Прерывания. Классы прерываний. Механизм прерываний. Приоритезация маскирование прерываний. Последовательность действий при обработке прерываний. Диспетчер прерываний. Обработка ловушек в Windows. Обработка прерываний в Windows. Обработка исключений в Windows.

Прерывания – это такие ситуации в операционной системе, при которых нормальный поток выполнения кода процессором прерывается. В системе периодически происходят прерывания по таймеру, при которых происходит планирование процессов, прерывания от устройств ввода-вывода. Система прерываний переводит процессор на выполнение потока команд, отличного от того, который выполнялся до этого, с последующим возвратом к исходному коду.

При первом приближении может показаться, что механизм прерываний похож на механизм выполнения процедур. Но между этими механизмами имеется важное отличие. Переход по команде безусловного или условного перехода происходит в заранее определенных программистом точках программы в зависимости от исходных данных, обрабатываемых программой. Прерывание же происходит в произвольной точке потока команд, которую программист не может прогнозировать. Прерывание возникает либо в зависимости от внешних по отношению к процессу выполнения программы событий, либо при появлении непредвиденных аварийных ситуаций в процессе выполнения программы. Сходство прерываний с процедурами состоит в том, что в обоих случаях выполняется некоторая подпрограмма, обрабатывающая специальную ситуацию, а затем продолжается выполнение основной ветви программы.

В зависимости от источника прерывания делятся на 3 класса:

- внешние (аппаратные);
- внутренние (исключения);
- программные.

Внешние прерывания могут возникать в результате действий пользователя или оператора за терминалом, или в результате поступления сигналов от аппаратных устройств – сигналов завершения операций ввода-вывода, вырабатываемых контроллерами внешних устройств компьютера, такими как принтер или винчестер, или же сигналов от датчиков управляемых компьютером технических объектов. Внешние прерывания называют также *аппаратными*, отражая тот факт, что прерывание возникает вследствие подачи некоторой аппаратурой сигнала, который передается на специальный вход прерывания процессора.

Отличительной чертой данного класса прерываний является то, что аппаратное прерывание является *асинхронным* событием. Оно может произойти в любой момент независимо от текущих команд, выполняемых

процессором. Аппаратура процессора работает так, что асинхронные прерывания возникают между выполнением двух соседних инструкций. При этом система после обработки прерывания продолжает выполнение процесса, начиная со следующей инструкции.

Важной особенностью процедур, выполняемых по запросам прерываний, является то, что они выполняют работу, чаще всего никак не связанную с текущим процессом. Например, драйвер диска может получить управление после того, как контроллер диска записал в соответствующие сектора информацию, полученную от процесса А. В типичном случае в этот момент процесс А будет находиться в состоянии ожидания завершения операции ввода-вывода и драйвер прервет какой-либо другой процесс В.

Внутренние прерывания, называемые также *исключениями* (exeption), происходят *синхронно* выполнению программы при появлении аварийной ситуации в ходе исполнения некоторой инструкции программы. Это может быть деление на нуль, ошибки защиты памяти, попытка выполнить привилегированную команду в пользовательском режиме и т.п. Повторный запуск программы в аналогичных условиях с теми же данными позволит воспроизвести исключение.

Программное прерывание возникает при выполнении особой команды процессора, выполнение которой имитирует прерывание, то есть переход на новую последовательность инструкций. Одной из причин появления инструкций прерываний в системе команд процессора является то, что их использование часто приводит к более компактному коду программ по сравнению с использованием стандартных команд выполнения процедур. Это объясняется тем, что разработчики процессора обычно резервируют для обработки прерываний небольшое количество возможных подпрограмм, так что длина операнда в команде программного прерывания меньше, чем в команде перехода на подпрограмму. Например, программное прерывание в процессорах x86 осуществляется с использованием инструкции INT, которая имеет длину операнда 1 байт (и, следовательно, предусматривает возможность применения 256 программ обработки прерывания). Другой причиной применения программных прерываний является возможность смены пользовательского режима на привилегированный одновременно с вызовом процедуры.

В результате программные прерывания часто используются для выполнения ограниченного количества функций ядра операционной системы – *системных вызовов*.

Прерываниям приписывается приоритет, с помощью которого они ранжируются по степени важности и срочности.

Прерывания обычно обрабатываются модулями операционной системы, так как действия, выполняемые по прерыванию, относятся к управлению ресурсами операционной системы – принтером, дисками, памятью, процессором и.п. Процедуры, вызываемые по прерываниям, обычно называют *обработчиками прерываний*, или *процедурами обслуживания прерываний* (Interrupt Service Routine, ISR). Аппаратные прерывания обрабатываются драйверами соответствующих внешних устройств, исключения – специальными модулями ядра, программные прерывания – процедурами ОС, обслуживающими системные вызовы.

Механизм прерываний поддерживается аппаратными средствами компьютера и программными средствами операционной системы. Аппаратная поддержка прерываний зависит от типа процессора и других аппаратных компонентов, передающих сигнал запроса прерывания от внешнего устройства к процессору (таких как контроллер внешнего устройства, шины подключения внешних устройств, контроллер прерываний).

Существуют два основных способа, с помощью которых шины выполняют прерывания: *векторный (vectored)* и *опрашиваемый (polled)*. В обоих способах процессору предоставляется информация об уровне приоритета прерывания на шине подключения внешних устройств. В случае векторных прерываний в процессор передается также информация о начальном адресе программы – обработчика прерывания.

Устройствам, использующим векторные прерывания, назначается вектор прерываний. Вектор прерываний, передаваемый в процессор, представляет собой целое число в диапазоне от 0 до 255, указывающий на одну из 256 программ обработки прерываний, адреса которых находятся в таблице обработчиков прерываний. Этот вектор может быть фиксированным, конфигурируемым или программируемым. При получении сигнала запроса прерывания процессор выполняет специальный цикл подтверждения прерывания, в котором устройство должно идентифицировать себя. В течение этого цикла устройство отвечает, выставляя на шину вектор прерываний. Затем процессор использует этот вектор для нахождения обработчика прерывания.

При использовании опрашиваемых прерываний процессор получает от запросившего прерывание устройства только информацию об уровне приоритета прерывания. С каждым уровнем прерываний может быть связано несколько устройств и соответственно несколько обработчиков прерываний. Процессор должен определить, какое устройство запросило прерывание. Это достигается вызовом всех обработчиков прерывания для данного уровня приоритета, пока один из обработчиков не подтвердит, что прерывание пришло от обслуживаемого им устройства.

Механизм прерываний чаще всего поддерживает приоритезацию и маскирование прерываний. *Приоритезация* означает, что все источники прерываний делятся на классы и каждому классу назначается свой уровень приоритета запроса на прерывание. Приоритеты могут обслуживаться как относительные и абсолютные. Обслуживание запросов прерываний по схеме с*относительными приоритетами*заключается в том, что при одновременном поступлении запросов прерываний из разных классов выбирается запрос, имеющий высший приоритет. Однако в дальнейшем при обслуживании этого запроса процедура обработки прерывания уже не откладывается даже в том случае, когда появляются более приоритетные запросы – решение о выборе нового запроса принимается только в момент завершения обслуживания очередного прерывания.

В схеме с *абсолютными приоритетами*более приоритетным прерываниям разрешается приостанавливать работу процедур обслуживания менее приоритетных прерываний. При поступлении запроса определенного класса его приоритет сравнивается с текущим, и если приоритет запроса выше, текущая процедура обработки прерываний вытесняется, а по завершении обслуживания нового прерывания происходит возврат к прерванной процедуре.

Упорядоченное обслуживание запросов прерываний наряду со схемами приоритетной обработки запросов может выполняться механизмом маскирования запросов. *Маскирование*прерывания означает, что оно не обслуживается. Схема маскирования предполагает возможность временного маскирования прерываний любого класса независимо от уровня приоритета.

Обобщенно последовательность действий по обработке прерываний можно описать таким образом:

1. При возникновении прерывания (или исключения) происходит первичное аппаратное распознавание типа прерывания. Если прерывания данного типа в этот момент запрещены, то процессор продолжает поддерживать естественный ход выполнения команд. В противном случае в зависимости от поступившей в процессор информации (уровень прерывания, вектор прерывания или тип условия внутреннего прерывания) происходит автоматический вызов процедуры обработки прерывания, адрес которой находится в специальной таблице операционной системы.
2. Автоматически сохраняется некоторая часть контекста прерванного потока (значение счетчика команд, слова состояния машины, некоторых регистров общего назначения и др.), которая позволит ядру возобновить исполнение потока после обработки прерывания.
3. Одновременно с загрузкой адреса процедуры обработки прерываний в счетчик команд может автоматически выполняться загрузка нового

значения слова состояния машины, которое определяет режим работы процессора при обработке прерывания, в том числе работу в привилегированном режиме.

4. Временно запрещаются прерывания данного типа, чтобы не образовалась очередь вложенных друг в друга потоков одной и той же процедуры. Это может производиться с использованием механизма маскирования прерываний.
5. После обработки прерывания ядром операционной системы прерванный контекст восстанавливается и работа потока возобновляется.

Для упорядочения работы обработчиков прерываний в операционных системах применяется тот же механизм, что и для упорядочения работы пользовательских процессов – механизм приоритетных очередей. Все источники прерываний делятся на несколько классов, причем каждому классу присваивается приоритет. В операционной системе выделяется программный модуль, который занимается диспетчеризацией обработчиков прерываний – диспетчер прерываний.

При возникновении прерывания диспетчер вызывается первым. Он запрещает ненадолго все прерывания, а затем выясняет причину прерывания. После этого диспетчер сравнивает назначенный данному источнику прерывания приоритет и сравнивает его с текущим приоритетом потока команд, выполняемого процессором. В этот момент процессор уже может выполнять инструкции другого обработчика прерываний, имеющего некоторый приоритет. Если приоритет нового запроса выше текущего, то выполнение текущего обработчика приостанавливается и он помещается в соответствующую очередь обработчиков прерываний. В противном случае в очередь помещается обработчик нового запроса.

Диспетчер прерываний в Windows называется *обработчиком ловушек* (trap handler). Рисунок 6.1 иллюстрирует некоторые ситуации, в которых активизируются обработчики ловушек.

Рис. 6.1. Диспетчеризация ловушек.

Ядро Windows различает прерывания и исключения. Прерывание является асинхронным событием. Исключение представляет собой синхронное событие. Прерывания и исключения можно генерировать как программно, так и аппаратно. При аппаратном прерывании или исключении процессор записывает статусную информацию в стек ядра для прерванного потока. Если поток выполнялся в пользовательском режиме, Windows переключается на стек режима ядра для потока. Затем создает в стеке ядра прерванного потока *фрейм ловушки* (trap frame), в котором сохраняет информацию о состоянии потока. Программное прерывание ядро обслуживает либо при обработке аппаратного прерывания, либо синхронно – при вызове потоком функции ядра, относящейся к данному программному прерыванию.

В большинстве случаев ядро устанавливает функции, выполняющие общую обработку ловушек до и после передачи управления другим функциям, которые ставят ловушки. Например, когда устройство генерирует прерывание, обработчик ловушек аппаратных прерываний (принадлежащий ядру) передает управление *процедуре обслуживания прерывания* (interrupt service routine,ISR), предоставленной драйвером соответствующего устройства. Если прерывание возникло в результате вызова системного сервиса, обработчик ловушек общесистемных сервисов передает управление функции указанного системного сервиса в исполнительной системе. Ядро также устанавливает обработчики для ловушек, которые оно не ожидает или не обрабатывает.

На аппаратных платформах, поддерживаемых Windows, прерывания, связанные с внешним вводом-выводом, поступают по одной из линий контроллера прерываний, который, в свою очередь связан с процессором единственной линией, по которой контроллер и уведомляет о прерывании. Как только процессор прерывается, он требует от контроллера *запрос*

прерывания(interrupt request, IRQ). Контроллер транслирует IRQ в номер прерывания, используемый как индекс в структуре, называемой*таблицей диспетчеризации прерываний*(interrupt dispatch table, IDT), и передает управление соответствующей процедуре. При загрузке Windows заносит в IDT указатели на процедуры ядра, обрабатывающие каждое прерывание и исключение (первая часть IDT используется для исключений, а аппаратные прерывания располагаются за ней).

Windowsувязывает аппаратные IRQ с номерами прерываний в IDT. Эта таблица используется системой и при конфигурировании обработчиков ловушек для исключений. Число IRQ на конкретной машине определяется архитектурой используемого в ней контроллера прерываний.

Хотя контроллеры прерываний различают уровни приоритетов прерываний, Windows использует свою схему приоритетов прерываний, известную под названием уровни запросов прерываний (interrupt request levels, IRQL). Ядро определяет стандартный набор IRQL для программных прерываний, а HAL увязывет IRQL с номерами аппаратных прерываний. IRQL, определенные для архитектуры x86, показаны на рис. 6.2.

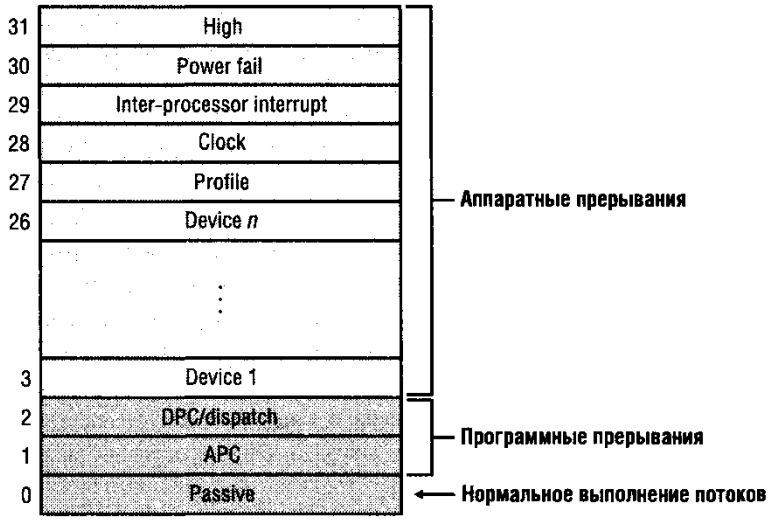

Рис. 6.2. Уровни запросов прерываний в x86-системах

Прерывания обслуживаются в порядке их приоритета, и прерывания с более высоким приоритетом вытесняют обработку прерываний с более низким приоритетом. При возникновении прерывания с высоким приоритетом процессор сохраняет информацию о состоянии прерванного потока и активизирует сопоставленный с данным прерыванием диспетчер ловушки. Последний повышает IRQL и вызывает процедуру обслуживания прерывания ISR. После выполнения ISR диспетчер прерывания понижает IRQL процессора до исходного уровня и загружает сохраненные ранее данные о состоянии машины. Прерванный поток возобновляется с той точки, где он

был прерван. Когда ядро понижает IRQL, могут «материализоваться» ранее замаскированные прерывания с более низким приоритетом. Тогда вышеописанный процесс повторяется ядром для обработки и этих прерываний. Типичная схема обслуживания прерываний показана на рис. 6.3.

В отличие от прерываний, которые могут возникнуть в любой момент, исключения являются прямым следствием действий выполняемой программы. Windowsвводит понятие структурной обработки исключений (structuredexceptionhandling,SHE), позволяющей приложениям получить управление при возникновении исключений. При этом приложение может исправить ситуацию, которая привела к исключению, или уведомить систему о том, что данное исключение ему неизвестно, и тогда система продолжит поиск подходящего обработчика для данного исключения.

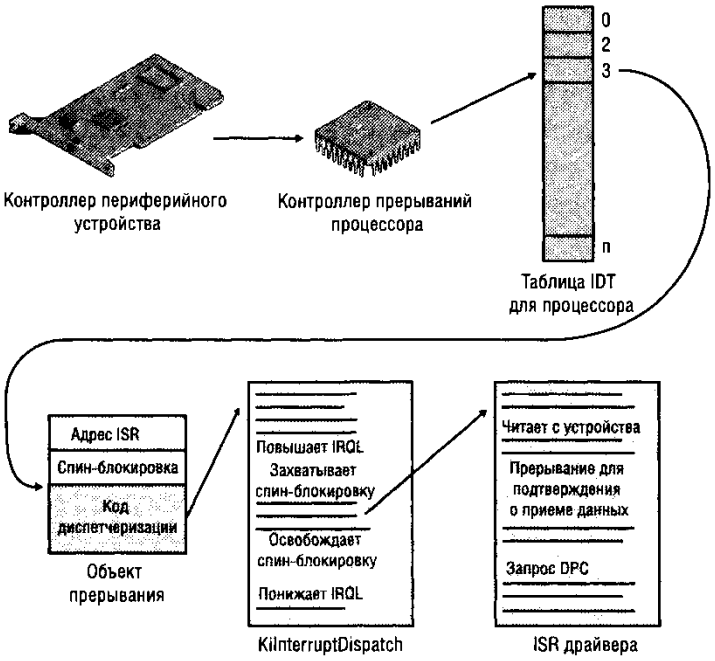

Рис. 6.3. Типичная схема обслуживания прерываний

В системах типа x86 все исключения имеют предопределенные номера прерываний, прямо соответствующие записям в IDT, ссылающимся на обработчики ловушек конкретных исключений. В таблице 6.1 перечислены исключения, определенные для систем типа x86, с указанием номеров прерываний.

Номер прерывания	Исключение
0	Divide Error (ошибка деления)
1	DEBUG TRAP (ловушка отладки)

2	NMI/NPXError(ошибкаNMI/NPX)
3	Breakpoint (точка прерывания)
4	Overflow (переполнение)
5	BOUND/Print Screen
6	InvalidOpcode(неправильный код операции)
7	NPX Not Available (NPX недоступен)
8	Double Exception (двойное исключение)
9	NPXSegmentOverrun(выход за пределы сегментаNPX)
A	Invalid Task State Segment (неправильный TSS)
B	Segment Not Present (сегмент отсутствует)
C	Stack Fault (ошибка стека)
D	General Protection (ошибка общей защиты)
E	Page Fault (ошибка страницы)
F	Зарезервировано Intel
10	FloatingPoint(ошибка в операции с плавающей точкой)
11	AlignmentCheck(ошибка контроля выравнивания)

Таблица 6.1. Исключения в системах типа x86

Все исключения, кроме достаточно простых, которые могут быть разрешены обработчиком ловушек, обслуживаются модулем ядра – диспетчером исключений (exception dispatcher). Его задача заключается в поиске обработчика, способного «справиться» с данным исключением.

Ядро перехватывает и обрабатывает некоторые из этих исключений прозрачно для пользовательских программ. Так, если при выполнении отлаживаемой программы встретилась точка прерывания, генерируется исключение, обрабатываемое ядром за счет вызова отладчика. Ряд исключений ядро обрабатывает, просто возвращая код неудачной операции.

Определенные исключения могут передаваться в неизменном виде пользовательским процессам. Например, при ошибке доступа к памяти или

при переполнении в ходе арифметической операции генерируется исключение, не обрабатываемое операционной системой. Для обработки этих исключений подсистема окружения может устанавливать *обработчики исключений на основе SEH-фремйма*. Этим термином обозначается обработчик исключения, сопоставленный с вызовом конкретной процедуры. При активизации такой процедуры в стек заталкивается стековый фрейм, представляющий вызов процедуры. Со стековым фреймом можно сопоставить один или несколько обработчиков исключений, каждый из которых защищает определенный блок кода исходной программы. При возникновении исключения ядро ищет обработчик, сопоставленный с предыдущим стековым фреймом, - и так до тех пор, пока не будет найден подходящий обработчик. Если обработчик не удалось найти, ядро вызывает собственные обработчики по умолчанию.

Когда происходит исключение, цепочка событий начинается в ядре. Процессор передает управление обработчику ловушек в ядре, который создает фрейм ловушки, а также запись исключения, содержащую сведения о ее причине и другую сопутствующую информацию.

Если исключение возникло в режиме ядра, то для его обработки диспетчер исключений просто вызывает процедуру поиска подходящего обработчика SEH-фрейма. Поскольку необработанные исключения режима ядра были бы фатальными ошибками операционной системы, диспетчер всегда находит какой-нибудь отладчик.

Если исключение возникло в пользовательском режиме, диспетчер предпринимает более сложные действия. Подсистема windowsпредусматривает порт отладчика (debuggerport) и порт исключений (exceptionport) для приема уведомлений об исключениях пользовательского режима вWindows-процессах. Они применяются ядром при обработке исключений по умолчанию, как показано на рис. 6.4.

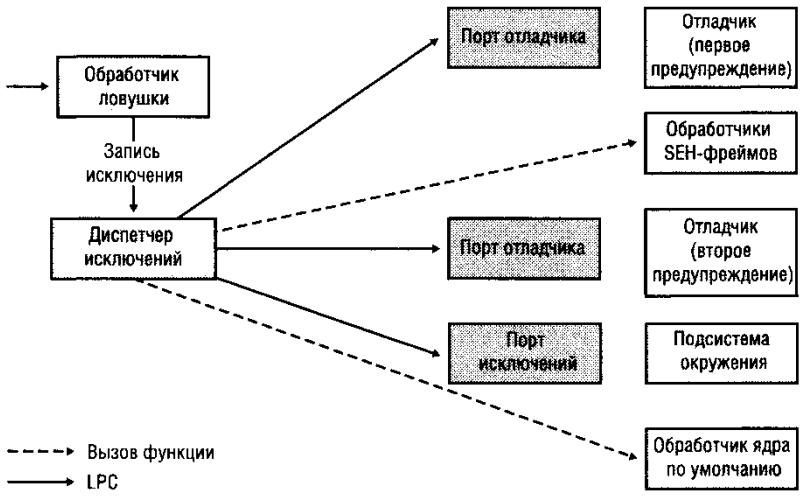

Рис. 6.4. Диспетчеризация исключений

Первым делом диспетчер исключений проверяет, подключен ли к процессу, вызвавшему исключение, отладчик. Если подключен, диспетчер исключений посылает отладчику первое предупреждение (в Windows XP и более поздних системах диспетчер посылает сообщение объекта отладчика объекту отладки, сопопоставленному с процессом).

Если к процессу не подключен отладчик или отладчик не в состоянии обработать данное исключение, диспетчер исключений переключается в пользовательский режим, копирует фрей ловушки в пользовательский стек и вызывает процедуру поиска обработчика SEH-фрейма. Если поиск не дал результатов, диспетчер возвращается в режим ядра и снова вызывает отладчик. При этом посылается второе предупреждение.

Если отладчик не запущен и обработчики SEH-фреймов не найдены, ядро посылает сообщение в порт исключений, сопоставленный с процессом потока. Порт исключений дает возможность подсистеме окружения транслировать исходное исключение в уведомление или исключение, специфичное для ее окружения. Далее выводится окно сообщения, уведомляющее пользователя о сбое, и процесс завершается.