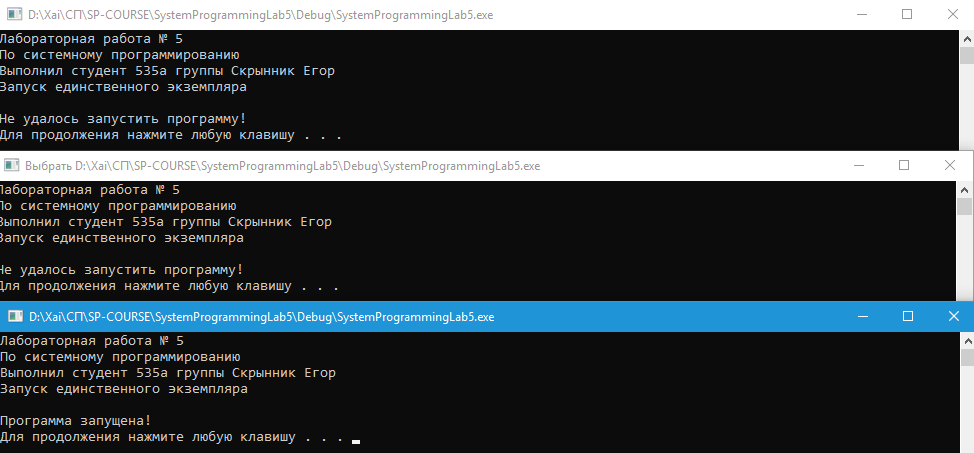
**Результат работы:**

Рисунок 1 – Попытка запуска других экземпляров записи.

**Код второй программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

#include <time.h>

#include <locale.h>

HANDLE semaphore;

// Генерация случайного integer

int random\_integer(int min, int max);

// Функция очистки файла

bool clear\_file(LPCSTR filename);

// Запуск потоков

HANDLE\* launch\_threads(int count);

// Функция потока

DWORD WINAPI thread\_function(LPVOID param);

int main()

{

setlocale(LC\_ALL, "russian");

int amount\_of\_handles; // Количество указателей

int amount\_of\_threads; // Количество потоков

srand(time(NULL));

printf("Лабораторная работа № 5\nПо системному программированию\nВыполнил студент 535а группы Скрынник Егор\nКонтроль потоков\n\n");

printf("Введите количество файловых указателей:");

scanf("%i", &amount\_of\_handles);

printf("Введите максимальное число потоков:");

scanf("%i", &amount\_of\_threads);

//Создание семафора

semaphore = CreateSemaphoreA(NULL, amount\_of\_handles, amount\_of\_handles, "MySemaphore");

if (semaphore == NULL) {

printf("\*\*\* Произошла ошибка во время создания семайора \*\*\*");

system("pause");

return 1;

}

//Очистка файла

if (clear\_file("threadsresult.txt"))

{

printf("\*\*\* Произошла ошибка во время очистки файла \*\*\*");

system("pause");

return 1;

}

WaitForMultipleObjects(amount\_of\_threads, launch\_threads(amount\_of\_threads), TRUE, INFINITE);

system("pause");

return 0;

}

HANDLE\* launch\_threads(int count)

{

HANDLE\* threads = new HANDLE[count];

for (int i = 0; i < count; i++)

{

int\* params = new int[2];

params[0] = i; //Количество потоков

params[1] = random\_integer(1, 3); //Время сна.

threads[i] = CreateThread(

NULL, // дескриптор защиты

0, // начальный размер стека

thread\_function, // функция потока

(LPVOID)params, // параметры потока

NULL, // опции создания

NULL); // идентификатор потока

}

return threads;

}

bool clear\_file(LPCSTR filename)

{

bool operation\_failed = false;

HANDLE file = CreateFileA(filename, GENERIC\_WRITE, FILE\_SHARE\_WRITE, NULL, CREATE\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (file == INVALID\_HANDLE\_VALUE)

{

operation\_failed = true;

}

CloseHandle(file);

return operation\_failed;

}

DWORD WINAPI thread\_function(LPVOID param)

{

int\* params = (int\*)param;

clock\_t start = clock();

int thread\_number = params[0];

LPSTR str = new CHAR[128];

DWORD result = WaitForSingleObject(semaphore, 500);

while (result != WAIT\_OBJECT\_0)

{

result = WaitForSingleObject(semaphore, 1000);

printf("Поток №%i ожидает указатель\n", thread\_number);

}

printf("Поток №%i получил указатель. Уход в сон.\n", thread\_number);

Sleep(params[1] \* 1000);

HANDLE file = CreateFileA("threadsresult.txt", GENERIC\_WRITE, FILE\_SHARE\_WRITE, NULL, OPEN\_ALWAYS, FILE\_ATTRIBUTE\_NORMAL, NULL);

if (file == INVALID\_HANDLE\_VALUE)

{

ReleaseSemaphore(semaphore, 1, NULL);

return 0;

}

SetFilePointer(file, 0, NULL, FILE\_END);

clock\_t finish = clock();

float time\_elapsed = (finish - start) / CLK\_TCK;

sprintf(str, "Поток №%i завершил свою работу спустя %f секунд\n\0", thread\_number, time\_elapsed);

WriteFile(file, str, strlen(str), NULL, NULL);

CloseHandle(file);

printf("Поток №%i освободил указатель.\n", thread\_number);

ReleaseSemaphore(semaphore, 1, NULL); //Освобождение указателя

return 0;

}

int random\_integer(int min, int max)

{

return min + rand() % (max + 1 - min);

}

**Результат работы:**

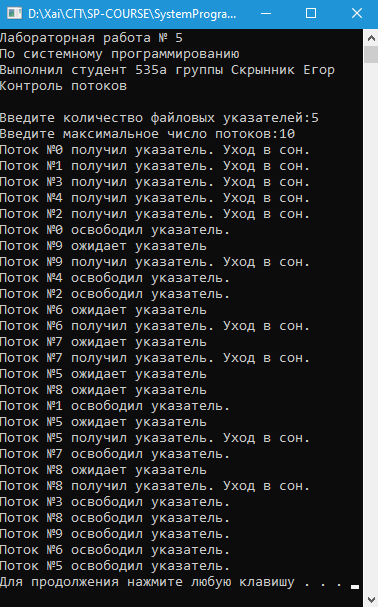


Рисунок 2 – Работа приложения.

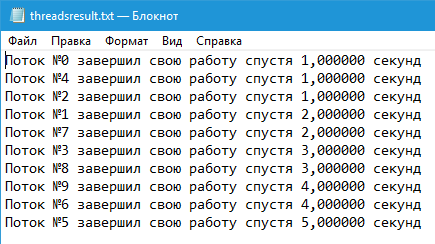


Рисунок 3 – Время выполнения операций потоков.

**Код третьей программы:**

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <Windows.h>

#include <time.h>

#include <locale.h>

#define ARRAY\_MAX\_SIZE 60000000

#define THREAD\_ACTION\_FIND\_MIN 0

#define THREAD\_ACTION\_FIND\_MAX 1

#define THREAD\_ACTION\_FIND\_AVG 2

#define THREAD\_ACTION\_TYPE 0

#define THREAD\_CRITICAL\_SECTION\_FLAG 1

CRITICAL\_SECTION section;

int\* array;

// Запуск потоков на выполнение

HANDLE\* launch\_threads(LPVOID use\_critical\_section);

// Функция потока

DWORD WINAPI thread\_function(LPVOID param);

// Функция заполнения массива

void fill\_array(int\* array);

// Функция генерация случайного целого

int random\_integer(int min, int max);

int main()

{

setlocale(LC\_ALL, "russian");

HANDLE\* threads;

clock\_t start;

float elapsed\_time;

srand(time(NULL));

printf("Лабораторная работа № 5\nПо системному программированию\nВыполнил студент 535а группы Скрынник Егор\n3х поточная работа\n\n");

InitializeCriticalSection(&section);

array = new int[ARRAY\_MAX\_SIZE];

fill\_array(array);

start = clock();

threads = launch\_threads((LPVOID)TRUE);

WaitForMultipleObjects(3, threads , TRUE, INFINITE);

for (int i = 0; i < 3; i++)

CloseHandle(threads[i]);

elapsed\_time = ((float)(clock() - start)) / CLK\_TCK;

printf("Время выполнения с использованием критической секции составляет %f секунд\n\n", elapsed\_time);

start = clock();

threads = launch\_threads((LPVOID)FALSE);

WaitForMultipleObjects(3, threads , TRUE, INFINITE);

for (int i = 0; i < 3; i++)

CloseHandle(threads[i]);

elapsed\_time = ((float)(clock() - start)) / CLK\_TCK;

printf("Время выполнения без использования критической секции составляет %f секунд\n\n", elapsed\_time);

DeleteCriticalSection(&section);

system("pause");

}

HANDLE\* launch\_threads(LPVOID use\_critical\_section)

{

HANDLE\* threads = new HANDLE[3];

for (int i = 0; i < 3; i++)

{

int\* params = new int[2];

params[THREAD\_ACTION\_TYPE] = i; //Тип операции

params[THREAD\_CRITICAL\_SECTION\_FLAG] = (int)use\_critical\_section;

threads[i] = CreateThread(

NULL, // дескриптор защиты

0, // начальный размер стека

thread\_function, // функция потока

(LPVOID)params, // параметры потока

NULL, // опции создания

NULL); // идентификатор потока

}

return threads;

}

DWORD WINAPI thread\_function(LPVOID param)

{

int\* params = (int\*)param;

if ((bool)params[THREAD\_CRITICAL\_SECTION\_FLAG])

{

while (!TryEnterCriticalSection(&section))

{

//Простой, ожиданиея пока освободится критическая секция

}

}

int min = array[0];

int max = array[0];

float avg = 0;

switch (params[THREAD\_ACTION\_TYPE])

{

case THREAD\_ACTION\_FIND\_MIN:

for (int i = 0; i < ARRAY\_MAX\_SIZE; i++)

{

if (min > array[i])

min = array[i];

}

printf("Минимум: %i\n", min);

break;

case THREAD\_ACTION\_FIND\_MAX:

for (int i = 0; i < ARRAY\_MAX\_SIZE; i++)

{

if (max < array[i])

max = array[i];

}

printf("Максимум: %i\n", max);

break;

case THREAD\_ACTION\_FIND\_AVG:

for (int i = 0; i < ARRAY\_MAX\_SIZE; i++)

{

avg += array[i];

}

printf("Среднее: %f\n", avg / ARRAY\_MAX\_SIZE);

break;

}

if ((bool)params[THREAD\_CRITICAL\_SECTION\_FLAG])

{

LeaveCriticalSection(&section);

}

return 0;

}

void fill\_array(int\* array)

{

for (int i = 0; i < ARRAY\_MAX\_SIZE; i++)

{

array[i] = random\_integer(0, 1000);

}

}

int random\_integer(int min, int max)

{

return min + rand() % (max + 1 - min);

}

**Результат работы:**

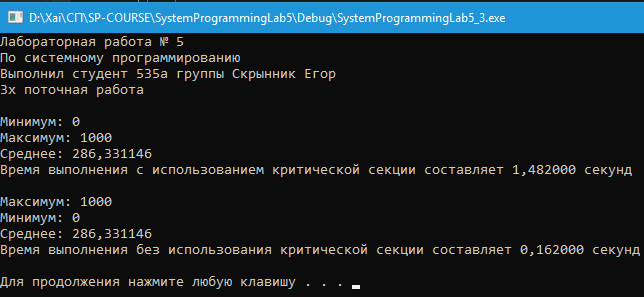


Рисунок 4 – Временные характеристики и результаты работы потоков.