



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

Студент \_\_\_\_\_ Топорков Павел \_\_\_\_\_

Группа \_\_\_\_\_ ИУ7-53Б \_\_\_\_\_

Дисциплина \_\_\_\_\_ Операционные системы \_\_\_\_\_

Преподаватель: \_\_\_\_\_ Рязанова Н. Ю. \_\_\_\_\_

подпись, дата

Фамилия, И.О.

Оценка \_\_\_\_\_

Москва — 2021 г.

# Оглавление

<b>1</b>	<b>Задание №1</b>	<b>2</b>
1.1	Задание . . . . .	2
1.2	Код программы . . . . .	2
1.3	Демонстрация работы программы . . . . .	6

# 1 Задание №1

## 1.1 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».

## 1.2 Код программы

Листинг 1.1: task1

```
0 #include <windows.h>
1 #include <stdbool.h>
2 #include <stdio.h>
3 #include <stdbool.h>
4
5 #define OK 0
6
7 #define CREATE_MUTEX_ERROR 1
8 #define CREATE_EVENT_ERROR 2
9 #define CREATE_READER_THREAD_ERROR 3
10 #define CREATE_WRITER_THREAD_ERROR 3
11
12 #define READERS_COUNT 3
13 #define WRITERS_COUNT 3
14
15 #define COUNT 3
16
17 const DWORD sleep_time_writer = 50;
18 const DWORD sleep_time_reader = 30;
19
20 volatile LONG waiting_writers = 0;
21 volatile LONG waiting_readers = 0;
22 volatile LONG active_readers = 0;
23
24 HANDLE can_read, can_write, mutex;
25 HANDLE reader_threads[READERS_COUNT];
26 HANDLE writer_threads[WRITERS_COUNT];
27
28 char data = 'A' - 1;
```

```

29 bool flag = false;
30
31
32 bool turn(HANDLE event) {
33     return WaitForSingleObject(event, 0) == WAIT_OBJECT_0;
34 }
35
36
37 void start_read() {
38     InterlockedIncrement(&waiting_readers);
39     if (turn(can_write) || flag) {
40         WaitForSingleObject(can_read, INFINITE);
41     }
42     WaitForSingleObject(mutex, INFINITE);
43     InterlockedDecrement(&waiting_readers);
44     InterlockedIncrement(&active_readers);
45     SetEvent(can_read);
46     ReleaseMutex(mutex);
47 }
48
49
50 void stop_read() {
51     InterlockedDecrement(&active_readers);
52     if (active_readers == 0) {
53         SetEvent(can_write);
54     }
55 }
56
57
58 DWORD WINAPI reader(CONST LPVOID param) {
59     for (; data < 'A' - 1 + WRITERS_COUNT * COUNT;) {
60         start_read();
61         printf("Reader %ld <<<< %c\n", GetCurrentThreadId(), data);
62         stop_read();
63         Sleep(sleep_time_reader);
64     }
65     return 0;
66 }
67
68
69 void start_write() {
70     InterlockedIncrement(&waiting_writers);
71     if (active_readers > 0 || flag) {
72         WaitForSingleObject(can_write, INFINITE);
73     }
74     InterlockedDecrement(&waiting_writers);
75     flag = true;
76 }

```

```

77
78
79 void stop_write(){
80     flag = false;
81     if (waiting_readers) {
82         SetEvent(can_read);
83         return;
84     }
85     SetEvent(can_write);
86 }
87
88
89 DWORD WINAPI writer(CONST LPVOID param) {
90     for (int i = 0; i < COUNT; ++i) {
91         start_write();
92         printf("Writer %ld >>>> %c\n", GetCurrentThreadId(), ++data);
93         stop_write();
94         Sleep(sleep_time_writer);
95     }
96     return 0;
97 }
98
99
100 int init_handles() {
101     mutex = CreateMutex(NULL, FALSE, NULL);
102     if (mutex == NULL) {
103         perror("CreateMutex");
104         return CREATE_MUTEX_ERROR;
105     }
106
107     can_read = CreateEvent(NULL, FALSE, FALSE, NULL);
108     if (can_read == NULL) {
109         perror("CreateEvent (canRead)");
110         return CREATE_EVENT_ERROR;
111     }
112
113     can_write = CreateEvent(NULL, FALSE, FALSE, NULL);
114     if (can_write == NULL) {
115         perror("CreateEvent (canWrite)");
116         return CREATE_EVENT_ERROR;
117     }
118
119     return OK;
120 }
121
122
123 int create_threads() {
124     for (int i = 0; i < WRITERS_COUNT; i++) {

```

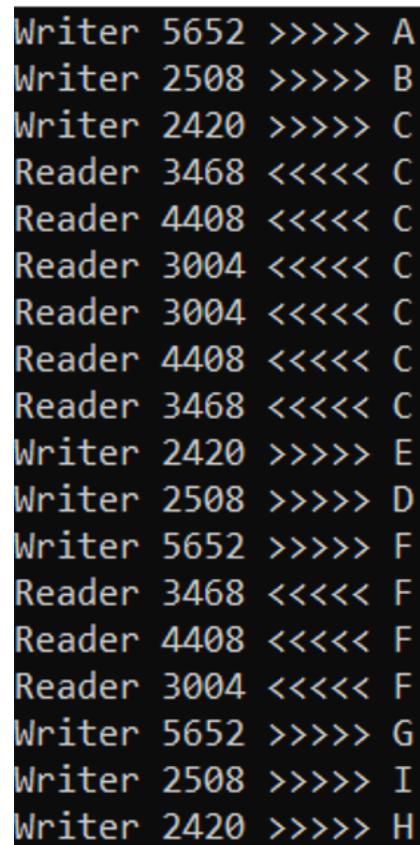
```

125     writer_threads[i] = CreateThread(NULL, 0, &writer, NULL, 0, NULL);
126     if (writer_threads[i] == NULL) {
127         perror("CreateThread (writer)");
128         return CREATE_WRITER_THREAD_ERROR;
129     }
130 }
131
132 for (int i = 0; i < READERS_COUNT; ++i) {
133     reader_threads[i] = CreateThread(NULL, 0, &reader, NULL, 0, NULL);
134     if (reader_threads[i] == NULL) {
135         perror("CreateThread (reader)");
136         return CREATE_READER_THREAD_ERROR;
137     }
138 }
139
140 return OK;
141 }
142
143
144 void close() {
145     for (int i = 0; i < READERS_COUNT; ++i) {
146         CloseHandle(reader_threads[i]);
147     }
148
149     for (int i = 0; i < WRITERS_COUNT; ++i) {
150         CloseHandle(writer_threads[i]);
151     }
152
153     CloseHandle(can_read);
154     CloseHandle(can_write);
155     CloseHandle(mutex);
156 }
157
158
159 int main(void)
160 {
161     int check = init_handles();
162     if (check) {
163         return check;
164     }
165
166     check = create_threads();
167     if (check) {
168         return check;
169     }
170
171     WaitForMultipleObjects(READERS_COUNT, reader_threads, TRUE, INFINITE);
172     WaitForMultipleObjects(WRITERS_COUNT, writer_threads, TRUE, INFINITE);

```

```
173  
174     close();  
175     return OK;  
176 }
```

### 1.3 Демонстрация работы программы



```
Writer 5652 >>>> A  
Writer 2508 >>>> B  
Writer 2420 >>>> C  
Reader 3468 <<<< C  
Reader 4408 <<<< C  
Reader 3004 <<<< C  
Reader 3004 <<<< C  
Reader 4408 <<<< C  
Reader 3468 <<<< C  
Writer 2420 >>>> E  
Writer 2508 >>>> D  
Writer 5652 >>>> F  
Reader 3468 <<<< F  
Reader 4408 <<<< F  
Reader 3004 <<<< F  
Writer 5652 >>>> G  
Writer 2508 >>>> I  
Writer 2420 >>>> H
```

Рисунок 1.1