2.6 Операционные системы семейства MS Windows

2.6.1 Этапы развития:

1) интерфейсные системы: Windows3.0 , 3.1 , 3.11
2) ОС Windows - 95, -98, -2000, -ME, -XP.

Характерные черты:

Многозадачные, многопотоковые ОС, которые обеспечивают управление сразу несколькими программами и каждая их этих программ может иметь несколько параллельных потоков (то есть независимо исполняемых частей).

Особенности организации работы:

Главным окном является рабочий стол, который содержит пиктограммы папок.

Папка – хранилище для файлов, каталогов, магнитных дисков, очереди заданий на печать и другие папки. Для отображения содержимого папок используется 2 типа окон:

- окно папки;
- окно "проводника"

"Проводник" – приложение, которое обеспечивает отображение файловой структуры и выполнение команд.

"Рабочий стол" – вершина иерархии всех папок. Имеет объектно-ориентированные свойства(если кликнуть правой кнопкой мыши на любую пиктограмму, то появится контекстное меню с указанием возможных действий.

WINDOWS NT - мультипрограммная ОС обеспечивает управление сразу несколькими программами, поддерживает работу как 32- разрядных приложений, для которых используется приоритетная многозадачность поддерживаемая средствами Windows 32, а также позволяет запускать 16 разрядные DOS- приложения.

Windows NT поддерживает четырехуровневые кольца защиты оперативной памяти:

- в 3-ем кольце размещаются прикладные программы и динамические компонуемые библиотеки, содержащие сервисные функции ОС;
- в 0-ом кольце – ядро ОС;
- в 1-ом кольце системные программы и драйверы;
- во 2-ом сервисные программы и драйверы.
- При такой организации критически важны компоненты ОС изолируется от прикладных программ, что обеспечивает большую стабильность ОС.

В кольце 3 размещены виртуальные машины, на которых исполняются прикладные программы.

Все программы Windows 16 разрядные работают каждая на своей виртуальной машине. 16-разрядные программы работают на виртуальной DOS Машине (VDM), причем все загруженные программы могут передавать данные через буфер обмена друг другу, а также приложениям Windows .

2.6.2 Как Windows выполняет программный код

Операционная система Windows для поддержки своей эффективности и целостности использует два режима: пользователя и ядра. Архитектура процессора Intel 80386 и следующих моделей определяет четыре уровня привилегий, называемых кольцами, для защиты кода и данных системы от случайного или преднамеренного изменения со стороны менее привилегированного кода. Такой метод выполнения кода называется моделью защиты Intel.

Уровень привилегий 0, известный *как режим ядра,* максимальный. Уровень привилегий 3, или *режим пользователя,* — минимальный. Когда код выполняется на некотором уровне привилегий, говорят, что он выполняется в соответствующем кольце. Операционные системы семейства Windows используют только кольца 0 и 3 (рис. 2.1).

Рис. 2.1 Кольца 0 и 3 в модели защиты Intel

Режим ядра Режим ядра (кольцо 0) — это наиболее привилегированный режим. Работая в нем, код имеет прямой доступ ко всей аппаратуре и всему адресному пространству.

Программное обеспечение, выполняющееся в режиме ядра:

- имеет прямой доступ к аппаратному обеспечению;
- имеет доступ ко всей памяти компьютера;
- не может быть вытеснено в страничный файл на жестком диске;
- выполняется с большим приоритетом, чем процессы режима пользователя.

В частности, в кольце 0 выполняется код ядра операционных систем Windows 95 и Windows NT. Поскольку компоненты режима ядра защищены архитектурно, процессор предотвращает их изменение другой программой. Хотя кольцо 0 предоставляет максимальную защиту, не следует запускать в кольце 0 что попало — ведь компоненты этого режима имеют доступ ко всей системе. Если программный компонент в режиме ядра потерпит крах, это может разрушить всю систему.

Поскольку одна из задач Windows 95 — максимальная обратная совместимость, многие старые 16-разрядные драйверы и приложения используют прямой доступ к аппаратуре. Windows NT не предоставляет таким приложениям требуемый уровень доступа, поэтому зачастую они не могут работать под управлением Windows NT Workstation и Windows NT Server.

Режим пользователя

Режим пользователя предоставляет меньше привилегий, нежели режим ядра, — в частности, он не обеспечивает прямой доступ к аппаратуре. Код, выполняющийся в кольце 3, ограничен выделенным ему адресным пространством, а для вызова системных сервисов использует интерфейс прикладного программирования (API) Windows.

Процессы режима пользователя характеризуются следующими особенностями.

- Не имеют прямого доступа к аппаратуре.

  Это защищает систему от неисправных приложений или неавторизованного доступа.

- Ограничены выделенным им адресным пространством.

  Таким образом операционная система обеспечивает свою целостность. Процессу выделяется определенная область адресов и запрещено выходить за эту область.

- Могут быть вытеснены из физической памяти в виртуальную память на жестком диске.

  Механизм виртуальной памяти позволяет использовать пространство на жестком диске как дополнительное ОЗУ. О виртуальной памяти подробно рассказано чуть позже в этой главе.

- Выполняются с меньшим приоритетом, чем компоненты режима ядра.

  Поскольку приоритет процессов режима пользователя ниже, они получают меньший доступ к процессору, чем процессы режима ядра. Это гарантирует, что операционная система не будет ожидать окончания работы такого процесса. Кроме того, неисправный программный компонент, выполняющийся в режиме пользователя, не вызовет крах всей системы и не повлияет на другие приложения, работающие параллельно.

2.6.3 Многозадачность

Многозадачность — способность операционной системы выполнять более одной программы (задачи) одновременно. Противоположный подход — однозадачность, когда один процесс должен быть завершен прежде, чем сможет начаться другой. MS-DOS — пример однозадачной среды, а Windows 95 и Windows NT — многозадачные среды.

Конечно же, и в многозадачной системе программы не выполняются одновременно — процессор переключается между ними. Благодаря этому Вы можете выполнить запрос к базе данных и продолжить работу с редактором текстов до тех пор, пока не появятся результаты запроса. Многозадачность, кроме того, позволяет компьютеру эффективно использовать время, которое иначе было бы потеряно в ожидании команды пользователя или ответа устройств ввода/вывода. Для понимания многозадачности необходимо сначала познакомиться с процессами и потоками.

Процессы и потоки

Приложение, разработанное для Windows, состоит из одного или более процессов (рис. 2.2). Процесс — это, попросту говоря, выполняемая программа. Ему принадлежат адресное пространство и выделенные ресурсы, а также один или более потоков, выполняющихся в его контексте.

Поток — это основная единица, которой операционная система выделяет процессорное время, и минимальный «квант» кода, который может быть запланирован для выполнения. Кроме того, это часть процесса, выполняющаяся в каждый момент времени. Поток работает в адресном пространстве процесса и использует ресурсы, выделенные процессу.

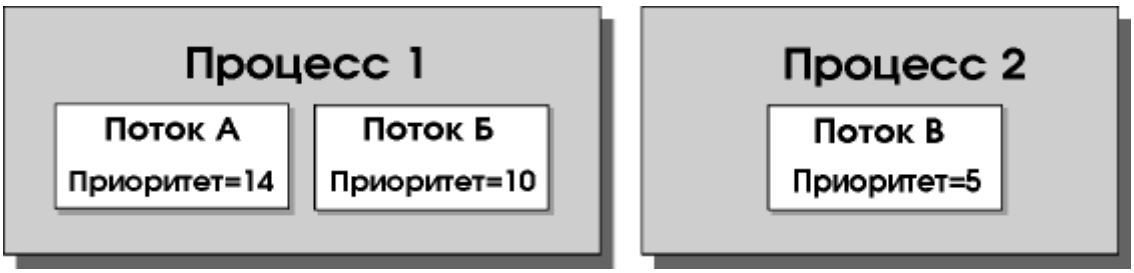

Рис. 2.2 Приоритеты потоков

Любой процесс содержит хотя бы один поток. Каждый процесс 16-разрядного Windows-приложения или программы MS-DOS имеет только один поток, тогда как процессы 32-разрядных Windows-приложений могут включать их несколько.

Примечание Ресурсами владеют процессы, а не потоки — последние только используют ресурсы, выделенные процессу. Например, если программа запросит порт, им будет управлять процесс. Любой поток процесса может обратиться к порту, но ни один из них не вправе самостоятельно запросить использование порта.

2.6.4 Вытесняющая и кооперативная многозадачность

Существуют два типа многозадачности: кооперативная (не вытесняющая) и вытесняющая (рис. 2.3). В кооперативной многозадачной среде (например, Windows 3.1) контроль над процессором никогда не отбирается у задачи — приложение должно самостоятельно отказаться от контроля над процессором, чтобы другое приложение заработало.

Вытесняющая многозадачность отличается от кооперативной тем, что операционная система может получить контроль над процессором без согласия выполняющегося приложения. Лишение приложения контроля над процессором называется *вытеснением.* Windows 95 и Windows NT используют вытесняющую многозадачность для MS-DOS и 32-разрядных Windows-приложений.

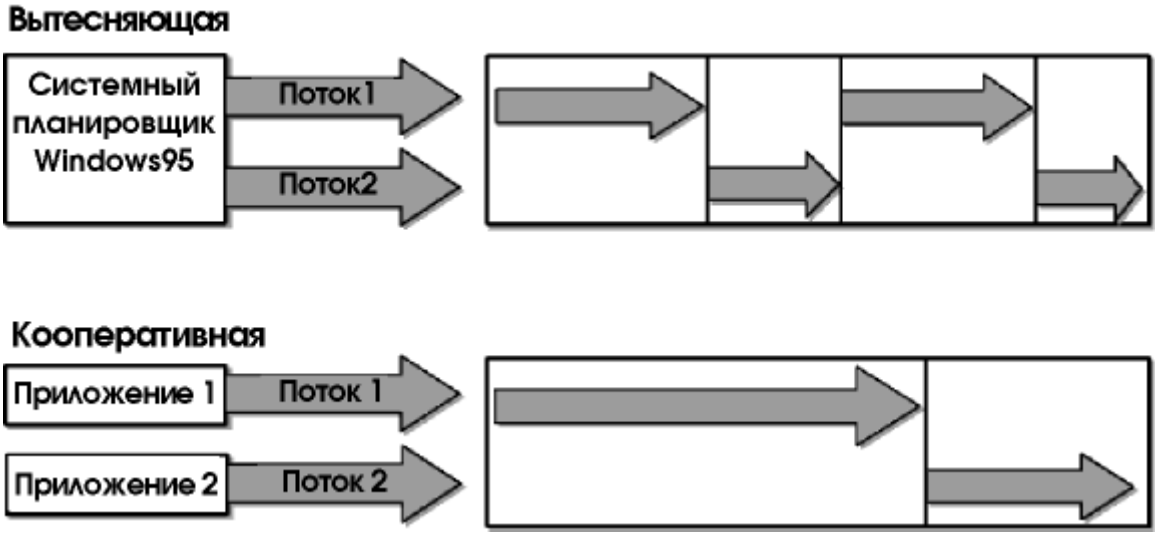

Рис.2.3 Вытесняющая и кооперативная многозадачность

Разработчики программ, выполняющихся под управлением кооперативной операционной системы, должны учитывать необходимость частого возврата управления процессором операционной системе. Программа, которая недостаточно часто отдает управление, блокирует кооперативную операционную систему.

Windows NT применяет вытесняющую многозадачность при выполнении 16-разрядных приложений Windows и MS-DOS. Windows NT обеспечивает полную защиту памяти 16-разрядных приложений, так как каждое из них выполняется в рамках собственной виртуальной машины. Windows 95, напротив, использует кооперативную многозадачность для всех 16-разрядных приложений — это необходимо для сохранения совместимости с 16-разрядными Windows-программами, которые сами контролируют свое выполнение.

Планирование

С помощью планирования операционная система определяет, какой поток использует процессор в данный момент времени. Windows реализует многозадачность, присваивая каждому потоку приоритет, что позволяет ему использовать ресурсы системы. Планирование основано на заранее заданной единице времени, называемой *квантом.* Фактическая продолжительность кванта времени зависит от конфигурации системы. Уровни приоритета находятся в диапазоне от 0 (наименьший приоритет) до 31 (наибольший приоритет). Поток с наибольшим приоритетом получает процессор в свое распоряжение (рис. 2.4).

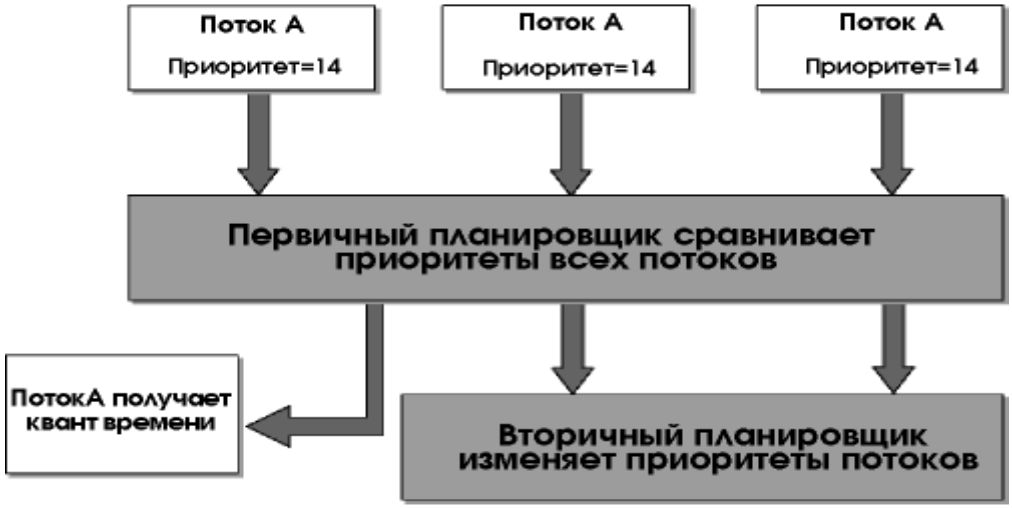

Рис. 2.4 Процесс планирования

Приоритет каждого потока определяется по:

- классу приоритета процесса, которому принадлежит поток;
- уровню приоритета потока внутри класса приоритета его процесса.

Классы приоритетов

Класс приоритета процесса и уровень приоритета потока определяют базовый приоритет потока. Уровни приоритетов Windows разделены на два класса:

- реального времени (приоритеты от 16 до 31) — используется для выполнения основных функций операционной системы и обычно не применяется для приложений;
- переменного приоритета (приоритет от 0 до 15) — определяет процессорный приоритет приложений; приоритет 0 возможен только для бесстраничного системного потока.

Уровни приоритетов

Процессам могут быть присвоены следующие базовые уровни приоритетов:

- низкий — запускает приложения с уровнем приоритета 4;
- обычный — запускает приложения с уровнем приоритета 7;
- высокий — запускает приложения с уровнем приоритета 13;
- реального времени — запускает приложения с уровнем приоритета 24.

Примечание: Не запускайте приложения с классом приоритета реального времени — это может привести к нестабильности в работе операционной системы.

2.6.5 Управление памятью

В Windows 95 и NT каждый процесс имеет собственное адресное пространство, что позволяет адресовать до 4 Гб памяти. Отметим, что Windows выделяет процессу 4 Гб адресов памяти, а не физического ОЗУ. Физическая память ограничена имеющимися системными ресурсами (ОЗУ и дисковое пространство). Windows выделяет каждому приложению 2 Гб адресов памяти, а другие 2 Гб резервируются для нужд ядра.

Большинство компьютеров не располагают 4 Гб ОЗУ, и по этой причине Windows использует механизм *виртуальной памяти.* Таким образом, Windows может перенести часть содержимого физической памяти на жесткий диск, когда объем доступного ОЗУ будет исчерпан. Этот процесс известен как *подкачка* (рис. 1.6).

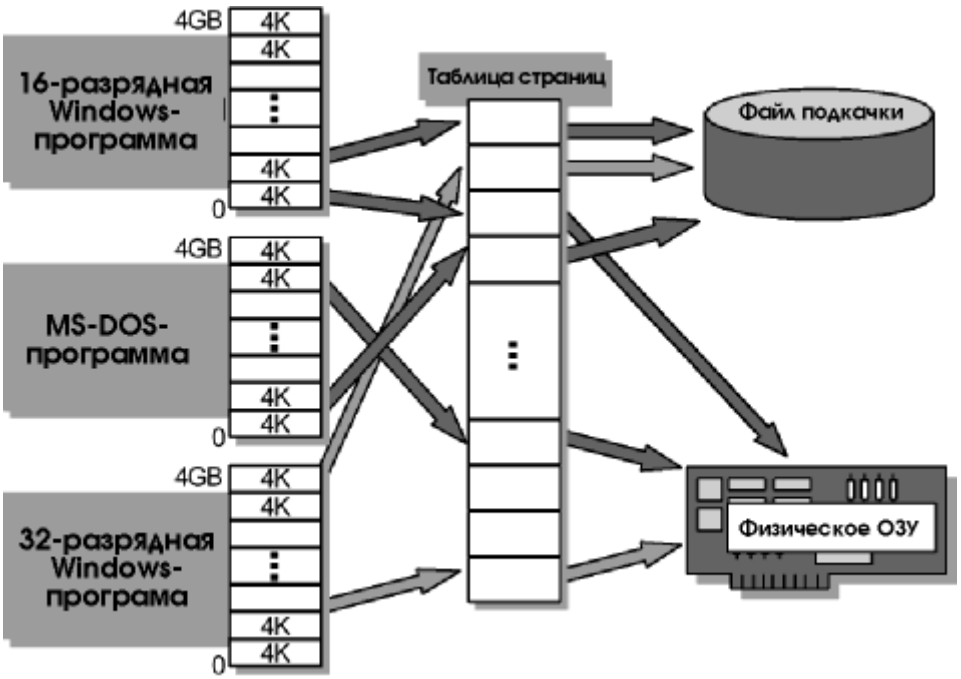

Рис. 2.5 Выделение виртуальной памяти приложениям

Виртуальные адреса, используемые процессом, не совпадают с адресами физической памяти. Для каждого процесса ядро поддерживает так называемую *таблицу страниц* — внутреннюю структуру, которая позволяет преобразовать виртуальные адреса в физические.

Виртуальная память

Процессоры Intel, начиная с модели 80386, позволяют отобразить область физической памяти на любую область 32-разрядных адресов. Виртуальная память Windows использует этот механизм для того, чтобы любая программа вела себя так, будто она имеет собственное физическое ОЗУ.

Windows для доступа к памяти применяет 32-разрядную линейную адресацию: приложения обращаются к памяти с помощью 32-разрядных адресов. Каждая программа имеет собственное виртуальное адресное пространство, которое диспетчер виртуальной памяти преобразует в адреса физического ОЗУ или в файле на жестком диске.

Постраничная подкачка Физическое и виртуальное (логическое) адресное пространство каждого процесса разделено на страницы — «кванты» памяти, размер которых зависит от компьютера. Например, для компьютеров x86 размер страницы составляет 4 кб. Ядро может перемещать страницы памяти в страничный файл на диске (Pagefile.sys) и обратно: таким образом, управление памятью становится более гибким. Когда страница перемещается в физическую память, ядро обновляет таблицы страниц соответствующих процессов. Если ядру требуется место в физической памяти, оно вытесняет самые старые страницы физической памяти в страничный файл. Манипуляции ядра с физической памятью совершенно незаметны (прозрачны) для приложений, которые работают только со своими виртуальными адресными пространствами.

2.6.6 Выполнение приложений

Windows 95 и Windows NT по-разному выполняют приложения, особенно 16-разрядные.

Механизм сообщений Windows В отличие от MS-DOS, Windows для управления приложениями использует модель сообщений. Сообщение генерируется всякий раз, когда происходит какое-то событие, например пользователь нажимает и отпускает клавишу на клавиатуре или передвигает мышь. Сообщение помещается в так называемую *очередь сообщений.* Активное приложение постоянно проверяет свою очередь и извлекает из нее поступившие сообщения.

Рис.2.6 Очереди сообщении

Обмен сообщениями в 16-разрядных версиях Windows

В Windows 3.1 очередь сообщений операционной системы — единая. Она обслуживает все 16-разрядные Windows-приложения, которые проверяют ее и извлекают адресованные им сообщения. Такое решение не лишено недостатков — например, если у какого-то приложения возникнут проблемы, оно может не позволить другим приложениям проверить очередь сообщений. В этом случае последние прекратят работу, пока не получат управление и не смогут проверить наличие адресованных им сообщений.

Обмен сообщениями в Windows 95

В Windows 95 проблемы единой очереди сообщений разрешены: у каждого выполняющегося Win32-приложения — своя очередь. Каждый поток в Win32-приложении имеет собственную очередь сообщений и, значит, никак не влияет на поведение других работающих приложений. Если Winl6- или Win32-приложение потерпит крах, остальные Win32-приложения будут действовать на основе вытесняющей многозадачности и смогут принимать поступающие сообщения из своих очередей.

Тем не менее в целях совместимости все 16-разрядные Windows-приложения под управлением Windows 95 используют общую очередь сообщений. Очевидно, если с одним из них что-то произойдет, остальным будет перекрыт доступ к очереди до тех пор, пока программа, вызвавшая проблему, не будет завершена.

Виртуальные машины

Windows NT выполняет приложения в рамках виртуальных машин (Virtual Machine, VM). Фактически VM — это создаваемая операционной системой среда для выполнения

приложения, которая полностью эмулирует все ресурсы компьютера. С точки зрения приложения, виртуальная машина — это полноценный компьютер, предоставляющий ему все имеющиеся ресурсы.

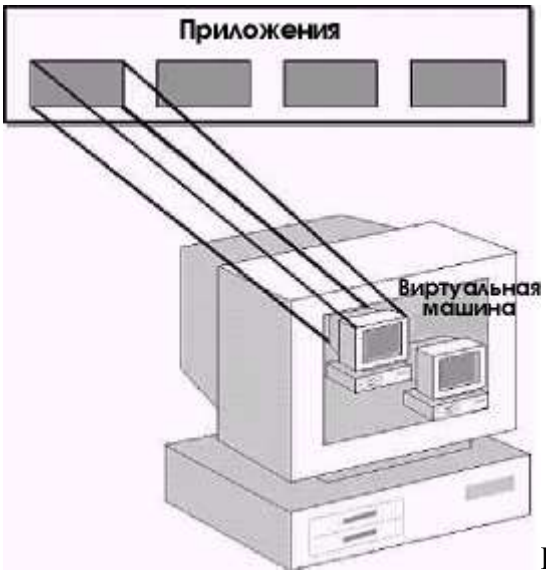

Рис. 2.7 Виртуальные машины

Каждое 16-разрядное Windows- и MS-DOS-приложение под управлением Windows NT выполняется в отдельном адресном пространстве, называемом виртуальной DOS-машиной (Virtual DOS Machine, VDM). При этом обеспечивается защита программы, а Windows NT может реализовать вытесняющую многозадачность для всех сервисов операционной системы и приложений.

В Windows 95 предусмотрено выполнение MS-DOS-приложений в отдельных VDM, однако, поскольку часть памяти доступна всем виртуальным машинам, MS-DOS-приложения представляют собой потенциальную угрозу стабильности системы.

2.6.7 Драйверы устройств в Windows

Драйвер устройства — это программный компонент, получающий команды от операционной системы и преобразующий их в команды конкретного устройства (рис. 2.8). Зачастую драйверы разрабатываются производителями аппаратного обеспечения, и компания Microsoft их напрямую не поддерживает.

Рис. 2.8 Архитектура драйверов устройств

Драйверы устройств позволяют разработчикам создавать аппаратно-независимые приложения. Другими словами, разработчику на стадии создания приложения не нужно заботиться о том, какую именно аппаратуру будет применять пользователь. Пользователь, в свою очередь, может корректировать конфигурацию компьютера, не затрагивая работоспособность приложений. Windows использует драйверы для таких компонентов, как:

- дисплеи;
- звуковые карты;
- устройства связи;
- принтеры;
- сетевые адаптеры.

В зависимости от того, для какой операционной системы семейства Windows разработан драйвер, он может принадлежать к одной из двух групп: защищенного режима и реального режима.

Драйверы защищенного и реального режима

Драйверы реального режима созданы для работы в реальном режиме операционной системы MS-DOS. Они не так безопасны и устойчивы, как драйверы защищенного режима, которые используют преимущества архитектуры защищенного режима процессоров 80386 и последующих моделей. Драйвер защищенного режима или драйвер виртуального устройства (Virtual Device Driver, VxD) обеспечивает быстрый разделяемый доступ к устройству. Кроме того, операционные системы семейства Windows выполняют 32-разрядный код защищенного режима более эффективно, чем 16-разрядный код реального режима.

Windows 95 поддерживает оба типа драйверов, а Windows NT — только драйверы защищенного режима. Компания Microsoft настоятельно рекомендует применять 32-разрядные драйверы защищенного режима везде, где возможно.

2.6.8 Интерфейс прикладного программирования Win32

Интерфейс прикладного программирования (Application Programming Interface, API) Win32 обеспечивает приложениям доступ ко всему спектру функций операционных систем семейства Windows. Функции, сообщения и структуры Win32 формируют последовательный и единообразный **API** для Windows 95 и Windows NT. Средства API Win32 позволяют разрабатывать приложения, успешно работающие на всех платформах и в то же время при надобности использующие уникальные особенности любой из них.

Многие функции API интегрированы в состав таких программ, как Visual Basic. Например, функцию API Win32 **MessageBox** можно вызвать непосредственно или через функцию Visual Basic **MsgBox.** Средствами Visual Basic обычно пользоваться легче, однако во многих случаях разработчики найдут непосредственное применение и самим функциям API Win32.

Основной код API Win32

Базовый код API Win32 содержится в трех библиотеках динамической загрузки (Dynamic Link Library, DLL): USER32, GDI32 и KERNEL32.

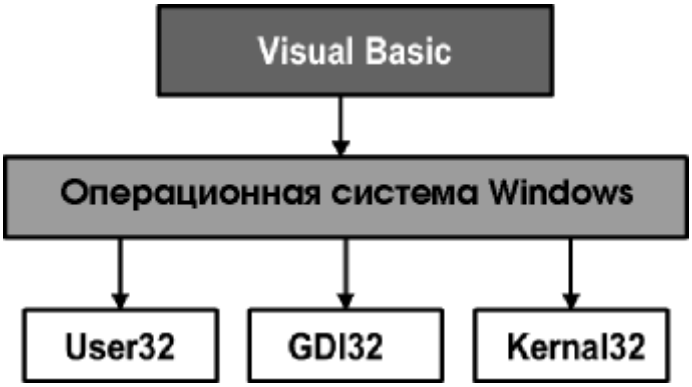

Рис.2.9 Библиотеки API Win32

USER32

User32.dll и User.exe создают и контролируют окна на экране, выполняя все запросы по созданию, перемещению, изменению размеров и уничтожению окон. **User.exe,** кроме того, обрабатывает запросы, относящиеся к значкам и другим элементам интерфейса пользователя, а также переадресует события, порожденные различными устройствами ввода, соответствующим приложениям.

GD132

Gdi32.dll и Gdi.exe контролируют интерфейс графических устройств (Graphics Device Interface, GDI). GDI выполняет графические операции при создании изображения на системном дисплее и других устройствах, включая:

- вывод на экран;
- вывод на принтер;
- включение/отключение пикселов.

KERNEL32 Kernel32.dll выполняет базовые функции операционной системы, в том числе:

- управление памятью;
- файловый ввод/вывод;
- загрузку программы;
- выполнение программы.

Примечание При объявлении функций API в Visual Basic 32-разрядные функции чувствительны к регистру символов, а эквивалентные 16-разрядные функции — нет. Это необходимо иметь в виду при преобразовании 16-разрядных приложений в 32-разрядные.

***32*- и 16-разрядные компоненты**

В Windows 95 включены 16-разрядные версии User, GDI и Kernel. Комбинация 16-разрядного и 32-разрядного кода позволяет сохранить совместимость с существующими

приложениями и драйверами и одновременно увеличить производительность системы по сравнению с Windows 3.1. Windows 95 использует 32-разрядный код везде, где это увеличивает производительность не в ущерб совместимости. Для включения в Windows 95 16-разрядных компонентов есть три основные причины:

- код для 16-разрядных систем обеспечивает обратную совместимость с приложениями и драйверами, разработанными для Windows 3.1;
- в некоторых случаях 16-разрядный код выполняется быстрее, чем аналогичный 32-разрядный;
- 32-разрядный код требует больше памяти, чем эквивалентный 16-разрядный.

Одна из основных задач Windows 95 — эффективная работа на компьютерах с ограниченным объемом ОЗУ, и применение 16-разрядного кода способствует решению этой задачи.

Подсистемы ввода/вывода и драйверы устройств, включая сетевые и файловые системы, являются полностью 32-разрядными, как и все компоненты управления памятью и планирования. Часто возникающая при этом проблема вызова 32-разрядной функции из 16-разрядного приложения (или наоборот) решается при помощи *шлюзования.*

Шлюзование

Эта операция выполняется, когда операционная система преобразует вызов 16-разрядной функции в вызов 32-разрядной. Процессы Windows 95 и Windows NT не могут содержать одновременно и 16-разрядный, и 32-разрядный код. Шлюз позволяет коду с одной стороны границы вызывать код с другой ее стороны. Каждая платформа использует один или несколько механизмов шлюзования:

- механизм *базовых шлюзов* позволяет 16-разрядному Windows-приложению в системе под управлением Windows 95 и Windows NT загрузить и вызвать 32-разрядную библиотеку;
- с помощью механизма *плоских шлюзов,* реализованного только в Windows NT, Win32-приложение загружает и вызывает 16-разрядную библиотеку и наоборот.

2.6.9 Реестр Windows

Реестр — это унифицированная база данных, содержащая информацию об аппаратной и программной конфигурации локального компьютера. Здесь же хранятся данные системы и приложений. Реестр выполняет ту же роль, что и INI-файлы в Windows 3.1. Windows 95 по-прежнему позволяет использовать INI-файлы, однако в первую очередь они обеспечивают обратную совместимость. Преимущества реестра — возможность присоединить к одному ключу множество элементов различных типов. Кроме того, сетевые средства обеспечивают доступ к реестру по сети для удаленного администрирования и диагностики.

Редактор реестра

В Windows 95 и Windows NT реестр можно просматривать и редактировать средствами редактора реестра REGEDIT.EXE, расположенного в папке Windows (рис. 2.10).

Будьте осторожны, изменяя элементы реестра при помощи редактора реестра, — он не распознает синтаксические и семантические ошибки и не предупреждает о создании некорректного элемента. Неверный элемент реестра может сделать систему неработоспособной. К счастью, в большинстве случаев Вам не придется модифицировать реестр напрямую — для изменения параметров системы чаще всего достаточно диспетчера устройств и других апплетов Панели управления.

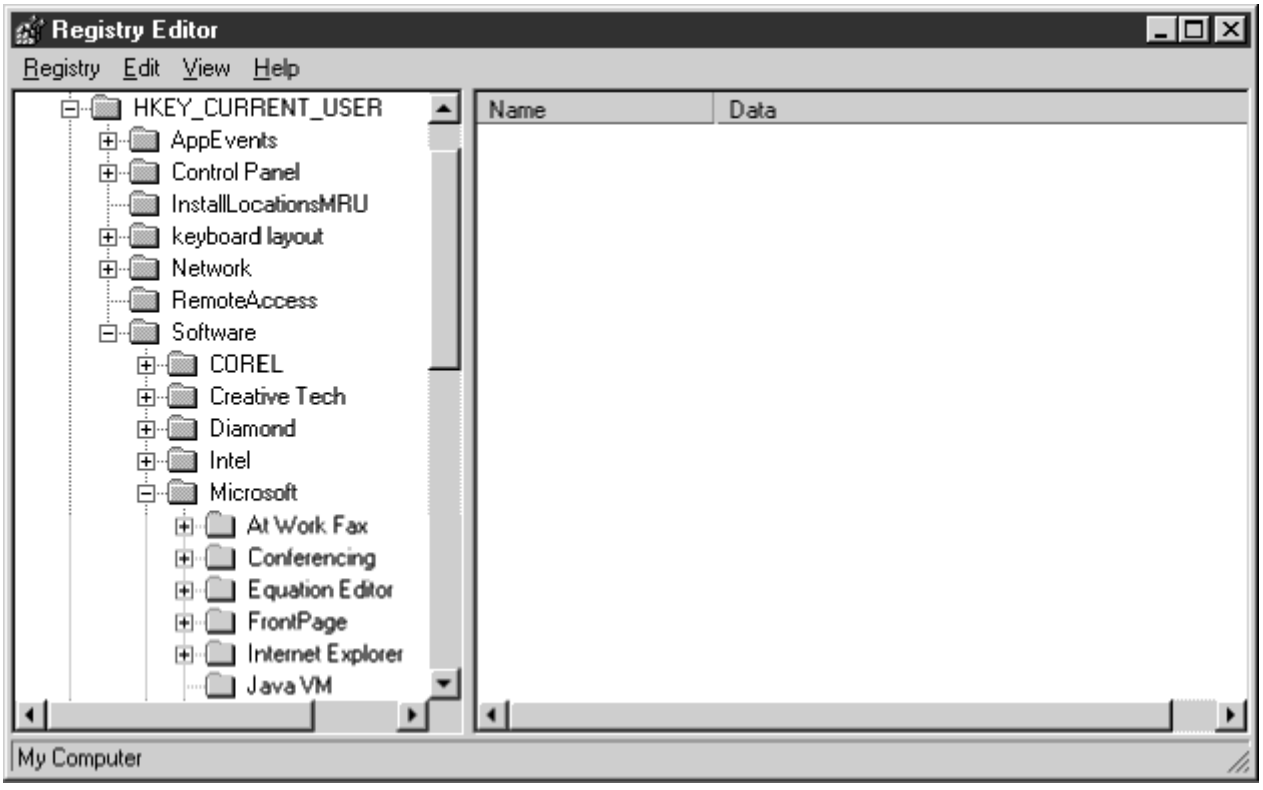

Рис. 2.10 Редактор реестра Windows

Структура реестра

Реестр — это древовидная иерархическая база данных. Он хранится в двух файлах, состав которых определяется конфигурацией системы. Обычно их два: один содержит настройки, специфичные для пользователя (файл USER.DAT), а другой — настройки, специфичные

для компьютера (обычно SYSTEM.DAT). Каждый узел иерархического дерева называется *ключом.* Реестр напоминает файловую систему: любой ключ может содержать вложенные ключи (аналог вложенных каталогов) и данные (аналог файлов). В ключе хранится произвольное число значений данных любого типа. Каждое значение называется *элементом реестра.* Компоненты ключей реестра перечислены ниже.

Таблица 2.5.1

Компонент ключа	Обязательный	Описание
Имя	Да	Строка, используемая для доступа к ключу. Должна быть уникальной среди других ключей того же уровня иерархии
Класс	Нет	Имя класса объекта. Предназначен для использования в коде методов класса, экземпляры которого хранятся в реестре, Приложениями обычно не используется
Дескриптор защиты	Нет	Ключи содержат стандартные дескрипторы защиты Windows NT, допускают управление доступом и могут быть подвергнуты аудиту
Время последней записи	Нет	Время, когда ключ был последний раз модифицирован. Любое изменение элемента считается изменением его родительского ключа
Элемент(ы)	Нет	Информация, хранящаяся в ключе: имя для идентификации значения, тип для определения типа данных и сами данные соответствующей длины и формата

HKEY CLASSES ROOT : Ключ HKEY_CLASSES_ROOT содержит те же данные, что и файл REG.DAT в Windows 3.1,— сведения о встраивании и связывании объектов (Object Linking and Embedding, OLE) и ассоциации файлов с приложениями, которые позволяют Windows запускать приложение, соответствующее выбранному файлу.

HKEY_LOCAL_MACHINE: Этот ключ содержит спецификации рабочей станции, драйверов и другие системные настройки, включая информацию о типах установленного оборудования, настройках портов, конфигурации программного обеспечения и т.п. Эта информация специфична для компьютера, а не для пользователя.

HKEY_CURRENT.CONFIG: Этот ключ содержит информацию о текущей конфигурации аппаратуры компьютера и используется в основном на компьютерах с несколькими аппаратными конфигурациями, например при подключении портативного компьютера к стыковочной станции и отключении от нее. Информация, содержащаяся в этом ключе, копируется из ключа HKEY_LOCAL_MACHINE.

HKEY_USERS: Это ключ содержит информацию обо всех пользователях данной рабочей станции. Здесь хранятся данные о каждом пользователе, а также типовые настройки, служащие шаблоном для новых ключей, создаваемых пользователем. Типовые настройки включают различные значения по умолчанию для программ, схем событий, конфигураций рабочего стола и т.п.

HKEY_CURRENT_USER: Этот ключ содержит настройки системы и программ, относящиеся к текущему пользователю. Он создается при регистрации пользователя в системе на основе информации из соответствующего раздела ключа HKEY_USERS. Именно здесь хранятся сведения о том, как данный пользователь сконфигурировал рабочую станцию — например, данные том, что каждый старт системы должен сопровождаться звуковым эффектом. Прочая информация может включать цветовые схемы, ярлыки, состояние рабочего стола и т.п.

HKEY_DYN_DATA: Этот ключ содержит динамическую информацию о состоянии различных устройств, причем она создается заново при каждом старте системы. Ключ HKEY_DYN_DATA используется как часть системы измерения производительности и для конфигурации устройств Plug-and-Play. Информация, содержащаяся здесь, меняется при добавлении новых устройств и удалении существующих. Для каждого устройства это данные о соответствующем аппаратном ключе, известных проблемах и текущем состоянии устройства. Ключ HKEY_DYN_DATA **также** содержит сведения о состоянии системы, формируемые с помощью утилиты System Monitor. Это ключ не входит в состав файлов реестра и всегда создается динамически.

Модификация реестра Windows

Работать с приложением гораздо приятнее, если при каждом запуске оно сохраняет информацию о действиях и предпочтениях пользователя. Эти данные можно применять и далее — например, сохранить имя последней базы данных, открытой пользователем, и указать его в качестве имени по умолчанию, когда пользователь обратится к базе данных в следующий раз.

Для сохранения параметров приложения в реестре применяются операторы Visual Basic **SaveSetting и GetSetting** и соответствующие функции API Windows. Они имеют следующий синтаксис:

- **SaveSetting** *(приложение, раздел, ключ, параметр)*
- **GetSetting** *(приложение, раздел, ключ, [по умолчанию])*

Пример В приведенном ниже коде с помощью оператора **SaveSetting** создаются элементы реестра для приложения, заданного в аргументе приложение, а затем применяется оператор **GetSetting** для получения значений параметров. Поскольку задано значение по умолчанию, возврат значения гарантирован.

Поместить настройки в реестр

SaveSetting "MyApp","Startup", "Top", 75

SaveSetting "MyApp","Startup", "Left", 50

использовать настройки из реестра для отображения текущей формы

```
Me.Left = GetSetting(appname := "MyApp",
        section := "Startup", key := "Left", default := "0")
Me.Top = GetSetting(appname := "MyApp",
        section := "Startup", key := "Top", default := "0")
```

Резюме

Изучение операционной системы Windows следует начинать с архитектуры системы. Операционная система Windows для поддержки своей эффективности и целостности использует два режима: пользователя и ядра.

Windows 95 и Windows NT — многозадачные операционные системы. Однако в действительности одновременно не выполняется больше одного процесса — процессор переключается между ними. В Windows 95 и NT каждый процесс имеет собственное адресное пространство, что позволяет адресовать до 4 Гб памяти.

Интерфейс прикладного программирования (API) Win32 обеспечивает приложениям доступ ко всему спектру функций операционных систем семейства Windows. Функции, сообщения и структуры Win32 образуют последовательный и единообразный API для Windows 95 и Windows NT. Средства API Win32 позволяют разрабатывать приложения, которые успешно работают на всех платформах, в то же время сохраняя возможность использовать уникальные особенности любой из них.

Реестр — это унифицированная база данных, где хранится информация об аппаратной и программной конфигурации локального компьютера. Преимущества реестра — возможность присоединить к одному ключу множество элементов различных типов. Кроме того, сетевые средства обеспечивают доступ к реестру по сети для удаленного администрирования и диагностики.