МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №4**

**з дисципліни «Операційні системи»**

**на тему: «Проекція файлу. Завдання синхронізації.»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Адрющенко В. О.

Дніпро, 2020

**Лабораторна робота №4**

**Тема.** Проекція файлу. Завдання синхронізації.

**Завдання.** Розробити програми, які моделюють рішення задачі синхронізації з використанням заданих системних ресурсів. Процес обмінюються даними через проекцію файлу. Програма повина бути реалізована як графічний додаток.

**Варіант 13.**

1) Взаємовиключний доступ

1) Подія

**Текст програми**

main.cpp

#include <windows.h>

#include <stdio.h>

#define THREADCOUNT 4

HANDLE ghWriteEvent;

HANDLE ghThreads[THREADCOUNT];

DWORD WINAPI ThreadProc(LPVOID);

void CreateEventsAndThreads(void) {

int i;

DWORD dwThreadID;

// Создаем объект события с ручным сбросом. Поток записи устанавливает это

// объект в сигнальное состояние, когда он заканчивает запись в общий буфер

ghWriteEvent = CreateEvent(

NULL, // атрибуты безопасности по умолчанию

TRUE, // событие ручного сброса

FALSE, // начальное состояние не сигнализируется

TEXT("WriteEvent") // имя объекта

);

if (ghWriteEvent == NULL) {

printf("CreateEvent failed (%d)\n", GetLastError());

return;

}

// Создаем несколько потоков для чтения из буфера

for (i = 0; i < THREADCOUNT; i++) {

// TODO: более сложные сценарии могут потребовать использования параметра

// в процедуру потока, например, событие в потоке для

// использоваться для синхронизации.

ghThreads[i] = CreateThread(

NULL, // безопасность по умолчанию

0, // размер стека по умолчанию

ThreadProc, // имя функции потока

NULL, // нет параметров потока

0, // флаги запуска по умолчанию

&dwThreadID);

if (ghThreads[i] == NULL) {

printf("CreateThread failed (%d)\n", GetLastError());

return;

}

}

}

void WriteToBuffer(VOID) {

// ЗАДАЧА: Записать в общий буфер.

printf("Main thread writing to the shared buffer...\n");

// Установить ghWriteEvent в сигнальное

if (!SetEvent(ghWriteEvent)) {

printf("SetEvent failed (%d)\n", GetLastError());

return;

}

}

void CloseEvents() {

// Закрыть все дескрипторы событий (в настоящее время только один глобальный дескриптор).

CloseHandle(ghWriteEvent);

}

int main(void) {

DWORD dwWaitResult;

// TODO: создать общий буфер

// Создаем события и потоки THREADCOUNT для чтения из буфера

CreateEventsAndThreads();

// В этот момент потоки чтения запущены и наиболее

// вероятно ожидание сигнала глобального события. Тем не мение,

// запись в буфер безопасна, потому что это событие

// событие ручного сброса.

WriteToBuffer();

printf("Main thread waiting for threads to exit...\n");

// Дескриптор каждого потока получает сигнал, когда поток

// завершено.

dwWaitResult = WaitForMultipleObjects(

THREADCOUNT, // количество handles в массиве

ghThreads, // массив дескрипторов потоков

TRUE, // подождите, пока все не будут сигнализированы

INFINITE);

switch (dwWaitResult) {

// Все объекты потока получили сигнал

case WAIT\_OBJECT\_0:

printf("All threads ended, cleaning up for application exit...\n");

break;

// Произошла ошибка

default:

printf("WaitForMultipleObjects failed (%d)\n", GetLastError());

return 1;

}

Sleep(10000);

// Закрыть события для очистки

CloseEvents();

return 0;

}

DWORD WINAPI ThreadProc(LPVOID lpParam) {

// lpParam в этом примере не используется

UNREFERENCED\_PARAMETER(lpParam);

DWORD dwWaitResult;

printf("Thread %d waiting for write event...\n", GetCurrentThreadId());

dwWaitResult = WaitForSingleObject(

ghWriteEvent, // event handle

INFINITE); // неопределенное ожидание

switch (dwWaitResult)

{

// Объект события получил сигнал

case WAIT\_OBJECT\_0:

// TODO: прочитать из общего буфера

printf("Thread %d reading from buffer\n",

GetCurrentThreadId());

break;

// ошибка

default:

printf("Wait error (%d)\n", GetLastError());

return 0;

}

// Теперь, когда мы закончили чтение буфера, мы можем использовать другой

// событие, сигнализирующее о том, что этот поток больше не читает. Этот

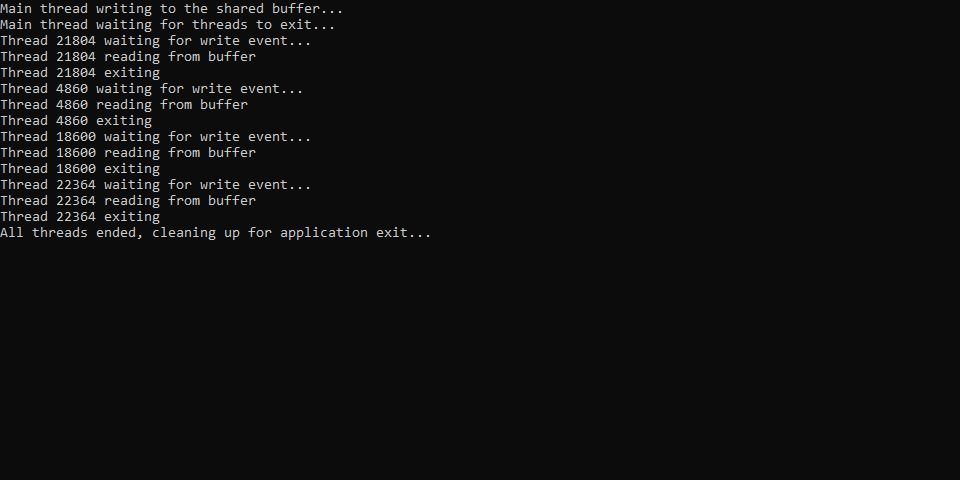
// пример просто использует дескриптор потока для синхронизации (

// дескриптор сигнализируется, когда поток завершается.)

printf("Thread %d exiting\n", GetCurrentThreadId());

return 1;

}



Висновок: Розроблена програма демонструє вирішення задачі синхронізації “ Взаємовиключний доступ ” із використанням стандартних функцій операційної системи Windows. Графічний інтерфейс розробленій за допомогою звичайної консолі в Microsoft Visual Studio.