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 Богаченко А.Е.

Группа ИУ7-56Б

Оценка (баллы) ______________________________

Преподаватели Рязанова Н. Ю.

Москва — 2020 г.

Задача «Читатели-Писатели»

Листинги кода

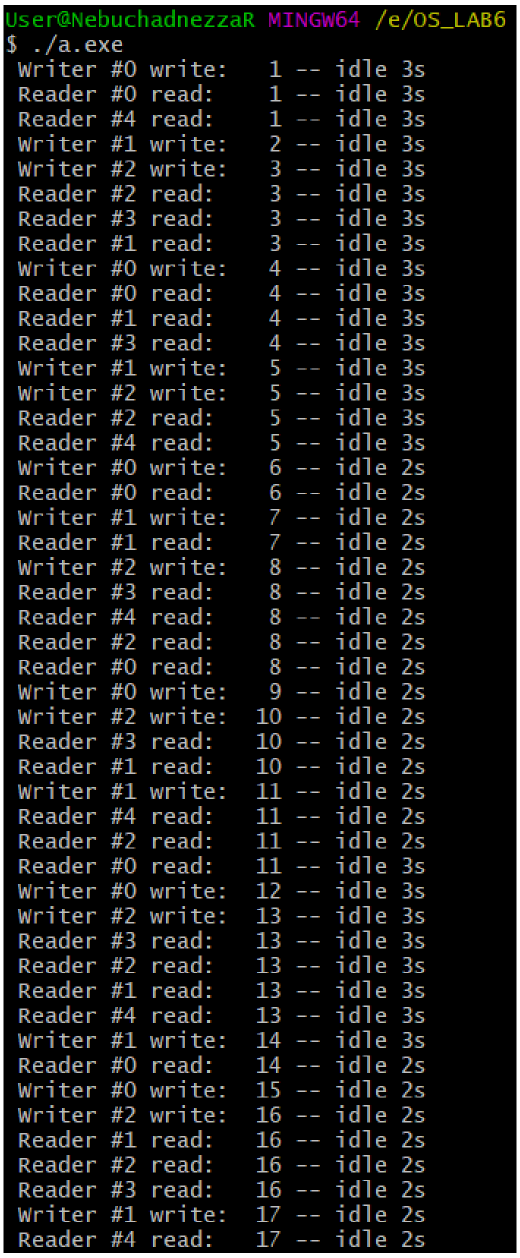
```c
#include <stdbool.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <windows.h>

#define READERS_COUNT 5
#define WRITERS_COUNT 3
#define ITERATIONS 10
#define MAX_RANDOM 3

HANDLE writers[WRITERS_COUNT];
HANDLE readers[READERS_COUNT];

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;

 static __volatile__ int value = 0;

 void read_start(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);
}

```

```
static inline void read_stop(void) {
    InterlockedDecrement(&active_readers);
    if (waiting_readers == 0) {
        SetEvent(can_write);
    }
}

DWORD WINAPI reader(LPVOID lpParams) {
    while (value < WRITERS_COUNT * ITERATIONS) {
        int sleep_time = rand() % MAX_RANDOM + 1;
        sleep(sleep_time);

        // !!! --- CRITICAL --- !!!
        read_start();
        printf(" Reader #%d read:  %3d -- idle %ds\n",
                        (int)lpParams,
                        value,
                        sleep_time);
        read_stop();
        // !!! --- CRITICAL --- !!!
    }

    return EXIT_SUCCESS;
}

void write_start(void) {
    InterlockedIncrement(&waiting_writers);
    if (is_writer_active || active_readers > 0) {
        WaitForSingleObject(can_write, INFINITE);
    }

    InterlockedDecrement(&waiting_writers);
    is_writer_active = true;
    ResetEvent(can_write);
}

static inline void write_stop(void) {
    is_writer_active = false;

    if (!waiting_writers) {
        SetEvent(can_read);
    } else {
        SetEvent(can_write);
    }
}

DWORD WINAPI writer(LPVOID lpParams) {
    for (short i = 0; i < ITERATIONS; ++i) {
        int sleep_time = rand() % MAX_RANDOM + 1;
        sleep(sleep_time);

        // !!! --- CRITICAL --- !!!
        write_start();
        ++value;
        printf(" Writer #%d write: %3d -- idle %ds\n",
                    (int)lpParams,
                    value,
                    sleep_time);
        write_stop();
        // !!! --- CRITICAL --- !!!
    }

    return EXIT_SUCCESS;
}

int init(void) {
    if ((mutex = CreateMutex(NULL, FALSE, NULL)) == NULL) {
        perror("create mutex error!");
        return EXIT_FAILURE;
    }

    if ((can_read = CreateEvent(NULL, FALSE, TRUE, NULL)) == NULL) {
        perror("create event `can_read` error!");
        return EXIT_FAILURE;
    }
    if ((can_write = CreateEvent(NULL, TRUE, TRUE, NULL)) == NULL) {
        perror("create event `can_write` error!");
        return EXIT_FAILURE;
    }

    return EXIT_SUCCESS;
}

int create_threads(HANDLE *threads,
        int threads_count,
                   DWORD (*on_thread)(LPVOID)) {
    for (short i = 0; i < threads_count; ++i) {
        if ((threads[i] = CreateThread(NULL, 0, on_thread, (LPVOID)i, 0,
                                       NULL)) == NULL) {
            perror("create thread error!");
            return EXIT_FAILURE;
        }
    }

    return EXIT_SUCCESS;
}
```

```c

int main(void) {
    setbuf(stdout, NULL);

    if (init() != EXIT_SUCCESS ||
        create_threads(writers, WRITERS_COUNT, writer) != EXIT_SUCCESS ||
        create_threads(readers, READERS_COUNT, reader) != EXIT_SUCCESS) {
        return EXIT_FAILURE;
    }

    WaitForMultipleObjects(WRITERS_COUNT, writers, TRUE, INFINITE);
    WaitForMultipleObjects(READERS_COUNT, readers, TRUE, INFINITE);

    CloseHandle(mutex);
    CloseHandle(can_read);
    CloseHandle(can_write);

    return EXIT_SUCCESS;
}
```

Листинг 1 – Реализация задачи.

Работа программы



```
User@NebuchadnezzaR MINGW64 /e/OS_LAB6
$ ./a.exe
 Writer #0 write:   1 -- idle 3s
 Reader #0 read:    1 -- idle 3s
 Reader #4 read:    1 -- idle 3s
 Writer #1 write:   2 -- idle 3s
 Writer #2 write:   3 -- idle 3s
 Reader #2 read:    3 -- idle 3s
 Reader #3 read:    3 -- idle 3s
 Reader #1 read:    3 -- idle 3s
 Writer #0 write:   4 -- idle 3s
 Reader #0 read:    4 -- idle 3s
 Reader #1 read:    4 -- idle 3s
 Reader #3 read:    4 -- idle 3s
 Writer #1 write:   5 -- idle 3s
 Writer #2 write:   5 -- idle 3s
 Reader #2 read:    5 -- idle 3s
 Reader #4 read:    5 -- idle 3s
 Writer #0 write:   6 -- idle 2s
 Reader #0 read:    6 -- idle 2s
 Writer #1 write:   7 -- idle 2s
 Reader #1 read:    7 -- idle 2s
 Writer #2 write:   8 -- idle 2s
 Reader #3 read:    8 -- idle 2s
 Reader #4 read:    8 -- idle 2s
 Reader #2 read:    8 -- idle 2s
 Reader #0 read:    8 -- idle 2s
 Writer #0 write:   9 -- idle 2s
 Writer #2 write:  10 -- idle 2s
 Reader #3 read:   10 -- idle 2s
 Reader #1 read:   10 -- idle 2s
 Writer #1 write:  11 -- idle 2s
 Reader #4 read:   11 -- idle 2s
 Reader #2 read:   11 -- idle 2s
 Reader #0 read:   11 -- idle 3s
 Writer #0 write:  12 -- idle 3s
 Writer #2 write:  13 -- idle 3s
 Reader #3 read:   13 -- idle 3s
 Reader #2 read:   13 -- idle 3s
 Reader #1 read:   13 -- idle 3s
 Reader #4 read:   13 -- idle 3s
 Writer #1 write:  14 -- idle 3s
 Reader #0 read:   14 -- idle 2s
 Writer #0 write:  15 -- idle 2s
 Writer #2 write:  16 -- idle 2s
 Reader #1 read:   16 -- idle 2s
 Reader #2 read:   16 -- idle 2s
 Reader #3 read:   16 -- idle 2s
 Writer #1 write:  17 -- idle 2s
 Reader #4 read:   17 -- idle 2s
```

Рисунок 1 – «Читатели-Писатели». Максимальная задержка – 3с.