МИНИСТЕРСТВО ВЫСШЕГО ОБРАЗОВАНИЯ И НАУКИ РФ

филиал федерального государственного бюджетного образовательного учреждения высшего образования «Национальный исследовательский университет «МЭИ» в г. Смоленске

Кафедра вычислительной техники

Отчет по лабораторной работе №7

по курсу: «Операционные системы»

Тема: «ОБЪЕКТЫ СИНХРОНИЗАЦИИ»

Группа: ИВТ1-18

Студент: Федченков М. В.

Преподаватель: Гаврилов А. И.

Вариант: № 15

Смоленск, 2019 г.

1. **Цель работы**

Целью данной работы является исследование объектов синхронизации, с помощью которых в многозадачной среде обеспечивается последовательный доступ к совместно используемым ресурсам. В данной работе рассматриваются следующие вопросы:

• Синхронизация потоков с помощью объектов пользовательского режима (критические секции);

• Синхронизация потоков с помощью объектов ядра (объекты Mutex, события, семафоры, процессы и потоки);

• Работа Wait- функций в различных режимах.

1. **Задание на лабораторную работу**

Задача "Производитель-потребитель". Взаимодействуют два процесса с жестко распределенными между ними функциями. Один процесс вырабатывает сообщения, предназначенные для восприятия и обработки другим процессом. Процесс, вырабатывающий сообщения, называют *производителем*, а воспринимающий сообщения — *потребителем*. Процессы взаимодействуют через некоторую обобщенную область памяти, которая по смыслу является критическим ресурсом. В эту область процесс-производитель должен помещать очередное сообщение (предполагается, что область способна хранить только одно сообщение), а процесс-потребитель должен считывать очередное сообщение. Необходимо согласовать работы двух процессов при одностороннем обмене сообщениями таким образом, чтобы удовлетворить следующим требованиям:

выполнять требования задачи взаимного исключения по отношению к критическому ресурсу — обобщенной памяти для хранения сообщения

учитывать состояние обобщенной области памяти, характеризующей возможность или невозможность посылки (принятия) очередного сообщения

Попытка процесса-производителя поместить очередное сообщение в область, из которой не было считано предыдущее сообщение процессом-потребителем, должна быть блокирована. Процесс-производитель должен быть переведен в состояние ожидания возможности поместить очередное сообщение через некоторое время в область памяти, по мере ее освобождения. Аналогично должна быть блокирована попытка процесса-потребителя считать сообщение из области в ситуации, когда процесс-производитель не поместил туда очередного сообщения.

1. **Анализ задания на лабораторную работу**

Нужно создать два приложения. Так чтобы, одно было сервером, а второе клиентом. Так же они будут взаимодействовать с одним файлом для односторонней передачи информации, при помощи mutex и потоков.

1. **Модульная структура программы**

TForm1 = class(TForm)

Button1: TButton;

Edit1: TEdit;

procedure Button1Click(Sender: TObject);

1. **Алгоритм**

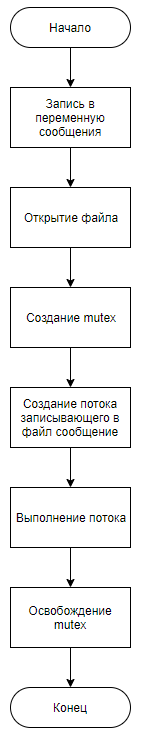


Рисунок 1 – Алгоритм программы Server

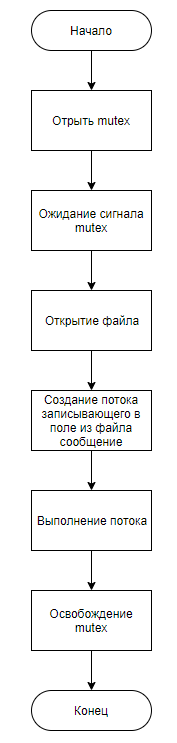


Рисунок 2 – Алгоритм программы Client

1. **Результат работы**

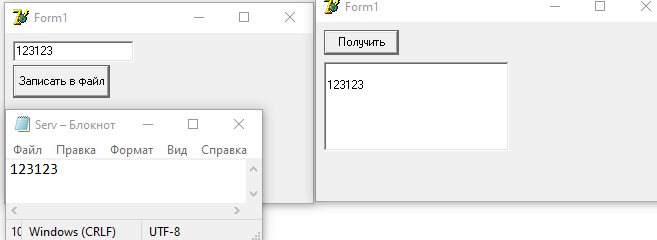


Рисунок 3 – Результат работы программ

1. **Контрольные вопросы**
2. Какие объекты ядра можно использовать как объекты синхронизации?

Для синхронизации можно использовать следующие объекты ядра:процессы, потоки, задания, файлы, консольный ввод, уведомления об изменении файлов, события, ожидаемые таймеры, семафоры, мьютексы.

1. Когда объект ядра «процесс» находится в свободном, а когда в занятом состоянии?

Объекты ядра «процесс» сразу после создания всегда находятся в занятом состоянии. В момент завершения процесса операционная система автоматически освобождает его объект ядра «процесс», и он навсегда остается в этом состоянии.

1. Как потоки различных процессов могут использовать один и тот же объект семафор, мьютекс или событие для синхронизации?

Они могут обращаться к ним по уникальному имени, указываемому в параметре lpName функций CreateSemaphore, CreateMutex, CreateEvent.

1. Чем объект мьютекс отличается от других объектов синхронизации?

Объект мьютекс отличается от остальных объектов ядра тем, что занявшему его потоку передаются права на владение им. Объекты мьютексы способны запоминать, какому потоку они принадлежат. Если какой-то посторонний поток попытается освободить мьютекс вызовом функции ReleaseMutex, то данная функция вернет FALSE. Вызов функции GetLastError даст значение ERROR\_NOT\_OWNER.

1. Чем объект семафор отличается от других объектов синхронизации?

Объекты ядра «семафор» используются для учета ресурсов. Как и все объекты ядра, они содержат счетчик числа пользователей, но, кроме того, поддерживают два 32 битных значения со знаком: одно определяет максимальное число ресурсов (контролируемое семафором), другое используется как счетчик текущего числа ресурсов.

1. В каких случаях следует использовать события со сбросом вручную, а в каких события с автосбросом?

Первые позволяют возобновлять выполнение сразу нескольких ждущих потоков, вторые — только одного, следовательно, и использовать их следует в зависимости от того, какое количество потоков нужно возобновлять.

**Приложение**

**А. Текст программы Server**

unit Server;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls;

type

TForm1 = class(TForm)

Button1: TButton;

Edit1: TEdit;

procedure Button1Click(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

mt:THANDLE;

f:TextFile;

Str:PChar;

implementation

{$R \*.dfm}

procedure Thread1;

begin

WaitForSingleObject(mt,INFINITE);

Writeln(f,Str);

ReleaseMutex(mt);

Sleep(20);

CloseFile(f);

end;

procedure TForm1.Button1Click(Sender: TObject);

var

hThread1,hThread2:THandle;

pFunc1,pFunc2:pointer;

ThreadID1,ThreadID2:CARDINAL;

begin

Str:=PChar(Edit1.Text+#0);

AssignFile(f,'D:\Serv.txt'); //Çàìåíèòü íà ñâîé ïóòü

Rewrite(f);

//Ñîçäàíèå mutex

mt:=CreateMutex(nil,FALSE,'Serv');

pFunc1:=@Thread1;

hThread1:=CreateThread(nil,0,pFunc1,nil,0,ThreadID1);

end;

end.

**Б. Текст программы Client**

unit Client;

interface

uses

Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,

Dialogs, StdCtrls;

type

TForm1 = class(TForm)

Button1: TButton;

Memo1: TMemo;

procedure Button1Click(Sender: TObject);

procedure FormCreate(Sender: TObject);

private

{ Private declarations }

public

{ Public declarations }

end;

var

Form1: TForm1;

mt:Thandle;

f:TextFile;

buf: string[80];

implementation

{$R \*.dfm}

procedure Thread3;

begin

WaitForSingleObject(mt,INFINITE);

while not EOF(f) do

begin

readln(f, buf);

end;

ReleaseMutex(mt);

Sleep(20);

CloseFile(f);

end;

procedure TForm1.Button1Click(Sender: TObject);

var hThread3:THandle;

pFunc3:pointer;

ThreadID3:CARDINAL;

begin

mt:=OpenMutex(MUTEX\_ALL\_ACCESS, FALSE,'Servv');

WaitForSingleObject(mt,INFINITE);

AssignFile(f,'D:\Serv.txt'); //Заменить на свой путь

Reset(f);

pFunc3:=@Thread3;

hThread3:=CreateThread(nil,0,pFunc3,nil,0,ThreadID3);

Memo1.Lines.Add(buf);

end;

procedure TForm1.FormCreate(Sender: TObject);

begin

Memo1.Clear();

end;

end.