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 по курсу «Операционные системы»

Тема Задача «Читатели – писатели» под ОС Windows

Студент Кононенко С.С.

Группа ИУ7-53Б

Оценка (баллы) _____

Преподаватели Рязанова Н.Ю.

Москва — 2020 г.

Задача «Читатели – писатели»

Листинг 1 – Реализация задачи «читатели – писатели»

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

#define READERS_COUNT 5
#define WRITERS_COUNT 3

#define READ_ITERS 7
#define WRITE_ITERS 8

#define READ_TIMEOUT 300
#define WRITE_TIMEOUT 300

#define DIFF 4000

HANDLE mutex;
HANDLE can_read;
HANDLE can_write;

LONG waiting_writers = 0;
LONG waiting_readers = 0;
LONG active_readers = 0;

bool active_writer = false;

int val = 0;

void start_read()
{
    InterlockedIncrement(&waiting_readers);

    if (active_writer || (WaitForSingleObject(can_write, 0) ==
       WAIT_OBJECT_0 && waiting_writers))
    {
        WaitForSingleObject(can_read, INFINITE);
    }
    WaitForSingleObject(mutex, INFINITE);

    InterlockedDecrement(&waiting_readers);
    InterlockedIncrement(&active_readers);
```

```
41
42      SetEvent(can_read);
43      ReleaseMutex(mutex);
44  }
45
46  void stop_read()
47  {
48      InterlockedDecrement(&active_readers);
49
50      if (active_readers == 0)
51      {
52          ResetEvent(can_read);
53          SetEvent(can_write);
54      }
55  }
56
57  void start_write(void)
58  {
59      InterlockedIncrement(&waiting_writers);
60
61      if (active_writer || active_readers > 0)
62      {
63          WaitForSingleObject(can_write, INFINITE);
64      }
65
66      InterlockedDecrement(&waiting_writers);
67
68      active_writer = true;
69  }
70
71  void stop_write(void)
72  {
73      active_writer = false;
74
75      if (waiting_readers)
76      {
77          SetEvent(can_read);
78      }
79      else
80      {
81          SetEvent(can_write);
82      }
83  }
84
85  DWORD WINAPI rr_run(CONST LPVOID lpParams)
86  {
87      int r_id = (int)lpParams;
88      srand(time(NULL) + r_id);
```

```
89
90      int stime;
91
92      for (size_t i = 0; i < READ_ITERS; i++)
93      {
94          stime = READ_TIMEOUT + rand() % DIFF;
95          Sleep(stime);
96          start_read();
97          printf("?Reader #%d read: %3d // Idle time: %dms\n", r_id, val,
                stime);
98          stop_read();
99      }
100
101     return 0;
102 }
103
104 DWORD WINAPI wr_run(CONST LPVOID lpParams)
105 {
106     int w_id = (int)lpParams;
107     srand(time(NULL) + w_id + READERS_COUNT);
108
109     int stime;
110
111     for (size_t i = 0; i < WRITE_ITERS; ++i)
112     {
113         stime = WRITE_TIMEOUT + rand() % DIFF;
114         Sleep(stime);
115         start_write();
116         ++val;
117         printf("!Writer #%d wrote: %3d // Idle time: %dms\n", w_id, val,
                stime);
118         stop_write();
119     }
120     return 0;
121 }
122
123 int main()
124 {
125     setbuf(stdout, NULL);
126
127     HANDLE readers_threads[READERS_COUNT];
128     HANDLE writers_threads[WRITERS_COUNT];
129
130     if ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL)
131     {
132         perror("Failed call of CreateMutex");
133
134         return -1;
```

```
135     }
136
137     can_read = CreateEvent(NULL, FALSE, FALSE, NULL);
138     can_write = CreateEvent(NULL, FALSE, FALSE, NULL);
139
140     if (can_read == NULL || can_write == NULL)
141     {
142         perror("Failed call of CreateEvent");
143
144         return -1;
145     }
146
147     for (size_t i = 0; i < READERS_COUNT; ++i)
148     {
149         readers_threads[i] = CreateThread(NULL, 0, rr_run, (LPVOID)i, 0,
                NULL);
150         if (readers_threads[i] == NULL)
151         {
152             perror("Failed call of CreateThread");
153             return -1;
154         }
155     }
156
157     for (size_t i = 0; i < WRITERS_COUNT; ++i)
158     {
159         writers_threads[i] = CreateThread(NULL, 0, wr_run, (LPVOID)i, 0,
                NULL);
160         if (writers_threads[i] == NULL)
161         {
162             perror("Failed call of CreateThread");
163
164             return -1;
165         }
166     }
167
168     WaitForMultipleObjects(READERS_COUNT, readers_threads, TRUE, INFINITE)
            ;
169     WaitForMultipleObjects(WRITERS_COUNT, writers_threads, TRUE, INFINITE)
            ;
170
171     CloseHandle(mutex);
172     CloseHandle(can_read);
173     CloseHandle(can_write);
174
175     return 0;
176 }
```

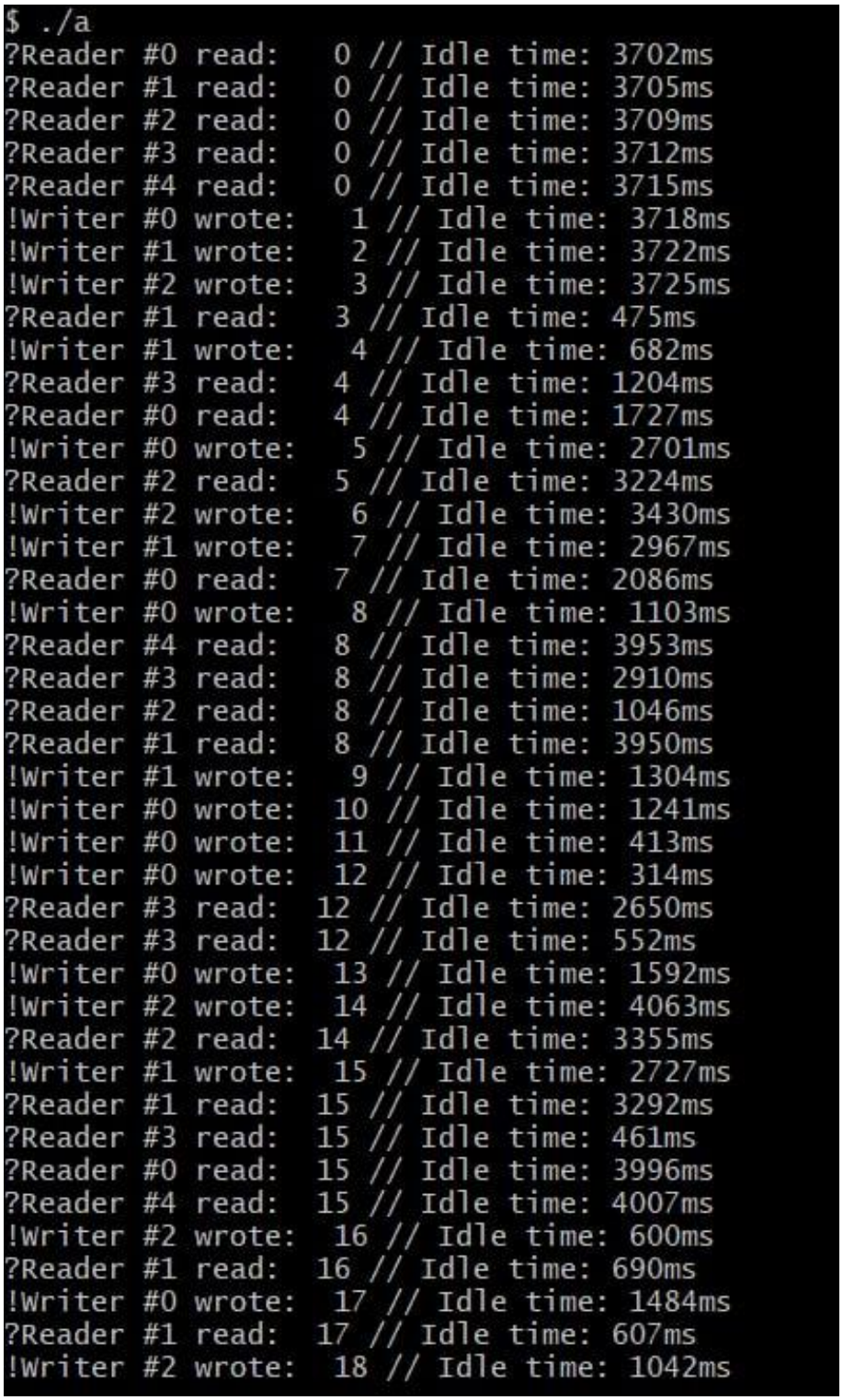

Рисунок 1 – Демонстрация работы программы