МГТУ им. Баумана

Лабораторная работа №6

По курсу: "Операционные системы"

# Реализация монитора Хоара «Читатели-писатели» под ОС Windows.

Работу выполнил: Мокеев Даниил, ИУ7-56

Преподаватель: Рязанова Н.Ю.

*Москва, 2020*

## 0.1 Листинг кода алгоритмов

Листинг 1: Реализация монитора Хоара

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

#define READERS 5
#define WRITERS 3
#define WRITERS_ITTER 4
#define SLEEP_TIME 200
#define READER_BORDER "\t\t\t\t\t\t"

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

HANDLE mutex;
HANDLE can_read;
HANDLE can_write;
volatile LONG waiting_writers = 0;
volatile LONG waiting_readers = 0;
volatile LONG active_readers = 0;
bool is_writer_active = false;

int value = 0;

void start_read(void) {
  InterlockedIncrement(&waiting_readers);
  if (is_writer_active || WaitForSingleObject(can_write, 0) == ↩
      WAIT_OBJECT_0) {
    WaitForSingleObject(can_read, INFINITE);
  }

  WaitForSingleObject(mutex, INFINITE);

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

  SetEvent(can_read);

  ReleaseMutex(mutex);
}

void stop_read(void) {
  InterlockedDecrement(&active_readers);

  if (waiting_readers == 0) {
    SetEvent(can_write);
```

```
45    }
46  }
47
48  DWORD WINAPI reader(LPVOID lpParams) {
49    while (value < 3 * WRITERS_ITTER) {
50      start_read();
51      printf(READER_BORDER"Reader #%ld; read value: %d\n", (int) lpParams,↩
            value);
52      stop_read();
53      Sleep(SLEEP_TIME);
54    }
55
56    return EXIT_SUCCESS;
57  }
58
59  void start_write(void) {
60    InterlockedIncrement(&waiting_writers);
61    if (is_writer_active || active_readers > 0) {
62      WaitForSingleObject(can_write, INFINITE);
63    }
64
65    InterlockedDecrement(&waiting_writers);
66    is_writer_active = true;
67    ResetEvent(can_write);
68  }
69
70  void stop_write(void) {
71    is_writer_active = false;
72
73    if (!waiting_writers) {
74      SetEvent(can_read);
75    } else {
76      SetEvent(can_write);
77    }
78  }
79
80  DWORD WINAPI writer(LPVOID lpParams) {
81    int i = 0;
82    for (int i = 0; i < WRITERS_ITTER; ++i) {
83      start_write();
84
85      value++;
86      printf("Writer #%ld wrote value: %ld\n", (int) lpParams, value);
87
88      stop_write();
89      Sleep(SLEEP_TIME);
90    }
91
92    return EXIT_SUCCESS;
```

```
93 }
94
95 int init_handles(void) {
96   if ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL) {
97     perror("error while CreateMutex");
98     return EXIT_FAILURE;
99   }
100
101   if ((can_read = CreateEvent(NULL, TRUE, TRUE, NULL)) == NULL) {
102     perror("error while CreateEvent can_read");
103     return EXIT_FAILURE;
104   }
105   if ((can_write = CreateEvent(NULL, FALSE, TRUE, NULL)) == NULL) {
106     perror("error while CreateEvent can_write");
107     return EXIT_FAILURE;
108   }
109
110   return EXIT_SUCCESS;
111 }
112
113 int create_threads(HANDLE *threads, int threads_count, DWORD (*on_thread↩
      )(LPVOID)) {
114   for (int i = 0; i < threads_count; ++i) {
115     if ((threads[i] = CreateThread(NULL, 0, on_thread, (LPVOID) i, 0, ↩
          NULL)) == NULL) {
116       perror("error while CreateThread");
117       return EXIT_FAILURE;
118     }
119   }
120
121   return EXIT_SUCCESS;
122 }
123
124 int main(void) {
125   setbuf(stdout, NULL);
126
127   int rc = EXIT_SUCCESS;
128
129   if ((rc = init_handles()) != EXIT_SUCCESS
130   || (rc = create_threads(writers, WRITERS, writer)) != EXIT_SUCCESS
131   || (rc = create_threads(readers, READERS, reader)) != EXIT_SUCCESS) {
132     return rc;
133   }
134
135   WaitForMultipleObjects(WRITERS, writers, TRUE, INFINITE);
136   WaitForMultipleObjects(READERS, readers, TRUE, INFINITE);
137
138   CloseHandle(mutex);
139   CloseHandle(can_read);
```

```
140     CloseHandle(can_write);
141
142     return rc;
143 }
```

```
C:\Users\daniil.mokeev\Desktop\operating-system\sem_5\lab06>main.exe
Writer #0 wrote value: 1
                                                Reader #2; read value: 1
                                                Reader #1; read value: 1
                                                Reader #0; read value: 1
Writer #1 wrote value: 2
                                                Reader #3; read value: 2
                                                Reader #4; read value: 2
Writer #2 wrote value: 3
Writer #0 wrote value: 4
                                                Reader #4; read value: 4
Writer #2 wrote value: 6
                                                Reader #0; read value: 4
                                                Reader #3; read value: 4
                                                Reader #2; read value: 4
Writer #1 wrote value: 5
                                                Reader #1; read value: 4
                                                Reader #4; read value: 6
Writer #0 wrote value: 7
Writer #1 wrote value: 8
                                                Reader #2; read value: 8
                                                Reader #1; read value: 8
                                                Reader #3; read value: 8
                                                Reader #0; read value: 8
Writer #2 wrote value: 9
                                                Reader #4; read value: 9
Writer #0 wrote value: 10
                                                Reader #2; read value: 10
                                                Reader #0; read value: 10
                                                Reader #3; read value: 10
                                                Reader #1; read value: 10
Writer #1 wrote value: 11
Writer #2 wrote value: 12
```

Рис. 1: Пример работы программы.