|  |  |
| --- | --- |
|  | **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

**Лабораторная работа № 6**

|  |  |
| --- | --- |
| Тема: Реализация монитора Хоара «Читатели-писатели» под ОС Windows **Студент** ВоякинА. Я.  **Группа ИУ7-54Б**  **Преподаватель** Рязанова Н. Ю. |  |

*Москва, 2020 г.*

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

**Листинг программы:**

#**include** <stdio.h>  
#**include** <stdbool.h>  
#**include** <windows.h>  
   
#define READERS\_NUM 5  
#define WRITERS\_NUM 3

const int iterations = 5;  
const DWORD sleep\_time = 200;  
volatile LONG active\_readers = 0;  
volatile LONG writers\_in\_wait = 0;  
volatile LONG readers\_in\_wait = 0;

bool writer\_is\_active = false;  
volatile int current\_num = 0;  
   
HANDLE mutex;  
HANDLE can\_read;  
HANDLE can\_write;

HANDLE writers[WRITERS\_NUM];  
HANDLE readers[READERS\_NUM];  
   
void start\_write()  
{  
 InterlockedIncrement(&writers\_in\_wait);  
 **if** ( WaitForSingleObject(can\_read, 0) == WAIT\_OBJECT\_0|| writer\_is\_active)  
 {  
 WaitForSingleObject(can\_write, INFINITE);  
 }  
   
 InterlockedDecrement(&writers\_in\_wait);  
 writer\_is\_active = true;  
 ResetEvent(can\_write);  
}  
   
void stop\_write()  
{  
 writer\_is\_active = false;  
   
 **if** (!writers\_in\_wait)  
 {  
 SetEvent(can\_read);  
 }   
   
 **else**  
 {  
 SetEvent(can\_write);  
 }  
}

DWORD WINAPI writer(LPVOID lpParams)  
{  
 for (int current\_it = 0; current\_it < iterations; current\_it++)  
 {  
 start\_write();  
 current\_num += 1;  
 printf("Писатель %ld написал: %d\n", (int) lpParams, current\_num);  
 stop\_write();  
 Sleep(sleep\_time);  
 }  
   
 return EXIT\_SUCCESS;  
}

void start\_read()  
{  
 InterlockedIncrement(&readers\_in\_wait);  
 **if** (WaitForSingleObject(can\_write, 0) == WAIT\_OBJECT\_0 || writer\_is\_active)  
 {  
 WaitForSingleObject(can\_read, INFINITE);  
 }  
 WaitForSingleObject(mutex, INFINITE);  
   
 InterlockedDecrement(&readers\_in\_wait);  
 InterlockedIncrement(&active\_readers);  
 SetEvent(can\_read);  
 ReleaseMutex(mutex);  
}  
   
void stop\_read()  
{  
 InterlockedDecrement(&active\_readers);  
 **if** (readers\_in\_wait == 0)  
 {  
 SetEvent(can\_write);  
 }  
}  
   
DWORD WINAPI reader(LPVOID lpParams)  
{  
 **while** (current\_num < iterations \* WRITERS\_NUM)  
 {  
 start\_read();  
 printf("Читатель %ld прочитал: %d\n", (int) lpParams, current\_num);  
 stop\_read();  
 Sleep(sleep\_time);  
 }  
 return EXIT\_SUCCESS;  
}  
   
int init\_handles()  
{  
 **if** ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL)  
 {  
 perror("Can not create mutex!");  
 return EXIT\_FAILURE;  
 }  
   
 **if** ((can\_read = CreateEvent(NULL, FALSE, TRUE, NULL)) == NULL)  
 {  
 perror("Can not create event can\_read!");  
 return EXIT\_FAILURE;  
 }  
   
 **if** ((can\_write = CreateEvent(NULL, TRUE, TRUE, NULL)) == NULL)  
 {  
 perror("Can not create event can\_write!");  
 return EXIT\_FAILURE;  
 }  
   
 return EXIT\_SUCCESS;  
}  
   
int create\_threads(HANDLE \*threads, int threads\_count, DWORD (\*on\_thread)(LPVOID))  
{  
 for (int cur\_thread\_num = 0; cur\_thread\_num < threads\_count; cur\_thread\_num++)  
 {  
 **if** ((threads[cur\_thread\_num] = CreateThread(NULL, 0, on\_thread, (LPVOID) cur\_thread\_num, 0, NULL)) == NULL)  
 {  
 perror("Can not create event create thread!");  
 return EXIT\_FAILURE;  
 }  
 }  
 return EXIT\_SUCCESS;  
}  
   
int main()  
{  
 int rc = EXIT\_SUCCESS;  
   
 **if** ((rc = init\_handles()) != EXIT\_SUCCESS || (rc = create\_threads(writers, WRITERS\_CNT, writer)) != EXIT\_SUCCESS  
 || (rc = create\_threads(readers, READERS\_CNT, reader)) != EXIT\_SUCCESS){ return rc;}  
   
 WaitForMultipleObjects(WRITERS\_NUM, writers, TRUE, INFINITE);  
 WaitForMultipleObjects(READERS\_NUM, readers, TRUE, INFINITE);  
   
 CloseHandle(mutex);  
 CloseHandle(can\_read);  
 CloseHandle(can\_write);  
   
 return rc;  
}

**Результат работы программы:**

Писатель 0 написал: 1  
Читатель 3 прочитал: 1  
Читатель 4 прочитал: 1  
Читатель 2 прочитал: 1  
Читатель 1 прочитал: 1  
Читатель 0 прочитал: 1  
Писатель 1 написал: 2  
Писатель 2 написал: 3  
Писатель 0 написал: 4  
Читатель 3 прочитал: 4  
Читатель 2 прочитал: 4  
Читатель 4 прочитал: 4  
Читатель 1 прочитал: 4  
Читатель 0 прочитал: 4  
Писатель 1 написал: 5  
Писатель 2 написал: 6  
Писатель 0 написал: 7  
Читатель 3 прочитал: 7  
Читатель 2 прочитал: 7  
Читатель 4 прочитал: 7  
Читатель 1 прочитал: 7  
Читатель 0 прочитал: 7  
Писатель 1 написал: 8  
Писатель 2 написал: 9  
Писатель 0 написал: 10  
Читатель 3 прочитал: 10  
Читатель 2 прочитал: 10  
Читатель 4 прочитал: 10  
Читатель 1 прочитал: 10  
Читатель 0 прочитал: 10  
Писатель 1 написал: 11  
Писатель 2 написал: 12  
Писатель 0 написал: 13  
Читатель 3 прочитал: 13  
Читатель 2 прочитал: 13  
Читатель 4 прочитал: 13  
Читатель 1 прочитал: 13  
Читатель 0 прочитал: 13  
Писатель 1 написал: 14  
Писатель 2 написал: 15