**АТОМЫ**

Если абстрагироваться от вопросов синхронизации, то обмен данными между потоками одного процесса не представляет никакой сложности — имея общее адресное пространство и общие открытые файлы, потоки получают беспрепятственный доступ к данным друг друга.

Другое дело — обмен данными потоков, выполняющихся в рамках разных процессов.

Для защиты процессов друг от друга ОС возводит мощные изолирующие преграды, которые не только защищают процессы, но и не позволяют им передавать друг другу данные.

Потоки разных процессов работают в разных адресных пространствах. Однако операционная система имеет доступ ко всем областям памяти, поэтому она может играть роль посредника в информационном обмене прикладных потоков.

При возникновении необходимости в обмене данными поток обращается с запросом к ОС. По этому запросу ОС, пользуясь своими привилегиями, создает различные системные средства связи, такие, например, как каналы или очереди сообщений.

Эти средства, так же как и рассмотренные выше средства синхронизации процессов, относятся к классу средств межпроцессного взаимодействия, то есть IPC (Inter Process Communications).

Тот факт, что механизмы IPC работают на уровне операционной системы, положительно сказывается на скорости и надежности программ и программных комплексов, построенных с их использованием. Эффективность приложений соответственно возрастает.

Таким образом, IPC становится необходим в том случае, если поток одного процесса должен передать данные потоку другого процесса.

Интерфейс Win32 API предоставляет приложениям возможность хранения строк в структурах, известных под названием таблиц атомов.

После того как программа сохранит строку в таблице атомов, она может обратиться к этой строке, используя атом.

Aтом (atom) представляет собой уникальное 16-разрядное значение, которое связано со строковой константой.

Использование атома в качестве ссылки на строку сродни применению дескриптора памяти для обращения к блоку общих глобальных данных.

Подобие имеющейся у приложения возможности передавать дескриптор памяти интерфейсу Win32 API с целью получения указателя на содержимое блока данных (с помощью функции GlobalLock()), приложение может также использовать средства Win32 API для получения значения строки, связанной с атомом.

Значение строки, которое представляет атом, известно под названием имени атома (atom name). Применение атомов:

1. Атомы применяются в приложениях ради экономии памяти. Сохранение неоднократно встречающихся строк в таблице атомов приложения позволяет экономить память, поскольку в данном случае значение строки сохраняется только один раз.

Прочие экземпляры строки занимают лишь по 2 байта памяти, которая требуется для атома, вместо целого ряда байтов, необходимых для хранения копии конкретной строки.

Еще одна причина хранения строк в виде атомов состоит в том, что таблицы атомов повышают общую производительность при сравнении строк, поскольку вместо сравнения каждого байта обеих строк в данном случае сравниваются лишь 16-разрядные атомы.

2. Кроме того, атомы имеют важное значение и для обмена данными между программами.

Таблица атомов, которая является глобальной для всех процессов, называется глобальной таблицей атомов (global atom table) и используется для обмена данными между процессами. Применение атомов для обмена данными между приложениями является одним из основополагающих принципов динамического обмена данными (DDE).

3. Атомы также широко используются для хранения строк переменной длины в дескрипторах объектов OLE (технологии встраивания и связывания объектов).

В результате применения атомов в дескрипторах данных OLE получаются более компактные структуры фиксированного размера с существенно более простым методом обмена.

Работа с таблицами атомов

Существует два вида таблиц атомов:

* + - 1. Локальные
      2. Глобальные.

1. У каждого процесса имеется собственная таблица атомов. Значение локального атома является уникальным для процесса, в котором оно определено.

С другой стороны, доступ к атомам, хранящимся в глобальной таблице атомов, может выполняться из любого приложения. Что касается любого конкретного глобального атома, то по заданному имени атома интерфейс Win32 API вернет каждому приложению один и тот же атом.

Функция AddAtom() сохраняет строки в локальной таблице атомов. Имена атомов сохраняются в том же регистре, в котором они вводились.

Всякий раз когда функция AddAtom() вызывается для уже существующего атома, Windows увеличивает счетчик внутренних ссылок на этот атом.

Синтаксис функции AddAtom() следующий

ATOM AddAtom (LPCTSTR lpszStringToStore )

Параметры:

lpszStringToStore – указатель на сохраняемую строку, которая завершается нулевым символом. Максимальная длина строки составляет 255 символов. Если первым символом строки оказывается знак #, а после него следуют цифры, представляющие целое число, меньшее константы MAXINATOM, тогда возвращается целый атом.

Возвращаемое значение. При успешном выполнении – вновь созданный атом. Если указанная строка уже существует в таблице атомов, возвращается тот же самый атом, что и исходная строка; в противном случае – 0.

Для уменьшения счетчика ссылок на атом применяется функция DeteteAtom(). Когда число ссылок достигает нуля, атом из таблицы атомов удаляется. Синтаксис функции DeleteAtom() следующий

ATOM DeleteAtom (ATOM nAtom)

Параметры:

nAtom – атом, который удаляется из локальной таблице атомов. Целые атомы не могут быть удалены, хотя значение NULL возвращается всякий раз, когда они используются в качестве аргументов функции DeleteAtom().

Возвращаемое значение. При успешном выполнении – NULL, в противном случае – значение nAtom.

Функция FindAtom() осуществляет поиск атома, который соответствует заданной строке, в локальной таблице атомов.

Поиск выполняется без учета регистра. Синтаксис функции FindAtom() следующий

ATOM FindAtom (LPCTSTR lpszString )

Параметры:

lpszString – указатель на искомую строку, которая завершается нулевым символом. Если первым символом строки оказывается знак #, а после него следуют цифры, представляющие целое число, меньшее константы MAXINATOM, тогда возвращается синтезированный целочисленный атом с тем же значением.

Возвращаемое значение. При успешном выполнении – 16-разрядное значение локального атома, связанного с указанной строкой; в противном случае – NULL.

Для возврата имени атома из таблицы атомов применяется функция GetAtomName(). При этом имя атома возвращается в том же регистре, в котором оно первоначально сохранялось.

2. Для управления, глобальными таблицами атомов имеется аналогичный набор функций управления атомами.

Например, функция GlobalAddAtom() добавляет атомы в глобальную таблицу атомов подобно тому, как это делает функция AddAtom() по отношению к локальной таблицы атомов.

К другим функциям управления глобальными таблицами атомов относятся GlobalDeleteAtom(), GlobalFindAtom() и GlobalGetAtomName().

И последняя функция управления атомами InitAtomTable() устанавливает количество элементов верхнего уровня в локальной таблице атомов в соответствие с заданным значением.

По умолчанию количество элементов верхнего уровня как в локальной, так и в глобальной таблице атомов равно 37.

Это отнюдь не означает, что в таблице можно разместить только 37 атомов, а скорее отражает тот факт, что вероятность конфликта, а значит, и замедления поиска в таблице, оказывается больше при наличии в таблице 37 элементов, чем, скажем, 80 элементов.

Прежде чем добавлять атомы в локальную таблицу атомов, необходимо сначала вызвать функцию InitAtomTable. Кроме того, значение, передаваемой этой функции, всегда должно быть простым числом.

Если простое число не используется, тогда вероятность конфликтов возрастет, а значит и замедлится поиск в таблице. Синтаксис функции FindAtom() следующий

BOOL InitAtomTable (DWORD nSize)

Параметры:

nSize – количество элементов верхнего уровня, устанавливаемых в локальной таблице атомов.

Возвращаемое значение. При успешном выполнении – TRUE, в противном случае – FALSE.

Соответствующая функция для установки числа элементов верхнего уровня в глобальных атомных таблицах отсутствует.

##### Применение глобальных атомов для обмена данными

Несмотря на то что глобальные атомы применяются исключительно для DDE, тем не менее, атомы можно без труда использовать и для обмена строковыми данными между отдельными экземплярами конкретного приложения.

Для этого отправитель вводит строку в глобальную таблицу атомов с помощью функции GlobalAddAtom(), a затем устанавливает параметр сообщения в соответствие со значением атома.

Получатель применяет функцию GtobalFindAtom() для получения содержимого переданных данных и удаляет соответствующий атом с помощью функции GlobalDeleteAtom().

При этом очень важно удалить атом, иначе глобальная таблица атомов окажется переполненной ненужными элементами.

Например.

// Имя сохраняемого атома.

LPCTSTR szAtom = "An atom.";

LRESULT CALLBACK WndProc( HWND hWnd, UINT uMsg, WPARAM wParam, LPARAM lParam )

{

switch( uMsg )

{

case WM\_CREATE:

// Увеличить до 73 число элементов в верхней части таблицы атомов.

InitAtomTable( 73 );

break;

case WM\_PAINT:

{

// Показать результаты поиска атома.

static PAINTSTRUCT ps;

static char szWorkArea[33];

static char szBuffer[128];

static ATOM aAnAtom;

aAnAtom = INVALID\_ATOM;

BeginPaint( hWnd, &ps );

if (aAnAtom = FindAtom( szAtom ))

{

GetAtomName( aAnAtom, szWorkArea, 32 );

wsprintf( szBuffer, "Атом не найден.", szWorkArea );

}

else

lstrcpy( szBuffer, "Атом может быть добавлен." );

TextOut( ps.hdc, 0, 0, szBuffer, lstrlen( szBuffer ) );

EndPaint( hWnd, &ps );

}

break ;

case WM\_COMMAND :

switch( LOWORD( wParam ) )

{

case IDM\_ADD:

// Ввести атом.

AddAtom( szAtom );

InvalidateRect( hWnd, NULL, TRUE );

break;

case IDM\_DELETE:

// Найти и удалить атом.

if ( FindAtom( szAtom ) )

DeleteAtom( FindAtom( szAtom ) );

InvalidateRect( hWnd, NULL, TRUE );

break;

Целочисленные атомы

В Windows 9.x и Windows NT/2000 поддерживаются также средства сохранения строк с десятичными числами в таблицах атомов. Атомы, которые представляют собой числовые строки, называются целочисленными (целыми) атомами.

Допустимыми являются только значения из диапазона 1-49151 (1-BFFFh).

Чтобы определить, может ли целое значение быть размещено в таблице атомов, его следует проверить, сравнив с константой MAXINATOM. Еще одна макрокоманда MAKEINATOM, преобразует число в целый атом.

В листинге показано, каким образом используются целые атомы. Следует заметить, что строка, сохраняемая в таблице атомов, на самом деле будет содержать цифры десятичного представления целого числа, которому предшествует знак "решетки".

Листинг. Применение целых атомов

char szStoredString[6];

WORD wValue = 1000;

ATOM aValue = AddAtom( MAKEINTATOM( wValue ) ) ;

// Значение атома равно 100.

// Строка будет содержать число "#1000".

GetAtomName( aValue, azStoredString, 6 );

Целые атомы на самом деле вообще не добавляются и не удаляются из системы.

Напротив, они синтезируются функциями управления атомами. В приведенном примере тот же самый результат может быть сохранен в строке szStoredString, если функция DeleteAtom() вообще не вызывается, при этом вместо значения wValue подставляется значение a Value.

Таким образом, функция DeleteAtom() может быть безболезненно вызвана и по отношению к целым атомам. А поскольку они вообще не существуют, они и не могут быть удалены, хотя функция DeleteAtom() так или иначе возвращает 0, что указывает на успешное завершение.