Министерство образования и науки Российской Федерации

Севастопольский государственный университет

Институт информационных технологий и управления в технических системах

ОТЧЕТ

по лабораторной работе №4

Синхронизация процессов и потоков в ОС Windows.

Семафоры.

Выполнил: ст. гр. ИС/б-21

Куркчи А.Е.

Проверил:

Шишкевич В.Е.

Севастополь

2015

# Цель

# Изучение программного интерфейса приложений(API) операционных систем Windows 9x, NT, ME, XP, 7. Приобретение практических навыков синхронизации потоков, с использованием семафоров в средах программирования Borland Delphi, C++ Builder или Visual Studio.

1. Постановка задачи
   1. Написать программу, содержащую два потока. Первый поток генерирует последовательность чисел и помещает их в кольцевой\* буфер из Nbuf элементов (с проверкой на свободное место в буфере с использованием механизма семафоров). Второй считывает данные из буфера и выводит их на экран.

\*) при заполнении кольцевого буфера добавление элементов продолжается сначала, т.е. для вычисления индекса очередного элемента используется операция деления по модулю Nbuf(Pascal: i mod Nbuf; Cи: I % Nbuf).

Количество элементов, тип последовательности и длина буфера выбираются в соответствии с вариантом задания (таблица 3.1).

* 1. Определить максимальную длину буфера при различных приоритетах потоков.

1. Текст программ

Программа Lab.04:

#include "stdafx.h"

**using** **namespace** std**;**

const int N **=** 90**;**

const int Nbuf **=** 10**;**

const int d **=** 4**;**

int buf**[**Nbuf**];**

long maxWriter **=** 0**;**

long maxReader **=** 0**;**

HANDLE writerSemaphore**;**

HANDLE readerSemaphore**;**

DWORD WINAPI writerThread**(**const LPVOID lpParam**)** **{**

**for(**int i**=**0**;**i**<**N**;**i**++)** **{**

WaitForSingleObject**(**writerSemaphore**,**INFINITE**);**

buf**[**i**%**Nbuf**]** **=** **(**i**+**1**)\***d**;**

long prev**;**

ReleaseSemaphore**(**readerSemaphore**,**1**,&**prev**);**

**if(**prev**+**1 **>** maxReader**)** **{**

maxReader **=** prev**+**1**;**

**}**

**}**

**return** 0**;**

**}**

DWORD WINAPI readerThread**(**const LPVOID lpParam**)** **{**

**for(**int i**=**0**;**i**<**N**;**i**++)** **{**

WaitForSingleObject**(**readerSemaphore**,**INFINITE**);**

cout **<<** buf**[**i**%**Nbuf**]** **<<** " "**;**

long prev**;**

ReleaseSemaphore**(**writerSemaphore**,**1**,&**prev**);**

**if(**prev**+**1 **>** maxWriter**)** **{**

maxWriter **=** prev**+**1**;**

**}**

**}**

**return** 0**;**

**}**

int main**(**int argc**,** char**\*** argv**[])** **{**

int pid **=** GetCurrentProcessId**();**

cout **<<** "\tMaster" **<<** endl**;**

cout **<<** "PID: " **<<** pid **<<** endl**;**

cout **<<** endl**;**

cout**.**flush**();**

writerSemaphore **=** CreateSemaphore**(NULL,**Nbuf**,**Nbuf**,**"writerSemaphore"**);**

readerSemaphore **=** CreateSemaphore**(NULL,**0**,**Nbuf**,**"readerSemaphore"**);**

HANDLE hWriterThread **=** CreateThread**(NULL,**0**,&**writerThread**,NULL,**CREATE\_SUSPENDED**,NULL);**

HANDLE hReaderThread **=** CreateThread**(NULL,**0**,&**readerThread**,NULL,**CREATE\_SUSPENDED**,NULL);**

SetThreadPriority**(**hWriterThread**,**THREAD\_PRIORITY\_IDLE**);**

SetThreadPriority**(**hReaderThread**,**THREAD\_PRIORITY\_HIGHEST**);**

ResumeThread**(**hWriterThread**);**

ResumeThread**(**hReaderThread**);**

WaitForSingleObject**(**hReaderThread**,**INFINITE**);**

CloseHandle**(**hWriterThread**);**

CloseHandle**(**hReaderThread**);**

cout **<<** endl**;**

cout **<<** "Semaphores max count:" **<<** endl**;**

cout **<<** "Writer: " **<<** maxWriter **<<** "; Reader: " **<<** maxReader **<<** ";" **<<** endl**;**

cout **<<** "Pause" **<<** endl**;**

system**(**"pause"**);**

**return** 0**;**

**}**

1. Результат

Результат работы программы Lab.04 с приоритетом у читателя больше, чем у писателя показан на рисунке 1.

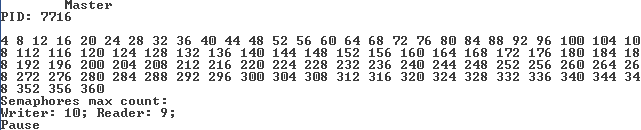


Рисунок 1 – запуск Lab.04

Результат работы программы Lab.04 с приоритетом у писателя больше, чем у читателя показан на рисунке 2.

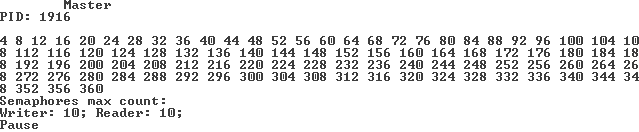


Рисунок 2 – запуск Lab.04

Выводы

# В ходе лабораторной работы были изучены методы синхронизации между процессами и потоками средствами Semaphore WinAPI и языка Visual C++. Semaphore используется для решения задачи параллельных вычислений и в частности параллельного доступа к буферу (задача поставщика-потребителя).