Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет
по лабораторной работе № 6

Дисциплина: Операционные системы

Тема: Реализация монитора Хоара «Читатели-писатели» под ОС Windows

Студент: Платонова Ольга

Группа: ИУ7-55Б

Преподаватели: Рязанова Н. Ю.

Москва, 2020 г.

Оглавление

Задание.

В лабораторной работе необходимо разработать многопоточное приложение, используя API ОС Windows такие как, потоки, события (eevent) и мьютексы (mutex). Потоки разделяют единственную глобальную переменную. Приложение реализует монитор Хоара «Читатели-писатели».

Код программы.

1. Файл errors.h

```
#ifndef EXIT_CODES_H
#define EXIT_CODES_H

enum ExitCodes
{
    SUCCESS,

    MUTEX_ERR,

    WR_EVENT_ERR,
    RD_EVENT_ERR,

    WR_THREAD_ERR,
    RD_THREAD_ERR
};
#endif
```

2. Файл main.c

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

#include "errors.h"
```

```
const int NumWriters = 3;
const int NumReaders = 5;
const int LastRecord = 15;

static bool isEndWrite = false;
static bool isEndRead = false;

static bool activeWriter = 0;
static int waitingWriters = 0;

static int activeReaders = 0;
static int waitingReaders = 0;

static int value = 0;

HANDLE writers[NumWriters];
HANDLE readers[NumReaders];

HANDLE mutex;
HANDLE canRead, canWrite;


//запись
void startWrite(void)
{
	InterlockedIncrement(&waitingWriters);

	if (activeReaders > 0 || activeWriter) {
		WaitForSingleObject(canWrite, INFINITE);
	}

	InterlockedDecrement(&waitingWriters);

	activeWriter = true;
	ResetEvent(canWrite);
}
```

```
void stopWrite(void)
{
	activeWriter = false;

	if (waitingWriters) {
		SetEvent(canWrite);
	}
	else {
		SetEvent(canRead);
	}
}


//чтение
void startRead(void)
{
	InterlockedIncrement(&waitingReaders);

	if (activeWriter || waitingWriters > 0) {
		WaitForSingleObject(canRead, INFINITE);
	}

	WaitForSingleObject(mutex, INFINITE);

	InterlockedDecrement(&waitingReaders);
	InterlockedIncrement(&activeReaders);

	SetEvent(canRead);
	ReleaseMutex(mutex);
}

void stopRead(void)
{
	InterlockedDecrement(&activeReaders);

	if (activeReaders == 0) {
		SetEvent(canWrite);
	}
}
```

```
DWORD WINAPI writer(const LPVOID lpParam)
{
	while (!isEndWrite) {
		startWrite();
			value++;
			printf("Writer %d:\t (id %d)\t %d --> \n", (int) lpParam, GetCurrentThreadId(), value);
		stopWrite();

		isEndWrite = (value >= LastRecord);
		Sleep(200);
	}

	return SUCCESS;
}

DWORD WINAPI reader(const LPVOID lpParam)
{
	while (!isEndRead) {
		startRead();
			printf("Reader %d:\t (id %d)\t %d <-- \n", (int) lpParam, GetCurrentThreadId(), value);
		stopRead();

		isEndRead = (value >= LastRecord);
		Sleep(200);
	}

	return SUCCESS;
}



int main(void)
{
	mutex = CreateMutex(NULL, FALSE, NULL);
	if (mutex == NULL) {
		perror("Create mutex");
		return MUTEX_ERR;
	}
```

```
//автосброс
canRead = CreateEvent(NULL, FALSE, FALSE, "RD event");
if (canRead == NULL) {
	CloseHandle(mutex);
	perror("Read event");
	return RD_EVENT_ERR;
}

//сброс вручную
canWrite = CreateEvent(NULL, TRUE, FALSE, "WR event");
if (canWrite == NULL) {
	CloseHandle(mutex);
	CloseHandle(canRead);
	perror("Write event");
	return WR_EVENT_ERR;
}



for (int i = 0; i < NumWriters; i++) {
	writers[i] = CreateThread(NULL, 0, &writer, (LPVOID) i, 0, NULL);
	if (writers[i] == NULL) {
		CloseHandle(mutex);
		CloseHandle(canRead);
		CloseHandle(canWrite);
		perror("Write thread");
		return WR_THREAD_ERR;
	}
}

for (int i = 0; i < NumReaders; i++) {
	readers[i] = CreateThread(NULL, 0, &reader, (LPVOID) i, 0, NULL);
	if (readers[i] == NULL) {
		CloseHandle(mutex);
		CloseHandle(canRead);
		CloseHandle(canWrite);
		perror("Read thread");
		return RD_THREAD_ERR;
	}
}
```

```
    WaitForMultipleObjects(NumWriters, writers, TRUE, INFINITE);
    WaitForMultipleObjects(NumReaders, readers, TRUE, INFINITE);

    CloseHandle(mutex);
    CloseHandle(canRead);
    CloseHandle(canWrite);

    return 0;
}
```

Результат работы программы.