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»

Отчёт

по лабораторной работе № 6

Название: Реализация монитора Хоара «Читатели-писатели»

Дисциплина: Операционные системы

Студент ИУ7-55Б _____ Д.В. Сусликов

(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Н.Ю. Рязанова

(Подпись, дата) (И.О. Фамилия)

Москва, 2020

Содержание

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».

Программа

В Листинге 1 описан код программы.

Листинг 1 – Задание

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

#define OK 0
#define ERROR 1

#define WRITERS 3
#define READERS 5
#define ITERS 5

int value = 0;

HANDLE writers[WRITERS];
HANDLE readers[READERS];

HANDLE can_read;
HANDLE can_write;
HANDLE mutex;

bool is_active_writer = false;
```

```
22  unsigned int active_readers = 0;
23
24  unsigned int waiting_writers = 0;
25  unsigned int waiting_readers = 0;
26
27  void start_write()
28  {
29    InterlockedIncrement(&waiting_writers);
30    if (is_active_writer || active_readers > 0)
31      WaitForSingleObject(can_write, INFINITE);
32
33    InterlockedDecrement(&waiting_writers);
34    is_active_writer = true;
35    ResetEvent(can_write);
36  }
37
38  void stop_write()
39  {
40    is_active_writer = false;
41    if (waiting_writers)
42      SetEvent(can_write);
43    else
44      SetEvent(can_read);
45  }
46
47  void start_read()
48  {
49    InterlockedIncrement(&waiting_readers);
50    if (is_active_writer || waiting_writers > 0)
51      WaitForSingleObject(can_read, INFINITE);
52
53    WaitForSingleObject(mutex, INFINITE);
54
55    InterlockedDecrement(&waiting_readers);
56    InterlockedIncrement(&active_readers);
```

```
57      SetEvent(can_read);
58
59      ReleaseMutex(mutex);
60    }
61
62    void stop_read()
63    {
64      InterlockedDecrement(&active_readers);
65      if (active_readers == 0)
66        SetEvent(can_write);
67    }
68
69    DWORD WINAPI reader(LPVOID lpParam)
70    {
71      while (value < WRITERS * ITERS)
72      {
73        start_read();
74        printf("Reader %d with id %d read %d\n", (int)lpParam,
              GetCurrentThreadId(), value);
75        stop_read();
76        Sleep(200);
77      }
78      return OK;
79    }
80
81    DWORD WINAPI writer(LPVOID lpParam)
82    {
83      for (int i = 0; i < ITERS; i++)
84      {
85        start_write();
86        value++;
87        printf("Writer %d with id %d wrote %d\n", (int)lpParam,
              GetCurrentThreadId(), value);
88        stop_write();
89        Sleep(200);
```

```
90      }
91      return OK;
92    }
93
94    bool check_error(HANDLE cur, const char* msg)
95    {
96      if (cur == NULL)
97      {
98        CloseHandle(mutex);
99        CloseHandle(can_read);
100       CloseHandle(can_write);
101       perror(msg);
102       return false;
103     }
104     return true;
105   }
106
107   int main()
108   {
109     mutex = CreateMutex(NULL, FALSE, NULL);
110     if (mutex == NULL)
111     {
112       perror("mutex");
113       return ERROR;
114     }
115
116     can_read = CreateEvent(NULL, FALSE, FALSE, TEXT("ReadEvent"));
117     if (can_read == NULL)
118     {
119       CloseHandle(mutex);
120       perror("can_read");
121       return ERROR;
122     }
123
124     can_write = CreateEvent(NULL, TRUE, FALSE, TEXT("WriteEvent"));
```

```
125     if (can_write == NULL)
126     {
127       CloseHandle(mutex);
128       CloseHandle(can_read);
129       perror("can_write");
130       return ERROR;
131     }
132
133     for (int i = 0; i < WRITERS; i++)
134     {
135       writers[i] = CreateThread(NULL, 0, &writer, (LPVOID)i, 0, NULL
            );
136       if (!check_error(writers[i], "Thread"))
137         return ERROR;
138     }
139
140     for (int i = 0; i < READERS; i++)
141     {
142       readers[i] = CreateThread(NULL, 0, &reader, (LPVOID)i, 0, NULL
            );
143       if (!check_error(readers[i], "Thread"))
144         return ERROR;
145     }
146
147     WaitForMultipleObjects(WRITERS, writers, TRUE, INFINITE);
148     WaitForMultipleObjects(READERS, readers, TRUE, INFINITE);
149
150     CloseHandle(mutex);
151     CloseHandle(can_read);
152     CloseHandle(can_write);
153
154     return 0;
155   }
```

Ниже на Рисунке 1 показан пример работы данной программы.

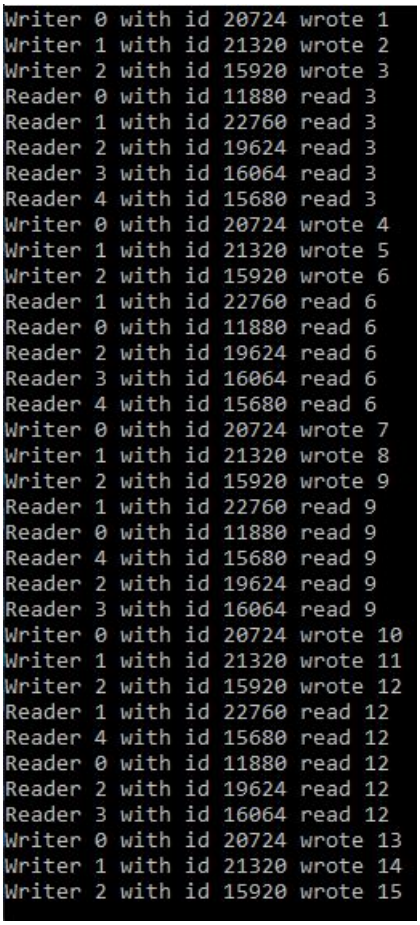

Рисунок 1 – Пример работы программы