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 Реализация монитора Хоара «Читатели-писатели» под ОС `Windows`

Студент Пересторонин П.Г.

Группа ИУ7-53Б

Преподаватель Рязанова Н. Ю.

Москва — 2020 г.

Оглавление

1 Задача «Читатели-писатели» под ОС Windows

1.1 Вывод программы

На рисунке 1.1 показан вывод программы, реализующей монитор Хоара «Читатели-писатели» под ОС `Windows`.

```
Reader #1 read:      0 (slept 2915 ms)
Reader #0 read:      0 (slept 2912 ms)
Reader #4 read:      0 (slept 2925 ms)
Reader #3 read:      0 (slept 2922 ms)
Reader #2 read:      0 (slept 2918 ms)
Writer #2 write:     1 (slept 2935 ms)
Writer #1 write:     2 (slept 2931 ms)
Writer #0 write:     3 (slept 2928 ms)
Reader #4 read:      3 (slept  519 ms)
Reader #1 read:      3 (slept 1042 ms)
Writer #1 write:     4 (slept 1248 ms)
Reader #3 read:      4 (slept 1770 ms)
Reader #0 read:      4 (slept 2293 ms)
Writer #0 write:     5 (slept 2499 ms)
Writer #1 write:     6 (slept 2315 ms)
Reader #2 read:      6 (slept 3022 ms)
Writer #0 write:     7 (slept 1219 ms)
Writer #2 write:     8 (slept 3996 ms)
Reader #1 read:      8 (slept 3298 ms)
Reader #0 read:      8 (slept 2202 ms)
Writer #0 write:     9 (slept  776 ms)
Reader #3 read:      9 (slept 2259 ms)
Reader #2 read:      9 (slept 1163 ms)
Reader #4 read:      9 (slept 4123 ms)
Reader #4 read:      9 (slept  713 ms)
Reader #3 read:      9 (slept 1417 ms)
Writer #0 write:    10 (slept 1435 ms)
Reader #1 read:     10 (slept 2059 ms)
Reader #2 read:     10 (slept 2122 ms)
Reader #0 read:     10 (slept 2763 ms)
Reader #3 read:     10 (slept 1574 ms)
Writer #1 write:    11 (slept 4071 ms)
Writer #0 write:    12 (slept 2094 ms)
Writer #2 write:    13 (slept 4180 ms)
Writer #0 write:    14 (slept 1346 ms)
Reader #1 read:     14 (slept 1713 ms)
Reader #3 read:     14 (slept 2240 ms)
Reader #4 read:     14 (slept 3888 ms)
Reader #2 read:     14 (slept 3259 ms)
Reader #0 read:     14 (slept 3398 ms)
Writer #1 write:    15 (slept 3750 ms)
Writer #2 write:    16 (slept 3367 ms)
Reader #1 read:     16 (slept 2387 ms)
Reader #4 read:     16 (slept 2167 ms)
Writer #0 write:    17 (slept 2735 ms)
Reader #1 read:     17 (slept  896 ms)
Reader #3 read:     17 (slept 3121 ms)
Writer #1 write:    18 (slept 2021 ms)
Reader #2 read:     18 (slept 2314 ms)
Writer #2 write:    19 (slept 2064 ms)
Reader #0 read:     19 (slept 2460 ms)
Writer #1 write:    20 (slept  459 ms)
Writer #2 write:    21 (slept 1179 ms)
Reader #4 read:     21 (slept 2234 ms)
Reader #0 read:     21 (slept 1784 ms)
Reader #2 read:     21 (slept 4009 ms)
Writer #1 write:    22 (slept 3808 ms)
Writer #2 write:    23 (slept 3571 ms)
```

Рис. 1.1: Результат работы программы

1.2 Листинг кода

В листинге 1.1 представлен исходный код программы, реализующей монитор Хоара «Читатели-писатели» под ОС `Windows`.

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

#define READERS_AMOUNT 5
#define WRITERS_AMOUNT 3
#define WRITE_ITERATIONS 8
#define READ_ITERATIONS 7
#define WRITE_TIMEOUT 300
#define READ_TIMEOUT 300
#define DIFF 4000

#define CREATE_MUTEX_FAILED 1
#define CREATE_EVENT_FAILED 2
#define CREATE_THREAD_FAILED 3

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 value = 0;

void start_read(void) {
    InterlockedIncrement(&waiting_readers);
    // WaitForSingleObject(object, how_long_wait)
    if (active_writer || (WaitForSingleObject(can_write, 0) == WAIT_OBJECT_0 &&
        waiting_writers))
        WaitForSingleObject(can_read, INFINITE);
    // fake mutex
    WaitForSingleObject(mutex, INFINITE);
    InterlockedDecrement(&waiting_readers);
    InterlockedIncrement(&active_readers);
    SetEvent(can_read);
    ReleaseMutex(mutex);
}

void stop_read(void) {
```

```
42      InterlockedDecrement(&active_readers);
43      if (active_readers == 0) {
44          ResetEvent(can_read);
45          SetEvent(can_write);
46      }
47  }
48
49  DWORD WINAPI run_reader(CONST LPVOID lpParams) {
50      int index = (int)lpParams;
51      int sleep_time;
52      srand(time(NULL) + index);
53      for (size_t i = 0; i < READ_ITERATIONS; i++) {
54          sleep_time = READ_TIMEOUT + rand() % DIFF;
55          Sleep(sleep_time);
56          start_read();
57          printf("␣␣Reader␣#%ld␣read:␣␣%5ld␣(slept␣%4d␣ms)\n", index, value, sleep_time);
58          stop_read();
59      }
60      return 0;
61  }
62
63  void start_write(void) {
64      InterlockedIncrement(&waiting_writers);
65      if (active_writer || active_readers > 0)
66          WaitForSingleObject(can_write, INFINITE);
67      InterlockedDecrement(&waiting_writers);
68      active_writer = true;
69  }
70
71  void stop_write(void) {
72      active_writer = false;
73      if (waiting_readers)
74          SetEvent(can_read);
75      else
76          SetEvent(can_write);
77  }
78
79  DWORD WINAPI run_writer(CONST LPVOID lpParams) {
80      int index = (int)lpParams;
81      int sleep_time;
82      srand(time(NULL) + index + READERS_AMOUNT);
83      for (int i = 0; i < WRITE_ITERATIONS; ++i) {
84          sleep_time = WRITE_TIMEOUT + rand() % DIFF;
85          Sleep(sleep_time);
86          start_write();
87          ++value;
88          printf("␣␣Writer␣#%ld␣write:␣%5ld␣(slept␣%4d␣ms)\n", index, value, sleep_time);
89          stop_write();
```

```
90      }
91      return 0;
92  }
93
94  int main(void) {
95      setbuf(stdout, NULL);
96      HANDLE readers_threads[READERS_AMOUNT];
97      HANDLE writers_threads[WRITERS_AMOUNT];
98      // CreateMutex(attr, lock_now?, name) (attr have to be NULL (docs.windows))
99      if ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL) {
100         perror("Failed call of CreateMutex");
101         return CREATE_MUTEX_FAILED;
102     }
103     // CreateEvent(attr, manually?, init_state, name)
104     if ((can_read = CreateEvent(NULL, FALSE, FALSE, NULL)) == NULL
105             || (can_write = CreateEvent(NULL, FALSE, FALSE, NULL)) == NULL) {
106         perror("Failed call of CreateEvent");
107         return CREATE_EVENT_FAILED;
108     }
109
110     // CreateThread(attr, stack_size, begin_func, func_param, flags,
            pointer_where_to_return_id)
111     for (int i = 0; i < READERS_AMOUNT; ++i)
112         if ((readers_threads[i] = CreateThread(NULL, 0, run_reader, (LPVOID)i, 0, NULL))
                == NULL) {
113             perror("Failed call of CreateThread");
114             return CREATE_THREAD_FAILED;
115     }
116
117
118     for (int i = 0; i < WRITERS_AMOUNT; i++)
119         if ((writers_threads[i] = CreateThread(NULL, 0, run_writer, (LPVOID)i, 0, NULL))
                == NULL) {
120             perror("Failed call of CreateThread");
121             return CREATE_THREAD_FAILED;
122         }
123
124
125     // WaitForMultipleObjects(array_size, pointer_to_array, all?, how_long_wait)
126     WaitForMultipleObjects(READERS_AMOUNT, readers_threads, TRUE, INFINITE);
127     WaitForMultipleObjects(WRITERS_AMOUNT, writers_threads, TRUE, INFINITE);
128
129     CloseHandle(mutex);
130     CloseHandle(can_read);
131     CloseHandle(can_write);
132
133     return 0;
134 }
```

Листинг 1.1: монитор Хоара «Читатели-писатели» под ОС Windows