Оглавление

1 Процессы и потоки в Windows

1.1 Основные понятия

Каждый процесс **содержит**, по крайней мере, один поток, содержащий, в свою очередь, как минимум одно волокно (облегченный поток).

Для управления определенными ресурсами процессы могут объединяться в задания. Все вместе — **задания, процессы, потоки и волокна** — образует общий набор инструментов реализации параллельного выполнения процессов и управления ресурсами как на однопроцессорных, так и на многопроцессорных машинах.

1.2 Процессы

Процессы представляют собой контейнеры ресурсов.

У каждого процесса есть виртуальное адресное пространство, размер которого определяется разрядностью процессора и системой программирования. Часть адресного пространства занимает собственно пользовательский процесс, а операционная система занимает остальную его часть.

Таким образом, операционная система присутствует в адресном пространстве каждого процесса, хотя она и защищена от изменений с помощью аппаратного блока управления памятью MMU.

У процесса есть

- **идентификатор процесса,**
- **один или несколько потоков,**
- **список дескрипторов объектов(управляемых в режиме ядра)**

- **маркер доступа, хранящий информацию защиты.**

Процессы создаются с помощью вызова Win32, который принимает на входе имя исполняемого файла, определяющего начальное содержимое адресного пространства, и создает первый поток.

Каждый процесс начинается с одного потока, но новые потоки могут создаваться динамически.

1.3 Потоки

Потоки формируют основу диспетчеризации центрального процессора, так как операционная система всегда для запуска выбирает поток, а не процесс.

Соответственно, у каждого потока есть состояние, тогда как у процессов состояний нет. Потоки могут динамически создаваться вызовом Win32, которому в адресном пространстве процесса задается адрес начала исполнения. У каждого потока есть идентификатор потока, выбираемый из того же пространства, что и идентификаторы процессов, поэтому один и тот же идентификатор никогда не будет использован одновременно для процесса и для потока.

Если поток выполняется в пользовательском режиме, то, когда он выполняет системный вызов, происходит переключение в режим ядра, и продолжается выполнение тот же потока, с теми же свойствами и ограничениями, которые были у него в режиме пользователя.

У каждого потока есть два стека, один используется в режиме ядра, а другой в режиме пользователя.

Помимо состояния, идентификатора и двух стеков, у каждого потока есть

- **контекст (в котором сохраняются его регистры, когда он не работает),**
- **область для локальных переменных,**
- **а также может быть свой собственный маркер доступа.**

Если у потока есть свой маркер доступа, то он перекрывает маркер доступа процесса, чтобы клиентские потоки могли передать свои права

доступа серверным потокам, выполняющим работу для них.

Когда поток завершает свою работу, он может прекратить свое существование.

Когда прекращает существование последний активный поток, процесс завершается.

Любой поток может получить доступ ко всем объектам его процесса. Все, что ему для этого нужно сделать - это получить дескриптор объекта и обратиться к соответствующему вызову Win32. Система не следит за тем, какой объект каким потоком создан. Как только дескриптор объекта помещен в таблицу дескрипторов процесса, любой поток процесса может его использовать.

Помимо «нормальных» потоков, работающих в процессах пользователя, в операционной системе Windows есть множество потоков **процессов-демонов**, не связанных ни с каким пользовательским процессом (они ассоциированы со специальной системой или простаивающими процессами).

Некоторые демоны выполняют административные задачи, как, например, запись «грязных» страниц на диск, тогда как другие формируют пул, и ими могут пользоваться компоненты исполняющей системы или драйверы, которым нужно выполнить какие-либо асинхронные задачи в фоновом режиме.

1.4 Волокна

Переключение потоков в операционной системе Windows занимает довольно много времени, так как для этого необходимо переключение в режим ядра, а затем возврат в режим пользователя.

Для предоставления сильно облегченного псевдопараллелизма в Windows используются **волокна**.

Волокна подобны потокам, но планируются в пространстве пользовательского процесса.

У каждого потока может быть несколько **волокон**, так же как у процесса может быть несколько потоков, с той разницей, что **когда волокно логически**

блокируется, оно помещается в очередь блокированных волокон после чего для работы **выбирается другое волокно в контексте того же потока**.

Операционная система не знает о смене волокон, так как все тот же поток продолжает работу.

Так как операционная система ничего не знает о волокнах, то с ними, в отличие от заданий, процессов и потоков, не связаны объекты исполняющей системы.

Для управления волокнами нет и настоящих системных вызовов. Однако для этого есть вызовы Win32 API. Они относятся к тем вызовам Win32 API, которые не обращаются к системным вызовам.

2 Состояния потоков

Поток, может находиться в одном из шести состояний.

- **Готовый к выполнению**. Поток, который может быть направлен на выполнение. Диспетчер отслеживает все готовые к выполнению потоки и осуществляет их планирование в соответствии с приоритетом.
- **Резервный**. Поток, который будет запущен следующим на данном процессоре. Поток находится в этом состоянии до тех пор, пока процессор не освободится. Если приоритет резервного потока достаточно высок, то он может вытеснить выполняющийся в данный момент поток. В противном случае резервный поток ждет, пока не произойдет блокировка выполняющегося потока или пока не истечет выделенный ему промежуток времени.
- **Выполняющийся**. Как только диспетчер переключит поток или процесс, резервный поток перейдет в состояние выполнения, и будет пребывать в этом состоянии до тех пор, пока не произойдет одно из следующих событий:
    - поток будет вытеснен,
    - закончится отведенный ему интервал времени,
    - поток будет блокирован или завершен

В первых двух случаях поток снова переходит в состояние готовности.

- **Ожидающий**. Поток входит в состояние ожидания, если
    - он блокирован каким-то событием (например, операцией ввода-вывода),
    - он добровольно ждет синхронизации.

После того как условия ожидания будут удовлетворены, поток переходит в состояние готовности, если все его ресурсы будут доступны.

- **Переходный**. Поток переходит в это состояние, если он готов к выполнению, но ресурсы недоступны (например, страницы стека потока могут находиться на диске). После того как необходимые ресурсы станут доступны, процесс переходит в состояние готовности.
- **Завершающийся**. Завершение потока может быть инициировано самим потоком, другим потоком или может произойти вместе с завершением родительского процесса. После завершения необходимых операций освобождения ресурсов и т.п. поток удаляется из системы.

3 Приоритеты процессов и потоков

3.1 Уровни приоритета

В Windows предусмотрено 32 уровня приоритета - от 0 до 31.

0 соответствует самому низкому приоритету, 31 - самому высокому. Этот диапазон делится на три части:

- **Приоритет 0** — соответствует приоритету потока обнуления страниц (zero page thread).
- **Пятнадцать варьируемых (динамических) уровней.**

  Приоритеты с 1 по 15 соответствуют динамическим уровням приоритетов. Большинство потоков работают именно в этом диапазоне приоритетов, и Windows может корректировать в некоторых случаях приоритеты потоков из этого диапазона.
- **Шестнадцать уровней реального времени (16-31);**

Приоритеты с 16 по 31 соответствуют приоритетам «реального времени». Этот уровень достаточно высок для того, чтобы поток, работающий с таким приоритетом, мог реально помешать нормальной работе других потоков в системе (например, помешать обрабатывать сообщения от клавиатуры и мыши). Windows самостоятельно не корректирует приоритеты этого диапазона.

4 Интерфейс Win32 API для управления процессами, потоками и волокнами

Интерфейс Win32 API содержит около 100 вызовов, работающих с процессами, потоками и волокнами. Значительное количество этих вызовов в той или иной мере имеет отношение к межпроцессному взаимодействию.

Не все они являются системными вызовами. Операционная система Windows ничего не знает о волокнах. Они полностью реализованы в пространстве пользователя. Поэтому функция **CreateFiber** выполняет свою работу целиком в пространстве пользователя, не обращаясь к системным вызовам (разве что только с просьбой выделить ей память). Многие другие вызовы Win32 также обладают подобным свойством, включая функции **EnterCriticalSection** и **LeaveCriticaiSection.**

Windows позволяет указывать, на каком именно процессоре должен выполняться тот или иной процесс. Для этой цели используется команда **Set Affinity (Задать соответствие)** диспетчера задач.

Применив эту команду, можно использовать для выполнения некоторого процесса только определенный процессор. Благодаря этому можно повысить производительность, так как сократится количество обновлений кэша процессоров, которые выполняются при переключении процессоров на выполнение одного и того же процесса.

Однако применив команду Set Affinity (Задать соответствие), можно существенно снизить производительность, так как система не сможет передавать процесс наименее занятому процессору. Таким образом, применять команду Set Affinity (Задать соответствие), следует осторожно.

Чтобы назначить процессу обработчика:

На вкладке **Процессы** диспетчера задач щелкнуть правой кнопкой на процесс, которому нужно назначить процессор, выбрать команду **Задать соответствие**, а затем выберите один или несколько обработчиков. Команда **Задать соответствие** доступна только в многопроцессорных ОС.