# Лабораторная работа № 5. События

## Функции для работы с событиями

**События** (*events*) являются средством условной синхронизации. Они могут находиться в двух состояниях – «событие произошло» и «событие не произошло». Фактически, поток, приостановленный на событии, ожидает наступления ситуации «событие произошло». Для обеспечения возможности многократного использования событий потоки могут заново инициализировать события, переводя их в состояние «событие не произошло».

События являются системными объектами Microsoft Windows и могут быть использованы для синхронизации потоков, определенных в контексте разных процессов. Для этого используются поименованные события. Анонимные события используются для синхронизации потоков в рамках одного процесса или для синхронизации потоков в процессах-предках и процессах-потомках, если процессы потомки наследуют указатели (handle) событий.

Для создания событий используется функция **CreateEvent**:

HANDLE CreateEvent(

LPSECURITY\_ATTRIBUTES lpEventAttributes,

BOOL bManualReset,

BOOL bInitialState,

LPCTSTR lpName

);

Параметры lpEventAttributes и lpName аналогичны таким же параметрам мьютекса и семафоров.

Параметр bManualReset используется для указания режима переключения события в состояние «событие не произошло». Если данный параметр равен true, то используется режим ручного переключения события, т.е. для того, чтобы событие перешло в состояние «событие не произошло», необходимо выполнить функцию ResetEvent. Значение параметра bManualReset, равное false означает, что переключение события в режим «событие не произошло» будет осуществляться автоматически. То есть всякий раз, когда какой-либо поток при помощи функции WaitForSingleObject или WaitForMultipleObjects дождется перехода события в состояние «событие произошло» и войдет в свою критическую секцию, событие автоматически перейдет в состояние «событие не произошло», а следовательно, другой поток тут же сможет войти в свою критическую секцию, если он был приостановлен на событии

Параметр bInitialState показывает состояние, в котором создается событие. Если он равен true, то событие создается как означенное, т.е. в состоянии «событие произошло», а если данный параметр равен false, то событие создается как неозначенное, т.е. в состоянии «событие не произошло».

Возвращаемые значения функции CreateEvent:

1. Если создание события прошло успешно, то функция вернет значение указателя (handle) созданного события.

2. Если во время создания события произошла ошибка, то функция вернет значение NULL, а более подробно об ошибке можно узнать, обратившись к функции GetLastError.

3. Если при создании поименованного события окажется, что событие с таким именем уже существует, то функция все равно вернет его указатель (handle), а значением функции GetLastError будет код ошибки ERROR\_ALREADY\_EXISTS.

Открыть уже созданное событие можно при помощи функции **OpenEvent**:

HANDLE OpenEvent(

DWORD dwDesiredAccess,

BOOL bInheritHandle,

LPCTSTR lpName

);

Входные параметры этой функции аналогичны параметрам функций открытия мьютекса и семафора. В случае успешного открытия объекта события функция возвращает его указатель, а в противном случае – значение NULL.

Дождаться перехода события в состояние «событие произошло» можно при помощи функции WaitForSingleObject:

DWORD WaitForSingleObject(HANDLE hHandle,DWORD dwMilliseconds);

Если же потоку необходимо дождаться означивания нескольких событий (или одного из нескольких событий), то можно воспользоваться функцией WaitForMultipleObjects.

Значение параметров данных функций аналогично тем, которые использовались при работе с потоками, за исключением того, что в качестве дескриптора hHandle используется дескриптор события, и вместо массива дескрипторов потоков lpHandles используется массив дескрипторов событий. Кроме того, если событие создано с автоматическим режимом повторной инициализации, то функции WaitForSingleObject и WaitForMultipleObjects не только пропускают поток в его критическую секцию при означивании события, но и переводят событие в состояние «событие не произошло».

Ручной перевод события в неозначенное состояние выполняется функцией ResetEvent:

BOOL ResetEvent(HANDLE hEvent);

В качестве параметра данной функции передается указатель события. В случае успешного перевода события в состояние «событие не произошло» функция возвращает значение true, в противном случае – значение false. При этом подробности об ошибке можно получить при помощи функции GetLastError.

## Пример программы

***Задача****.* Реализовать модель выполнения банковским счетом клиента при снятии наличных средств. Первый поток должен позволять указывать сумму денежных средств, снимаемых клиентом. Второй поток уменьшает сумму на счете на указанное значение. Клиент не может ввести новую сумму до тех пор, пока не будет обработана предыдущая операция.

Оба потока прекращают свою работу после того, как пользователь введёт нулевую сумму.

***Решение****.* Данная задача является классическим примером задачи о производителе и потребителе. Поток, генерирующий сумму, является производителем, а поток, изменяющий состояние счета – потребителем. Для решения задачи введем два события – «сумма указана», «счет изменен». Кроме того, для блокирования одновременного доступа к критическим ресурсам – сумма операции и сумма на счете клиента, введем мьютекс.

#include <windows.h>

#include <iostream>

HANDLE hEventEnter, // событие ввода суммы

hEventChanged, // событие изменения счета

hMutex;

// переменные управления счетом

float sum = 0, total = 0;

// функция производителя

DWORD WINAPI atm(LPVOID param)

{

do

{

// дождаться, пока будет обработана предыдущая операция

WaitForSingleObject(hEventChanged, INFINITE);

// войти в критическую секцию и сгенерировать новую сумму

WaitForSingleObject(hMutex,INFINITE);

std::cout << "Введите сумму: ";

std::cin >> sum;

// выйти из критической секции

ReleaseMutex(hMutex);

// сбросить событие обработки (операция не была обработана)

ResetEvent(hEventChanged);

// сообщить о готовности новой информации

SetEvent(hEventEnter);

} while (sum != 0);

return 0;

}

// функция потребителя

DWORD WINAPI bank(LPVOID param)

{

do

{

// дождаться ввода новой информации

WaitForSingleObject(hEventEnter, INFINITE);

// войти в критическую секцию и обработать операцию

WaitForSingleObject(hMutex,INFINITE);

total -= sum;

std::cout << "-- total = "<<total<<'\n';

// освободить критическую секцию

ReleaseMutex(hMutex);

// сбросить событие ввода информации

ResetEvent(hEventEnter);

// сообщить о том, что операция учтена

SetEvent(hEventChanged);

} while (sum != 0);

return 0;

}

int \_tmain(int argc, \_TCHAR\* argv[])

{

// создание событий - оба в ручном режиме

// событие ввода создаем в состоянии "не произошло"

hEventEnter = CreateEvent(NULL,true,false,NULL);

// событие обработки создаем в состоянии "произошло"

hEventChanged = CreateEvent(NULL,true,true,NULL);

// создание мьютекса

hMutex = CreateMutex(NULL,false,NULL);

std::cout << "Введите нач.состояние счета: ";

std::cin >> total;

// запуск потоков

HANDLE hThread[2];

HANDLE hAtm = CreateThread(NULL,0,atm,NULL,0,NULL);

HANDLE hBank = CreateThread(NULL,0,bank,NULL,0,NULL);

hThread[0] = hAtm;

hThread[1] = hBank;

// ожидание завершения потоков

WaitForMultipleObjects(2,hThread,true,INFINITE);

std::cout << "Total = " << total;

// закрытие объектов

CloseHandle(hAtm);

CloseHandle(hBank);

CloseHandle(hEventEnter);

CloseHandle(hEventChanged);

CloseHandle(hMutex);

return 0;

}

## Задания

1. Написать программу трёхленточной сортировки линейного списка, распараллелив процессы разбора и слияния списков. Количество потоков и способы их синхронизации определить самостоятельно.
2. Смоделировать механизм рандеву для решения задачи нахождения значения функции ex по ее разложению в ряд Маклорена. В качестве вызывающей задачи взять функцию нахождения суммы элементов ряда, а в качестве обслуживающих задач – функцию нахождения факториала и функцию нахождения степени аргумента.

*Примечание: найдите, что такое “механизм рандеву”.*

1. Даны n файлов, содержащие неубывающие последовательности чисел. Написать программу, выполняющую слияние этих файлов в один таким образом, чтобы выборку данных из файлов и формирование результирующего файла выполняли три разных потока. Каждый из потоков выборки данных может прочитать очередное значение из своего файла только тогда, когда поток формирования результата передал соответствующее число в файл. Поток формирования результата может выполнять обработку данных только тогда, когда оба потока выборки предоставили ему данные.
2. Написать 2 программы. Первая программа - меню с вариантами действий второй программы (без вывода результата). Вторая программа содержит в себе действия:

1 - перемножить 2 больших числа (числа спросить у пользователя);

2 - удалить из массива вещественных чисел дубликаты (массив дан в файле), сохранить в другой файл;

3 - вывести самое часто встречающееся слово в тексте (файл дан) и количество вхождений;

4 - вывести минимальный элемент среди элементов, расположенных ниже главной диагонали, и максимальный элемент, среди элементов расположенных выше побочной диагонали квадратной матрицы размером nxn (размер вводится пользователем, значения генерируются автоматически);

5 - вывести все простые числа от 0 до n (вводится пользователем), используя алгоритм «Решето Эратосфена»;

*Примечание: обе программы должны обрабатывать случаи закрытия любой из них.*

*Примечание 2: дополнительный балл тем, кто сделает возможным одновременный ввод и выполнение нескольких команд.*