## инистерство 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)

**ОТЧЕТ**

**по лабораторной работе №** 6

**Название:**  «Реализация монитора Хоара «Читатели- писатели» под ОС Windows»

**Дисциплина: Операционные системы**

|  |  |  |  |
| --- | --- | --- | --- |
| Студент | ИУ7-56Б  (Группа) | (Подпись, дата) | Ж. Р.Турсунов  (И.О. Фамилия) |
| Преподаватель |  | (Подпись, дата) | Н.Ю.Рязанова  (И.О. Фамилия) |

Москва, 2020

# Задание 1: Реализация монитора Хоара «Читатели-писатели» под ОС Windows

# Листинг кода:

#include <stdio.h>

#include <time.h>

#include <windows.h>

volatile LONG nr **=** 0**;**

volatile BOOL wrt **=** 0**;**

volatile LONG waiting\_readers **=** 0**;**

volatile LONG waiting\_writers **=** 0**;**

HANDLE c\_read**;**

HANDLE c\_write**;**

HANDLE rwMutex**;**

#define READERS 5

#define WRITERS 4

#define RW\_COUNT (READERS + WRITERS)

#define ITERS 10

#define MAX\_READER\_SLEEP 3000

#define MAX\_WRITER\_SLEEP 1700

unsigned short int buffer **=** 0**;**

void start\_read**()**

**{**

WaitForSingleObject**(**rwMutex**,** INFINITE**);**

InterlockedIncrement**(&**waiting\_readers**);**

**if** **(**wrt **||** waiting\_writers**)**

WaitForSingleObject**(**c\_read**,** INFINITE**);**

ResetEvent**(**c\_read**);**

InterlockedDecrement**(&**waiting\_readers**);**

InterlockedIncrement**(&**nr**);**

SetEvent**(**c\_read**);**

ReleaseMutex**(**rwMutex**);**

**}**

void stop\_read**()**

**{**

InterlockedDecrement**(&**nr**);**

**if** **(**nr **==** 0**)**

SetEvent**(**c\_write**);**

**}**

void start\_write**()**

**{**

WaitForSingleObject**(**rwMutex**,** INFINITE**);**

InterlockedIncrement**(&**waiting\_writers**);**

**if** **(**nr **>** 0 **||** wrt**)**

WaitForSingleObject**(**c\_write**,** INFINITE**);**

ResetEvent**(**c\_write**);**

InterlockedDecrement**(&**waiting\_writers**);**

wrt **=** 1**;**

ReleaseMutex**(**rwMutex**);**

**}**

void stop\_write**()**

**{**

wrt **=** 0**;**

**if** **(**waiting\_readers **>** 0**)**

SetEvent**(**c\_read**);**

**else**

SetEvent**(**c\_write**);**

**}**

// Читатель

DWORD reader**(**LPVOID lpParameter**)**

**{**

int reader\_id **=** **\*((**int **\*)**lpParameter**);**

**while** **(**buffer **<** ITERS **\*** WRITERS**)**

**{**

Sleep**(**rand**()** **%** MAX\_READER\_SLEEP**);**

start\_read**();**

printf**(**"\tReader #%d read: %d\n"**,** reader\_id**,** buffer**);**

stop\_read**();**

**}**

**return** 0**;**

**}**

// Писатель

DWORD writer**(**LPVOID lpParameter**)**

**{**

int writer\_id **=** **\*((**int **\*)**lpParameter**);**

**for** **(**int i **=** 0**;** i **<** ITERS**;** i**++)**

**{**

Sleep**(**rand**()** **%** MAX\_WRITER\_SLEEP**);**

start\_write**();**

**++**buffer**;**

printf**(**"Writer #%d wrote: %d\n"**,** writer\_id**,** buffer**);**

stop\_write**();**

**}**

**return** 0**;**

**}**

int main**()**

**{**

HANDLE threads**[**RW\_COUNT**]** **=** **{**0**};**

int id**[**RW\_COUNT**]** **=** **{**0**};**

rwMutex **=** CreateMutex**(NULL,** FALSE**,** **NULL);**

**if** **(**rwMutex **==** **NULL)**

**{**

perror**(**"Error: CreateMutex!\n"**);**

**return** EXIT\_FAILURE**;**

**}**

c\_read **=** CreateEvent**(NULL,** FALSE**,** TRUE**,** **NULL);**

**if** **(**c\_read **==** **NULL)**

**{**

perror("Error: CreateEvent for can\_read!\n");

return EXIT\_FAILURE;

}

c\_write = CreateEvent(NULL, TRUE, TRUE, NULL);

if (c\_write == NULL)

{

perror("Error: CreateEvent for can\_write!\n");

return EXIT\_FAILURE;

}

for (int i = 0; i < RW\_COUNT; ++i)

{

if (i < READERS)

{

id[i] = i; threads[i] = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)&reader, (id + i), 0, NULL);

if (threads[i] == NULL)

{

perror("Error: CreateThread readers!\n");

return EXIT\_FAILURE;

}

}

else

{

id[i] = i - READERS;

threads[i] = CreateThread(NULL, 0, (LPTHREAD\_START\_ROUTINE)&writer, (id + i), 0, NULL);

if (threads[i] == NULL)

{

perror("Error: CreateThread writers!\n");

return EXIT\_FAILURE;

}

}

}

WaitForMultipleObjects(RW\_COUNT, threads, TRUE, INFINITE);

CloseHandle(rwMutex);

CloseHandle(c\_read);

CloseHandle(c\_write);

return EXIT\_SUCCESS;

}

# Полученный результат:

# 