5. "Классические" ОСРВ. Объектно-ориентированные системы.

5.1 Windows NT. Основные характеристики.

С середины 1993 года Microsoft начала выпуск новых операционных систем "новой технологии" (New Technology - NT) Windows NT.

Операционная система Windows NT с самого начала проектировалась с учетом всех требований, предъявляемых к современным ОС:

- расширяемости,
- переносимости,
- надежности,
- совместимости,
- производительности.

Эти свойства были достигнуты за счет применения передовых технологий структурного проектирования, таких как ***клиент-сервер, микроядра, объекты***.

В Windows NT используется ***механизм многозадачности с вытеснением*** (preemptive multitasking). Для управления нитями Windows NT Server использует механизм приоритетов. В определенные моменты производятся оценка приоритетов и перераспределение нитей по процессорам, в результате чего последовательные стадии одного потока программы могут выполняться разными процессорами или откладываться до высвобождения очередного процессора.

Windows NT поддерживает симметричную многопроцессорную организацию вычислительного процесса (СМП), в соответствии с которой ОС может выполняться на любом свободном процессоре или на всех процессорах одновременно, разделяя память между ними. Учитывая, что многозадачность реализуется на уровне нитей, разные части одного и того же процесса могут действительно выполняться параллельно. Следовательно, многонитевые серверы могут обслуживать более одного клиента. Windows NT Server поддерживает до 16 параллельных процессоров, что актуально для таких серверов, как Symmetry 750 фирмы Sequent с 16 процессорами Intel. Следует, однако, иметь в виду, что реализация СМП в Windows NT Server нацелена на оптимизацию производительности и не обеспечивает резервирования в целях повышения отказоустойчивости. В случае выхода из строя одного из процессоров система останавливается.

В Windows NT Server в полной мере реализован потенциал масштабируемости архитектуры СМП. Однопроцессорную систему можно легко развивать, наращивая число процессоров, без замены версии ОС или приложений.

При управлении устройствами ввода/вывода Windows NT Server использует асинхронный подход. Для завершения процесса и начала выполнения новой задачи не нужно ждать поступления сигнала об окончании таких операций, как чтение или запись.

Каждый процесс создается с использованием одной нити, которая служит специфическим отображением выполнения программы процессором. Впоследствии программа может создавать новые нити, и Windows NT Server будет распределять их и управлять ими, не привлекая к этому приложения высокого уровня.

Для того чтобы прикладная программа могла использовать несколько потоков, не нужно предусматривать этого в ее алгоритме. Отдельный поток создается для каждой операции. Например, в одном потоке программа может воспроизводить сложную графическую форму, а

другой использовать для редактирования объемного чертежа. Каждый из этих потоков (или, с точки зрения пользователя, операций) работает на отдельном процессоре, не требуя никаких управляющих вмешательств со стороны приложения. Потоки внутри процесса используют общую область памяти и, следовательно, не должны специально обмениваться данными.

В соответствии с требованием совместимости, Windows NT обеспечивает среду выполнения не только для приложений с исходным программным интерфейсом Win32 API. При выполнении на процессорах фирмы Intel защищенные подсистемы Windows NT обеспечивают двоичную совместимость существующих приложений фирмы Microsoft, включая MS-DOS, Win16, OS/2. На MIPS RISC процессорах двоичная совместимость достигается для приложений MS-DOS и 16-битных Windows-приложений (с использованием эмуляции). Windows NT обеспечивает также совместимость на уровне исходных текстов для POSIX-приложений, которые твердо придерживаются интерфейса, определенного в стандарте IEEE 1003.1.

Помимо совместимости программных интерфейсов, Windows NT поддерживает существующие файловые системы, включая файловую систему MS-DOS (FAT), файловую систему CD-ROM, файловую систему OS/2 (HPFS) и собственную файловую систему (NTFS).

В отличие от большинства других операционных систем, Windows NT изначально разрабатывался с учетом возможности работы в сети. В результате этого функции совместного использования файлов, устройств и объектов встроены в интерфейс с пользователем. Администраторы могут централизованно управлять и контролировать работу сетей в масштабах крупных предприятий. Особенно важно отметить возможность распространения работы приложений типа клиент-сервер на многокомпьютерные системы.

5.2 Структура Windows NT

При разработке структуры Windows NT была в значительной степени использована концепция микроядра. В соответствии с этой идеей ОС разделена на несколько подсистем, каждая из которых выполняет отдельный набор сервисных функций – например, сервис памяти, сервис по созданию процессов, или сервис по планированию процессов.. Рассмотрим более подробно, как это работает.

Структурно Windows NT может быть представлена в виде двух частей (рисунок 1):

1. часть операционной системы, работающая в режиме пользователя,
2. и часть операционной системы, работающая в режиме ядра.

Часть Windows NT, работающая в режиме ядра, называется ***executive*** - исполнительной частью. Она включает ряд компонент, которые управляют виртуальной памятью, объектами (ресурсами), вводом-выводом и файловой системой (включая сетевые драйверы), взаимодействием процессов и частично системой безопасности. Эти компоненты взаимодействуют между собой с помощью межмодульной связи. Каждая компонента вызывает другие с помощью набора тщательно специфицированных внутренних процедур.

Вторую часть Windows NT, работающую в режиме пользователя, составляют серверы - так называемые защищенные подсистемы. Серверы Windows NT называются защищенными подсистемами, так как каждый из них выполняется в отдельном процессе, память которого отделена от других процессов системой управления виртуальной памятью NT executive. Так как подсистемы автоматически не могут совместно использовать память, они общаются друг с другом посредством посылки сообщений. Сообщения могут передаваться как между клиентом и сервером, так и между двумя серверами. Все сообщения проходят через исполнительную часть Windows NT. Ядро Windows NT планирует нити защищенных подсистем точно так же, как и нити обычных прикладных процессов.

Поддержку защищенных подсистем обеспечивает исполнительная часть - Windows NT executive, которая работает в пространстве ядра и никогда не сбрасывается на диск. Ее составными частями являются:

Менеджер объектов.

Создает, удаляет и управляет объектами NT executive - абстрактными типами данных, используемых для представления ресурсов системы.

Монитор безопасности.

Устанавливает правила защиты на локальном компьютере. Охраняет ресурсы операционной системы, выполняет защиту и регистрацию исполняемых объектов.

Менеджер процессов.

Создает и завершает, приостанавливает и возобновляет процессы и нити, а также хранит о них информацию.

Менеджер виртуальной памяти.

Подсистема ввода-вывода.

Включает в себя следующие компоненты:

- менеджер ввода-вывода, предоставляющий средства ввода-вывода, независимые от устройств;
- файловые системы - NT-драйверы, выполняющие файл-ориентированные запросы на ввод-вывод и транслирующие их в вызовы обычных устройств;
- сетевой редиректор и сетевой сервер - драйверы файловых систем, передающие удаленные запросы на ввод-вывод на машины сети и получающие запросы от них;
- драйверы устройств NT executive - низкоуровневые драйверы, которые непосредственно управляют устройством;
- менеджер кэша, реализующий кэширование диска.

Исполнительная часть, в свою очередь, основывается на службах нижнего уровня, предоставляемых ядром (его можно назвать и микроядром) NT.

В функции ***ядра*** входит:

- планирование процессов,
- обработка прерываний и исключительных ситуаций,
- синхронизация процессоров для многопроцессорных систем,
- восстановление системы после сбоев.

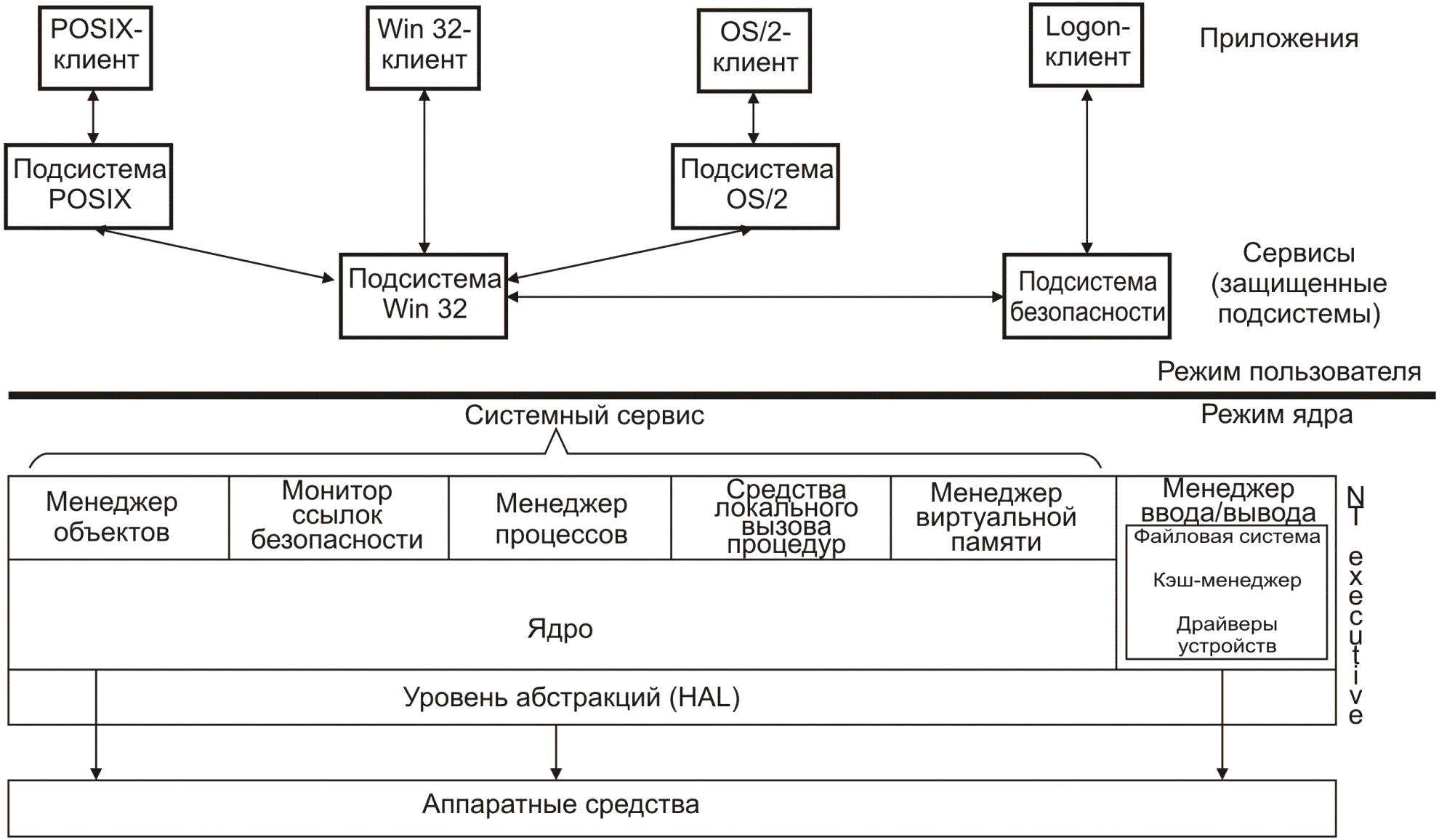

Рисунок 1 Структура Windows NT

Ядро работает в привилегированном режиме и никогда не удаляется из памяти. Обратиться к ядру можно только посредством прерывания. Ядро расположено над уровнем аппаратных абстракций (Hardware Abstraction Level HAL), который концентрирует в одном месте большую часть машинно-зависимых процедур. HAL располагается между NT executive и аппаратным обеспечением и скрывает от системы такие детали, как контроллеры прерываний, интерфейсы ввода/вывода и механизмы взаимодействия между процессорами. Такое решение позволяет легко переносить Windows NT с одной платформы на другую путем замены только слоя HAL.

При создании NT разработчики руководствовались задачами улучшения производительности и сетевых возможностей, а также требованием поддержки определенного набора прикладных сред. Эта цель была достигнута продуманным разделением функций между модулями ядра и остальными модулями. Например, передача данных в файловую систему и по сети производится быстрее в пространстве ядра, поэтому внутри ядра NT выделены буфера для небольших по объему (от 16 до 32 Кб) операций чтения и записи, являющихся типичными для приложений клиент-сервер и распределенных приложений. Размещение этих функций ввода-вывода внутри ядра, может, и портит академическую чистоту микроядра NT, но соответствует цели создания NT.

Защищенные подсистемы Windows NT работают в пользовательском режиме и создаются Windows NT во время загрузки операционной системы. Сразу после создания они начинают бесконечный цикл своего выполнения, отвечая на сообщения, поступающие к ним от прикладных процессов и других подсистем. Среди защищенных подсистем можно выделить подкласс, называемый подсистемами окружения. Подсистемы окружения реализуют интерфейсы приложений операционной системы (API). Другие типы подсистем, называемые интегральными подсистемами, исполняют необходимые операционной системе задачи. Например, большая часть системы безопасности Windows NT реализована в виде интегральной подсистемы, сетевые серверы также выполнены как интегральные подсистемы.

Наиболее важной подсистемой окружения является Win32 - подсистема, которая обеспечивает доступ для приложений к 32-bit Windows API. Дополнительно эта система обеспечивает графический интерфейс с пользователем и управляет вводом/выводом данных пользователя. Также поддерживаются подсистемы POSIX, OS/2,16-разрядная Windows и MS-DOS.

Каждая защищенная подсистема работает в режиме пользователя, вызывая системный сервис NT executive для выполнения привилегированных действий в режиме ядра. Сетевые серверы могут выполняться как в режиме пользователя, так и в режиме ядра, в зависимости от того, как они разработаны.

Подсистемы связываются между собой путем передачи сообщений. Когда, например, пользовательское приложение вызывает какую-нибудь API-процедуру, подсистема окружения, обеспечивающая эту процедуру, получает сообщение и выполняет ее либо обращаясь к ядру, либо посылая сообщение другой подсистеме. После завершения процедуры подсистема окружения посылает приложению сообщение, содержащее возвращаемое значение. Посылка сообщений и другая деятельность защищенных подсистем невидима для пользователя.

Основным средством, скрепляющим все подсистемы Windows NT в единое целое, является механизм вызова локальных процедур (Local Procedure Call - LPC). LPC представляет собой оптимизированный вариант более общего средства - удаленного вызова процедур (RPC), которое используется для связи клиентов и серверов, расположенных на разных машинах сети.

Средства LPC поддерживают несколько способов передачи данных между клиентами и серверами: один обычно используется для передачи коротких сообщений, другой - для длинных сообщений, а третий оптимизирован специально для использования подсистемой Win32. Каждая подсистема устанавливает порт - канал связи, посредством которого с ней могут связываться другие процессы.

Микроядро NT служит, главным образом, средством поддержки для переносимой основной части ОС - набора пользовательских сред. Концентрация машинно-зависимых программ внутри микроядра делает перенос NT на разнообразные процессоры относительно легким. Из операционной системы Windows NT ядро вряд ли может быть вычленено для отдельного использования. Это является одной из причин того, что некоторые специалисты не считают

Windows NT истинно микроядерной ОС в том смысле, в котором таковыми являются Mach и Chorus.

5.3 Концепции Windows NT

Множественные прикладные среды

При разработке NT важнейшим рыночным требованием являлось обеспечение поддержки по крайней мере двух уже существующих программных интерфейсов - OS/2 и POSIX, а также возможности добавления других API в будущем.
Для того чтобы программа, написанная для одной ОС, могла быть выполнена в рамках другой, недостаточно лишь обеспечить совместимость API. Кроме этого необходимо обеспечить ей <родное> окружение: структуру процесса, средства управления памятью, средства обработки ошибок и исключительных ситуаций, механизмы защиты ресурсов и семантику файлового доступа. Отсюда ясно, что поддержка нескольких прикладных программных сред является очень сложной задачей, тесно связанной со структурой операционной системы. Она была успешно решена в Windows NT, при этом в полной мере использовался опыт разработчиков ОС Mach из университета Карнеги-Меллона, которые смогли в своей клиент-серверной реализации UNIX отделить базовые механизмы операционной системы от серверов API различных ОС.

Windows NT поддерживает 5 прикладных сред операционных систем: MS-DOS, 16-разрядный Windows, OS/2 1.x, POSIX и 32-разрядный Windows (Win32). Все они реализованы как подсистемы окружения. Каждая работает в собственном защищенном пользовательском пространстве. Подсистема Win32 обеспечивает поддержку дисплея, клавиатуры и мыши для четырех остальных подсистем. 16-битовые приложения DOS и Windows работают на VDM (virtual DOS machines - виртуальные машины DOS), каждая из которых эмулирует полный (80x86) процессор с MS-DOS. В NT VDM является приложением Win32, значит, как и обычные модули прикладных сред для UNIX, приложения DOS и 16-битовой Windows расположены в слое непосредственно над подсистемой Win32. Подсистемы OS/2 и POSIX построены по-другому. В качестве полноценных подсистем NT они могут взаимодействовать с подсистемой Win32 для получения доступа к вводу и выводу, но также могут обращаться непосредственно к исполнительной системе NT за другими средствами операционной системы. Подсистема OS/2 может выполнять многие имеющиеся приложения OS/2 символьного режима, включая OS/2 SQL Server, и поддерживает именованные каналы и NetBIOS. Однако возможности подсистемы POSIX весьма ограничены, несмотря на ее непосредственный доступ к службам ядра. Приложения POSIX должны быть откомпилированы специально для Windows NT. NT не поддерживает двоичный код, предназначенный для других POSIX-совместимых систем, таких, как UNIX. К тому же подсистема POSIX NT не поддерживает непосредственно печать, сетевой доступ (за исключением доступа к удаленным файловым системам) и многие средства Win32, например отображение на память файлов и графику.
NT executive выполняет базовые функции операционной системы и является той основой, на которой подсистемы окружения реализуют поддержку своих приложений. Все подсистемы равноправны и могут вызвать <родные> функции NT для создания соответствующей среды для своих приложений.
Каждая подсистема окружения имеет свое представление о том, что такое, например, процесс или описатель файла, поэтому структуры данных, используемые в каждом окружении, могут не совпадать. Следовательно, как только подсистема Win32 передала прикладной процесс другой подсистеме окружения, данное приложение становится клиентом этой подсистемы вплоть до завершения процесса. При этом подсистема Win32 перенаправляет входные сообщения от пользователя этому приложению, а также отображает вывод приложения на экране.

Объектно-ориентированный подход.

Хотя NT и не является полностью объектно-ориентированной, в ее основе лежат объекты. Единообразная форма именования, совместного использования и учета системных ресурсов, простой и дешевый способ обеспечения безопасности системы и ее модификации - все эти преимущества могут быть достигнуты при использовании объектной модели.

В Windows NT любой ресурс системы, который одновременно может быть использован более чем одним процессом, включая файлы, совместно используемую память и физические устройства, реализован в виде объекта и управляется рядом функций. Такой подход сокращает число изменений, которые необходимо внести в операционную систему в процессе ее эксплуатации. Аналогично, если требуется поддержка новых ресурсов, то надо добавить только новый объект, не изменяя при этом остального кода операционной системы.

Наиболее фундаментальное отличие между объектом и обыкновенной структурой данных заключается в том, что внутренняя структура данных объекта скрыта от наблюдения. Это отделяет средства реализации объекта от кода, который только использует его, такая техника позволяет легко изменять в последствии реализацию объектов.

Группа разработчиков NT executive решила использовать объекты для представления системных ресурсов, потому что объекты обеспечивают централизованные средства для выполнения трех важных задач ОС:

- Поддержка воспринимаемых человеком имен системных ресурсов;
- Разделение ресурсов и данных между процессами;
- Защита ресурсов от несанкционированного доступа.

Не все структуры данных в NT executive являются объектами. Объектами сделаны только такие данные, которые нужно разделять, защищать, именовать или делать видимыми для программ пользовательского режима (с помощью системных функций). Структуры, которые используются только одним компонентом executive для выполнения внутренних функций, не являются объектами.

Менеджер объектов - это компонента NT executive, которая ответственна за создание, удаление, защиту и слежение за NT-объектами. Менеджер объектов централизует операции управления ресурсами, которые в противном случае будут разбросаны по всей ОС.

Менеджер объектов NT выполняет следующие функции:

- Выделяет память для объекта.
- Присоединяет к объекту так называемый дескриптор безопасности, который определяет, кому разрешено использовать объект, и что они могут с ним делать.
- Создает и манипулирует структурой каталога объектов, в котором хранятся имена объектов.
- Создает описатель объекта и возвращает его вызывающему процессу.

Процессы пользовательского режима, включая подсистемы окружения, должны иметь описатель объекта перед тем, как их нити смогут использовать этот объект.

Каждый NT-объект является объектом определенного типа. Тип определяет данные, которые хранит объект, и "родные" системные функции, которые могут к нему применяться. Для того, чтобы управлять различными объектами единообразно, менеджер объектов требует, чтобы каждый объект содержал несколько полей стандартной информации в определенном месте объекта. До тех пор, пока эти данные имеются, менеджер объектов не заботится о том, что еще хранится в объекте.

Каждый объект состоит из двух частей

- заголовка объекта
- тела объекта,

которые содержат стандартные и переменные данные объекта соответственно. Менеджер объектов работает с заголовком объекта, а другие компоненты executive работают с телами объектов тех типов, которые они сами создают. Заголовок объекта используется менеджером без учета типа объекта.

В заголовке объекта любого типа содержится

- имя,
- каталог,
- дескриптор безопасности,
- квоты на использование ресурсов,
- счетчик открытых описателей,
- база данных открытых описателей,

- признак постоянный/временный,
- режим пользователя/ядра,
- указатель на тип объекта.

Кроме заголовка объекта, каждый объект имеет тело объекта, формат и содержание которого уникально определяется типом этого объекта; у всех объектов одного и того же типа одинаковый формат тела. При создании объекта исполнительная часть может оперировать данными в телах всех объектов этого типа.

Процессы и нити.

Понятия процессов и нитей являются одними из ключевых понятий в современных операционных системах, поэтому стоит уделить им несколько большее внимание. В разных ОС процессы реализуются по-разному. Эти различия заключаются в том, какими структурами данных представлены процессы, как они именуются, какими способами защищены друг от друга и какие отношения существуют между ними.

Процессы Windows NT имеют следующие характерные свойства:

- Процессы Windows NT реализованы в форме объектов, и доступ к ним осуществляется посредством службы объектов.
- Процесс Windows NT имеет многонитевую организацию.
- Как объекты-процессы, так и объекты-нити имеют встроенные средства синхронизации.
- Менеджер процессов Windows NT не поддерживает между процессами отношений типа "родитель-потомок".

В любой системе понятие "процесс" включает следующее:

- исполняемый код,
- собственное адресное пространство, которое представляет собой совокупность виртуальных адресов, которые может использовать процесс,
- ресурсы системы, такие как файлы, семафоры и т.п., которые назначены процессу операционной системой.
- хотя бы одну выполняемую нить.

Адресное пространство каждого процесса защищено от вмешательства в него любого другого процесса. Это обеспечивается механизмами виртуальной памяти. Операционная система, конечно, тоже защищена от прикладных процессов. Чтобы выполнить какую-либо процедуру ОС или прочитать что-либо из ее области памяти, нить должна выполняться в режиме ядра. Пользовательские процессы получают доступ к функциям ядра посредством системных вызовов. В пользовательском режиме выполняются не только прикладные программы, но и защищенные подсистемы Windows NT.

В Windows NT процесс - это просто объект, создаваемый и уничтожаемый менеджером объектов. Объект-процесс, как и другие объекты, содержит заголовок, который создает и инициализирует менеджер объектов. Менеджер процессов определяет атрибуты, хранимые в теле объекта-процесса, а также обеспечивает системный сервис, который восстанавливает и изменяет эти атрибуты.

В число атрибутов тела объекта-процесса входят:

- Идентификатор процесса - уникальное значение, которое идентифицирует процесс в рамках операционной системы.
- Токен доступа - исполняемый объект, содержащий информацию о безопасности.
- Базовый приоритет - основа для исполнительного приоритета нитей процесса.
- Процессорная совместимость - набор процессоров, на которых могут выполняться нити процесса.
- Предельные значения квот - максимальное количество страничной и нестраничной системной памяти, дискового пространства, предназначенного для выгрузки страниц, процессорного времени - которые могут быть использованы процессами пользователя.
- Время исполнения - общее количество времени, в течение которого выполняются все нити процесса.

Нить является выполняемой единицей, которая располагается в адресном пространстве процесса и использует ресурсы, выделенные процессу. Подобно процессу нить в Windows NT реализована в форме объекта и управляется менеджером объектов.

Объект-нить имеет следующие атрибуты тела:

- Идентификатор клиента - уникальное значение, которое идентифицирует нить при ее обращении к серверу.
- Контекст нити - информация, которая необходима ОС для того, чтобы продолжить выполнение прерванной нити. Контекст нити содержит текущее состояние регистров, стеков и индивидуальной области памяти, которая используется подсистемами и библиотеками.
- Динамический приоритет - значение приоритета нити в данный момент.
- Базовый приоритет - нижний предел динамического приоритета нити.
- Процессорная совместимость нитей - перечень типов процессоров, на которых может выполняться нить.
- Время выполнения нити - суммарное время выполнения нити в пользовательском режиме и в режиме ядра, накопленное за период существования нити.
- Состояние предупреждения - флаг, который показывает, что нить должна выполнять вызов асинхронной процедуры.
- Счетчик приостановок - текущее количество приостановок выполнения нити.

Алгоритм планирования процессов и нитей.

В Windows NT реализована вытесняющая многозадачность, при которой операционная система не ждет, когда нить сама захочет освободить процессор, а принудительно снимает ее с выполнения после того, как та израсходовала отведенное ей время (квант), или если в очереди готовых появилась нить с более высоким приоритетом. При такой организации разделения процессора ни одна нить не займет процессор на очень долгое время.

В ОС Windows NT нить в ходе своего существования может иметь одно из шести состояний (рисунок 2). Жизненный цикл нити начинается в тот момент, когда программа создает новую нить. После инициализации нить проходит через следующие состояния:

Готовность.

При поиске нити на выполнение диспетчер просматривает только нити, находящиеся в состоянии готовности, у которых есть все для выполнения, но не хватает только процессора.

Первоочередная готовность (standby).

Для каждого процессора системы выбирается одна нить, которая будет выполняться следующей (самая первая нить в очереди). Когда условия позволяют, происходит переключение на контекст этой нити.

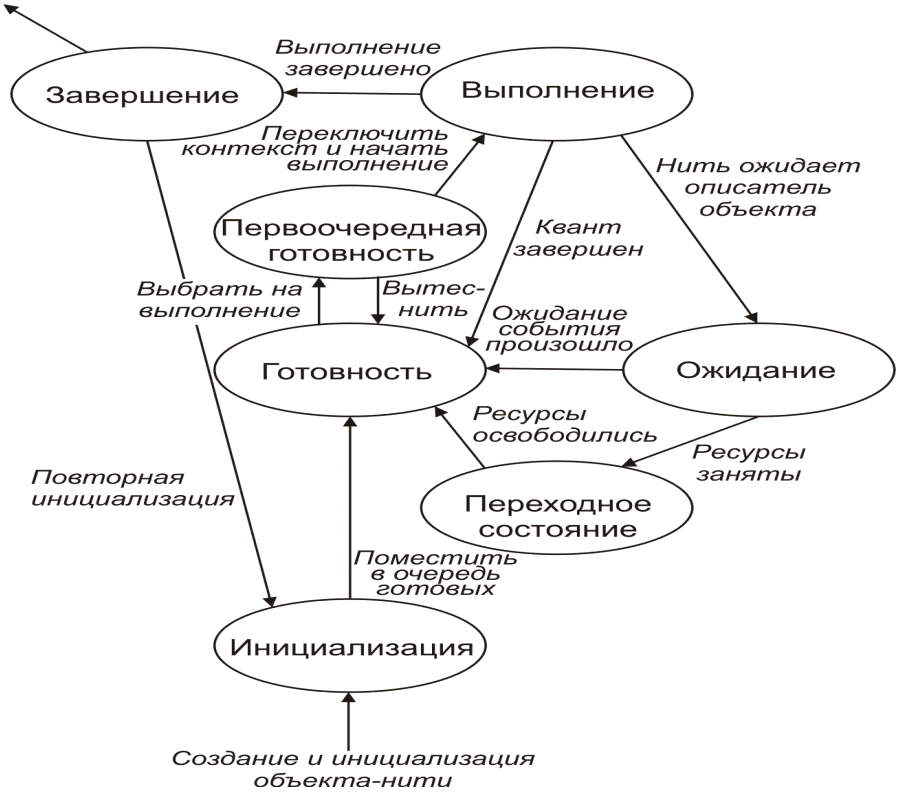

Рисунок 2. Граф состояний нити

Выполнение.

Как только происходит переключение контекстов, нить переходит в состояние выполнения и находится в нем до тех пор, пока либо ядро не вытеснит ее из-за того, что появилась более приоритетная нить или закончился квант времени, выделенный этой нити, либо нить завершится вообще, либо она по собственной инициативе перейдет в состояние ожидания.

Ожидание.

Нить может входить в состояние ожидания несколькими способами: нить по своей инициативе ожидает некоторый объект для того, чтобы синхронизировать свое выполнение; операционная система (например, подсистема ввода-вывода) может ожидать в интересах нити; подсистема окружения может непосредственно заставить нить приостановить себя. Когда ожидание нити подойдет к концу, она возвращается в состояние готовности.

Переходное состояние.

Нить входит в переходное состояние, если она готова к выполнению, но ресурсы, которые ей нужны, заняты. Например, страница, содержащая стек нити, может быть выгружена из оперативной памяти на диск. При освобождении ресурсов нить переходит в состояние готовности.

Завершение.

Когда выполнение нити закончилось, она входит в состояние завершения. Находясь в этом состоянии, нить может быть либо удалена, либо не удалена. Это зависит от алгоритма работы менеджера объектов, в соответствии с которым он и решает, когда удалять объект. Если executive имеет указатель на объект-нить, то она может быть инициализирована и использована снова.

Диспетчер ядра использует для определения порядка выполнения нитей алгоритм, основанный на приоритетах, в соответствии с которым каждой нити присваивается число - приоритет, и нити с более высоким приоритетом выполняются раньше нитей с меньшим приоритетом. В самом начале нить получает приоритет от процесса, который создает ее. В свою очередь, процесс получает приоритет в тот момент, когда его создает подсистема той или иной прикладной среды. Значение базового приоритета присваивается процессу системой по умолчанию или системным администратором. Нить наследует этот базовый приоритет и может изменить его, немного увеличив или уменьшив. На основании получившегося в результате

приоритета, называемого приоритетом планирования, начинается выполнение нити. В ходе выполнения приоритет планирования может меняться.

Windows NT поддерживает 32 уровня приоритетов, разделенных на два класса

- класс реального времени
- класс переменных приоритетов.

Нити реального времени, приоритеты которых находятся в диапазоне от 16 до 31, являются более приоритетными процессами и используются для выполнения задач, критичных ко времени.

Каждый раз, когда необходимо выбрать нить для выполнения, диспетчер прежде всего просматривает очередь готовых нитей реального времени и обращается к другим нитям, только когда очередь нитей реального времени пуста. Большинство нитей в системе попадают в класс нитей с переменными приоритетами, диапазон приоритетов которых от 0 до 15. Этот класс имеет название "переменные приоритеты" потому, что диспетчер настраивает систему, выбирая (понижая или повышая) приоритеты нитей этого класса.

Алгоритм планирования нитей в Windows NT объединяет в себе обе базовых концепции - квантование и приоритеты. Как и во всех других алгоритмах, основанных на квантовании, каждой нити назначается квант, в течение которого она может выполняться. Нить освобождает процессор, если:

- блокируется, уходя в состояние ожидания;
- завершается;
- исчерпан квант;
- в очереди готовых появляется более приоритетная нить.

Использование динамических приоритетов, изменяющихся во времени, позволяет реализовать адаптивное планирование, при котором не дискриминируются интерактивные задачи, часто выполняющие операции ввода-вывода и недоиспользующие выделенные им кванты. Если нить полностью исчерпала свой квант, то ее приоритет понижается на некоторую величину. В то же время приоритет нитей, которые перешли в состояние ожидания, не использовав полностью выделенный им квант, повышается. Приоритет не изменяется, если нить вытеснена более приоритетной нитью.

Для того чтобы обеспечить хорошее время реакции системы, алгоритм планирования использует наряду с квантованием концепцию абсолютных приоритетов. В соответствии с этой концепцией при появлении в очереди готовых нитей такой, у которой приоритет выше, чем у выполняющейся в данный момент, происходит смена активной нити на нить с самым высоким приоритетом.

В многопроцессорных системах при диспетчеризации и планировании нитей играет роль их процессорная совместимость: после того, как ядро выбрало нить с наивысшим приоритетом, оно проверяет, какой процессор может выполнить данную нить и, если атрибут нити "процессорная совместимость" не позволяет нити выполняться ни на одном из свободных процессоров, то выбирается следующая в порядке приоритетов нить.

Обработка прерываний в NT

В Windows NT единственный способ управления аппаратурой - через драйвер устройства. Поскольку приложение реального времени имеет дело с внешними событиями, разработчик должен написать и включить в ядро драйвер устройства, дающий доступ к аппаратуре. Этот драйвер реагирует на прерывания, генерируемые соответствующим устройством.

Прерывания обрабатываются в два этапа.

- Сначала выполняется очень короткая программа обслуживания прерываний (ISR).
- Впоследствии работа завершается выполнением DPC - процедуры отложенного вызова.

Возникает следующий поток событий:

- Происходит прерывание.
- Процессор сохраняет PC, SP и вызывает диспетчер.
- ОС сохраняет контекст и вызывает ISR.
- В ISR выполняется критическая работа (чтение/запись аппаратных регистров).

- DPC ставится в очередь.
- ОС восстанавливает контекст.
- Процессор восстанавливает PC, SP.
- Ожидающие в очереди DPC выполняются на уровне приоритета DISPATCH_LEVEL.
- После завершения всех DPC ОС переходит к выполнению приложений.

ISR должно быть как можно короче, поэтому большинство драйверов выполняют значительную часть работы в DPC (которая может быть вытеснена другим ISR), ожидая окончания других DPC, ранее попавших в очередь (все DPC имеют одинаковый приоритет).

Из документации по NT следует, что ISR может быть вытеснена другим ISR с более высоким приоритетом, и что DPC имеет более высокий приоритет, чем пользовательские и системные нити. Но поскольку все DPC имеют одинаковый уровень приоритета и ISR должна быть сведена к минимуму, ваш DPC будет вынужден ждать других и ваше приложение будет зависеть от остальных драйверов устройств непредсказуемым образом. Задержки системных вызовов обусловлены именно тем, что DPC от драйверов жесткого диска и сети блокируют все другие.

В настоящих ОС РВ разработчик знает, на каком уровне выполняются драйверы всех других устройств. Обычно имеется свободное пространство для прерываний выше стандартных драйверов. Вся критическая работа выполняется в ISR, что позволяет настроить конфигурацию драйвера в зависимости от временных ограничений приложения.

Сетевые средства.

Средства сетевого взаимодействия Windows NT направлены на реализацию взаимодействия с существующими типами сетей, обеспечение возможности загрузки и выгрузки сетевого программного обеспечения, а также на поддержку распределенных приложений.

Windows NT с точки зрения реализации сетевых средств имеет следующие особенности:

- ***Встроенность*** на уровне драйверов. Это свойство обеспечивает быстродействие.
- ***Открытость*** - обуславливается легкостью динамической загрузки-выгрузки, мультиплексируемостью протоколов.
- ***Наличие RPC***, именованных конвейеров и почтовых ящиков для поддержки распределенных приложений.
- ***Наличие дополнительных сетевых средств***, позволяющих строить сети в масштабах корпорации: дополнительные средства безопасности централизованное администрирование отказоустойчивость (UPS, зеркальные диски).

Открытая архитектура сетевых средств Windows NT обеспечивает работу своих рабочих станций (и серверов) в гетерогенных сетях не только путем предоставления возможности динамически загружать и выгружать сетевые средства, но и путем непосредственного переключения с программных сетевых средств, ориентированных на взаимодействие с одним типом сетей, на программные средства для другого типа сетей в ходе работы системы.

5.4 ОС Windows NT как ОСРВ

1 Возможности использования NT как ОСРВ

Аргументы Microsoft "за" использование Windows NT в качестве ОСРВ.

- ***Многозадачность системы***.
- ***Поддержка многопроцессорности***.
- ***Preemption задач.***
- ***Preemption прерываний*** и возможность их маскирования. Распределение прерываний между процессорами в многопроцессорной системе. Для уменьшения времени пребывания в процедуре обработки прерывания (Interrupt Service Routine, ISR) обработчик вызывает специальную процедуру (Deffered Procedure Call, DPC) и заканчивает работу. Основная

деятельность по обработке прерывания выполняется DPC. Процедуре DPC назначается приоритет и она начинает выполнятся, управляемая планировщиком.

- ***Асинхронный ввод/вывод.***
- ***Прямой доступ к оборудованию посредством интерфейса HAL*** (Hardware Abstraction Level). HAL обеспечивает изоляцию приложения от деталей реализации оборудования, обеспечивая платформенно-независимый прямой доступ к оборудованию.
- ***Специальная схема приоритетов***. Все приоритеты разделены на две группы: 1. Класс динамических приоритетов (16): приоритеты от 0 до 15 (0 . самый низкий). Эти приоритеты динамически меняются планировщиком по алгоритму, близкому к принятому в UNIX системах.
2. Класс приоритетов реального времени (7): приоритеты 16, 22.26, 31. Эти приоритеты фиксированы, задачи из этого класса планируются на основе приоритетной очереди и получают управление раньше задач с динамическими приоритетами.
- Пространство ввода-вывода для задач из класса реального времени не участвует в страничном обмене механизма виртуальной памяти.
- Для закрепления страниц задачи в памяти существует специальный системный вызов (VirtualLock()).
- Предоставляются объекты синхронизации: критические секции, таймеры, события, mutex...

Аргументы "против"использования Windows NT в качестве ОСРВ.

- ***Ядро системы не preemptive***.
- ***Недостатки механизма DPC***:

1. Все процедуры DPC, вызываемые обработчиком прерываний, получают один и тот же приоритет (из-за малого количества приоритетов) и обрабатываются планировщиком в порядке поступления (FIFO). Тем самым низкоприоритетные прерывания будут обрабатываться ранее высокоприоритетны, но поступивших позднее.
2. Система не дает возможности узнать, сколько DPC стоит в очереди, поэтому нельзя узнать, когда начнет обрабатываться прерывание. Таким образом, существует случайная задержка между приходом прерывания и началом его обработки. Отметим, что для таких быстродействующих устройств, как дисковые и сетевые контроллеры, задержка в несколько миллисекунд может оказаться критической.
3. Для каждого прерывания только один экземпляр DPC может быть в очереди. Следовательно, процедура DPC должна уметь обрабатывать повторяющиеся прерывания, а оборудование должно уметь буферизировать прерывания во избежание потери данных. Это удорожает как драйверы устройств, так и оборудование. Отметим, что, несмотря на недостатки DPC, Windows NT вынуждена выносить обработку прерываний из ISR в DPC, поскольку во время ISR все прерывания запрещены и их можно потерять.

- ***Малое количество приоритетов в классе реального времени*** (7) приводит к тому, что много задач будут иметь одинаковый приоритет и планироваться алгоритмом типа round robin. Следовательно, время до начала исполнения задачи будет случайной величиной (зависит от текущей загруженности системы).
- ***Не решена проблема инверсии приоритетов***. Вместо традиционного для ОСРВ механизма наследования приоритетов Windows NT назначает задаче, из-за низкоприоритетности простаивающей в течение некоторого периода времени, случайный уровень приоритета, позволяющий ей начать работу. Это непредсказуемо и неприемлемо для ОСРВ. Поскольку приоритет задач класса реального времени не меняется, то этот механизм действует только на задачи из динамического класса. Тем самым ситуация только ухудшается: приоритет низкоприоритетных задач реального времени не меняется, а высокоприоритетные задачи могут быть вытеснены совсем низкоприоритетными задачами из динамического класса.
- ***Высокоприоритетные задачи могут блокироваться низкоприоритетными***. Дело в том, что приоритеты задач из класса реального времени все-таки могут менятся. Некоторые

компоненты ядра работают на уровне приоритета динамического класса. Следовательно, некоторые системные вызовы приводят к понижению приоритета и выводу задачи из класса реального времени. При этом она может быть заблокирована другой задачей, имевшей до этого более низкий приоритет.

- ***Все страницы неактивного процесса*** (например, ожидающего данных), ***могут быть перенесены на диск***, несмотря на закрепление их вызовом VirtualLock(). Это приводит к случайным задержкам при активизации процесса (например, при поступлении ожидавшихся данных).
- Для Windows NT официально ***не приводятся времена системных вызовов*** и времена блокирования прерываний.
- ***Систему невозможно использовать без дисплея и клавиатуры***.
- ***Система предъявляет слишком большие для ОСРВ запросы на память***.

Для устранения этих недостатков ряд компаний предлагает программные (аппаратные) средства.

2 Расширения реального времени для ОС Windows NT

За последнее время сразу несколько фирм объявили о создании расширений реального времени для Windows NT. Этот означает, что подобные продукты были востребованы, что и подтверждает динамика их рыночного развития. В самом деле, появление в свое время UNIX'ов реального времени означало ни что иное, как попытку применить господствующую программную технологию для создания приложений реального времени. Появление расширений реального времени для Windows NT имеет те же корни, ту же мотивацию. Огромный набор прикладных программ под Windows, мощный программный интерфейс WIN32, большое количество специалистов, знающих эту систему. Конечно, соблазнительно получить в системе реального времени все эти возможности. Так как эти продукты - новые и вызывают много вопросов, остановимся на них чуть подробнее.

Подробнее о расширениях реального времени для Windows NT

Несмотря на то, что Windows NT создавалась как сетевая операционная система, и сочетание слов "Windows NT" и "реальное время" многими воспринимается как нонсенс.

В нее при создании были заложены элементы реального времени, а именно - двухуровневая система обработки прерываний (ISR и DPC), классы реального времени (процессы с приоритетами 15-31 планируются в соответствии с правилами реального времени). Может быть, причина появления этих элементов кроется в том, что у разработчиков Windows NT за плечами есть опыт создания классической для своего времени операционной системы реального времени RSX11M (для компьютеров фирмы DEC). Конечно, даже поверхностный анализ Windows NT показывает, что эта система не годится для построения систем жесткого реального времени (система непредсказуема - время выполнения системных вызовов и время реакции на прерывания сильно зависит от загрузки системы; система велика; нет механизмов защиты от зависаний и пр. и пр.). Поэтому даже в системах мягкого реального времени Windows NT может быть использована только при выполнении целого ряда рекомендаций и ограничений.

Разработчики расширений пошли двумя путями:

1. Использовали ядра классических операционных систем реального времени в качестве дополнения к ядру Windows NT ("два в одном флаконе"). Таковы решения фирм "LP Eleknroniks" и "Radisys". В первом случае параллельно с Windows NT (на одном компьютере!) работает операционная система VxWorks, во-втором случае - InTime. Кроме того, предоставляется набор функций для связи приложений реального времени и приложений Windows NT.

Вот как, например это выглядит у LP Elektroniks: вначале стандартным образом загружается Windows NT, затем с помощью специального загрузчика загружается операционная система VxWorks, распределяя под себя необходимую память Windows (что в дальнейшем позволяет избежать конфликтов памяти между двумя ОС). После этого полной "хозяйкой" на компьютере уже становится VxWorks, отдавая процессор ядру Windows NT только в случаях,

когда в нем нет надобности для приложений VxWorks. В качестве канала для синхронизации и обмена данными между Windows NT и VxWorks служат псевдодрайверы TCP/IP в обеих системах. Технология использования двух систем на одном компьютере понятна - работу с объектом выполняет приложение реального времени, передавая затем результаты приложениям Windows NT для обработки, передачи в сеть, архивирования и пр.

2. Вариант расширений реального времени фирмы VenturCom выглядит иначе: здесь сделана попытка "интегрировать" реальное время в Windows NT путем исследования причин задержек и зависаний и устранения этих причин с помощью подсистемы реального времени. Решения фирмы "VenturCom" (RTX 4.2) базируются на модификациях уровня аппаратных абстракций Windows NT (HAL - Hardware Abstraction Layer) - программного слоя, через который драйверы взаимодействуют с аппаратурой. Модифицированный HAL и дополнительные функции (RTAPI) отвечают также за стабильность и надежность системы, обеспечивая отслеживание краха Windows NT, зависания приложений или блокировку прерываний. В состав RTX входит также подсистема реального времени RTSS, с помощью которой Windows NT расширяется дополнительным набором объектов (аналогичным стандартным, но с атрибутами реального времени). Среди новых объектов - нити (потоки, процессы) реального времени, которые управляются специальным планировщиком реального времени (256 фиксированных приоритетов, алгоритм - приоритетный с вытеснением). Побочным результатом RTX является возможность простого создания программ управления устройствами, т.к. среди функций RTAPI есть и функции работы с портами ввода-вывода и физической памятью. Предложения VenturCom характерны еще и тем, что они предоставляют совершенно экзотическую для NT возможность, а именно - возможность конфигурирования Windows NT и создания встроенных конфигураций (без дисков, клавиатуры и монитора) (Интегратор Компонентов - CI).

Несмотря на всю неоднозначность (а точнее - однозначность!) отношения традиционных пользователей систем реального времени ко всему, что связано с "Microsoft", необходимо констатировать факт: появился новый класс операционных систем реального времени - а именно расширения реального времени для Windows NT. Результаты независимых тестирований этих продуктов показывают, что они могут быть использованы для построения систем жесткого реального времени. Область применения расширений реального времени - большие системы реального времени, где требуется визуализация, работа с базами данных, доступ в интернет и пр.

1.1.1.1 Hyperkernel

Система Hyperkernel выпускается фирмой Nematron. Представляет собой ядро, обеспечивающее детерминистичное планирование и работающее на уровне привилегий 0 процессора Intel 80x86 вместе с Windows NT. Задачи Hyperkernel не видны Windows NT. Для них определены 8 уровней приоритетов с preemptive планированием. В качестве средства разработки используются стандартные для Windows NT компиляторы Visual C/C++ и специальные библиотеки. Используется API Win32 и стандартный HAL. Разрешение таймера: 1 микросекунда, минимальный квант времени 20 микросекунд. Время задержки реакции на прерывание: 5 микросекунд, переключение контекста - 4 микросекунды на Intel Pentium 133Mhz.

LP RT-Technology

Система LP RT-Technology выпускается фирмой LP Elektronik GmbH и включает три компоненты.

1. Плату для шины ISA, обеспечивающую 7 дополнительных уровней прерываний. Для взаимодействия с остальной системой плата использует NMI немаскируемое прерывание процессора Intel 80x86.

2. LP-RTWin Toolkit комплект разработчика ISR, используемый совместно с Visual C/C++ и отладчиком SoftICE фирмы NuMega.
3. LP-VxWin RTAcc программный комплекс, обеспечивающий сосуществование Windows NT и VxWorks на одном PC. Две операционные системы взаимодействуют посредством протокола TCP/IP через разделяемую память. В качестве средства разработки используется Tornado - комплект разработчика для VxWorks.

Realtime ETS Kernel

Система Realtime ETS Kernel выпускается фирмой Phar Lap SoftWare в двух вариантах.

1. TNT Embedded ToolSuite, Realtime Edition, включающий: Realtime ETS Kernel, ETS TCP/IP, отладчик CodeView с поддержкой Borland Turbo Debugger ассемблер 386ASM, linker, поддержку компиляторов Visual C/C++, Borland C/C++, Watcom C/C++ и API Win32.
2. Realtime ETS Kernel полная замена Windows NT, включает: компактное ядро (28Kb), поддерживающее Win32 API и использующее стандартные библиотеки компиляторов; ядро имеет 32 уровня приоритетов и может быть записано в ПЗУ.

Component Integrator

Система Component Integrator выпускается фирмой VenturCom, Inc. В отличие от предыдущих систем, здесь не вводится новое ядро, а предлагаются пакеты, расширяющие ряд узких мест Windows NT.

1. Embedded Component Kit (ECK), включающий: поддержку работы без дисплея и клавиатуры, уменьшение потребности в памяти, отключение страничного обмена, поддержку флеш-памяти (возможность загрузки с флеш-памяти объемом 10Mb) и шины VME.
2. Real-time Extension (RTX), включающий: поддержку таймеров (разрешение 1 микросекунда, минимальный квант времени 100 микросекунд), отключение виртуальной памяти, работа с физической памятью (т.е. тождественное отображение виртуальной памяти на физическую), 128 новых уровней привилегий. Время задержки реакции на прерывание: 5.20 микросекунд.

Willows RT

Система Willows RT выпускается фирмой Willows SoftWare, Inc. В отличие от предыдущих систем, здесь приложение вообще не будет запускаться в Window NT. Windows NT используется только как платформа разработки, приложение же будет работать под управлением настоящей ОСРВ (QNX, VxWorks, ...). Система состоит из трех частей.

1. Библиотеки TWIN32 и TWIN16, заменяющие библиотеки Microsoft и реализующие API.
2. TWIN Platform Abstraction Layer, ядро, обеспечивающее системные вызовы Microsoft.
3. TWIN Binary Interface, обеспечивающий трансляцию вызовов Platform Abstraction Layer в API используемой ОСРВ.

5.5 ОС QNX

1 Основные характеристики ОС QNX

Система QNX выпускается фирмой QNX SoftWare Systems (USA). Основные характеристики:

1. Классическая система тип: self-hosted
2. Архитектура: на основе микроядра

3. Стандарт: POSIX 1003
4. Свойства как ОСРВ:
   - многозадачность: POSIX 1003;
   - многопроцессорность;
   - 32 уровня приоритетов;
   - планирование: FIFO, round robin, адаптивное;
   - preemptive ядро.
5. ОС разработки (host): UNIX/Windows.
6. Процессоры (target): Intel 80x86
7. Линии связи host-target: Ethernet, Arcnet, Serial, Token Ring.
8. Минимальный размер: 60Kb
9. Средства синхронизации и взаимодействия: POSIX 1003 (семафоры, mutex…)
10. Средства разработки:
    - комплекты разработчика, включающие компилятор C/C++, отладчик, анализатор от QNX и независимых поставщиков (например, Watcom/SyBase);
    - X Windows/Motif для QNX.

Операционная система QNX идеальна для приложений работающих в масштабе жесткого реальном времени. Она обеспечивает все неотъемлемые компоненты системы в реальном масштабе времени. Это:

- многозадачный режим,
- приоритетное планирование с приоритетным управлением,
- быстрое контекстное переключение

Прежде всего, QNX является реально мультизадачной ОС, с гибкой системой приоритетов и дисциплин диспетчеризации, благодаря которой имеется полная гарантия того, что процесс с наивысшим приоритетом начнет выполняться практически немедленно. Все программы в системе защищены друг от друга, то есть любой сбой одной из программ не приводит к сбою всей системы. Процессы свободно обмениваются между собой, непосредственно или через распределенные ресурсы, но в любом случае независимо от того, запущены они на одном и том же или на разных компьютерах. Файловая система QNX приспособлена к сложным условиям реального производства. Внезапное отключение компьютера не приводит к ее порче, после включения будет обеспечена нормальная работа системы. Сетевая технология OS QNX под названием FLEET (таблица 1), помимо прочего, дает возможность работать через несколько (до трех) сетевых каналов одновременно, причем разных — Ethernet, Arcnet, Serial, Token Ring. Более того, сетевой менеджер позволяет сети QNX работать с любыми устройствами, используя их драйверы. Таким образом, без лишних хлопот в QNX реализована технология удаленных коммуникаций. Через модем можно легко подсоединиться в качестве терминала или узла сети.

Строго говоря, узлы сети QNX по существу действуют как мощный единый компьютер. Ему доступны все устройства и ресурсы, если только не установлены программные ограничения. Любые диски, принтеры, устройства сбора данных и прочее могут быть добавлены к системе простым подключением к любой машине в сети. Последнее обстоятельство особенно привлекательно для создания именно прикладных систем с распределенными ресурсами.

Таблица 1. Сетевая технология FLEET

Параметр	Термин	Свойства
F	Fault-tolerant	Если одна физическая сеть вышла из строя, QNX перейдет на использование других автоматически
L	Load-balancing	При загруженности одного физического канала будет выбираться свободный
E	Efficient	Сетевые драйверы QNX полностью используют все аппаратные возможности сетевого оборудования для увеличения пропускной способности

E	Extensible	Для поддержки новых сетей требуются только новые драйверы
T	Transparent	Так как нет разницы между локальным и сетевым взаимодействием процессов, любые приложения могут работать в сети без модификаций

QNX замечательно гибкая система. Разработчики могут легко настраивать операционную систему, чтобы выполнять потребности своих приложений. От "скелетной" конфигурации ядра с несколькими маленькими модулями к обширной сетевой системе, призванной обслуживать сотни пользователей, QNX позволяет устанавливать систему для использования только тех ресурсов, которые требуются для работы.

QNX достигла этого уникального звания эффективности, модульности и простоты через два фундаментальных принципа:

- Микроядерная архитектура;
- Основанные на сообщениях межпроцессорные связи.

2 Архитектура Микроядра QNX

QNX состоит из маленького ядра, ответственного за группу взаимодействующих процессов. Как показывает следующая иллюстрация, структура больше похожа на группу, чем на иерархию — как несколько игроков равного ранга взаимодействуют друг с другом и со своей "четвертью" ядра.

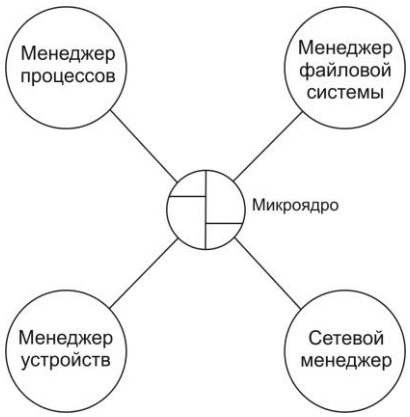

Рисунок 3. Микроядро QNX, координирующееся системными менеджерами

Истинное ядро

Ядро — основа любой операционной системы. В большинстве систем "ядро" включает в себя так много функций, что на самом деле оно и является полной операционной системой.

Но Микроядро QNX — это действительно ядро. Прежде всего, подобно ядру операционной системы реального времени, Микроядро QNX очень мало. Во-вторых, ему посвящено только две существенные функции:

- Передача сообщений — Микроядро управляет маршрутизацией всех сообщений между всеми процессами всей системы;
- Планирование — это часть Микроядра, которая вызывается всякий раз, когда процесс изменяет состояние как результат сообщения или прерывания.

В отличие от процессов, само Микроядро никогда не выполняет процесс. Оно введено только как прямой результат запросов ядра или от процесса или от аппаратного прерывания.

Системные процессы

Все сервисы QNX, кроме обеспеченных Микроядром, обрабатываются через стандарт QNX процессов. Типичная конфигурация QNX имеет следующие системные процессы:

- Менеджер процессов (Proc)
- Менеджер файловой системы (Fsys)
- Менеджер устройств (Dev)
- Сетевой менеджер (Net)

Системные процессы фактически не отличаются от любой пользовательской программы — они не имеют никаких частных или скрытых интерфейсов, которые являются недоступными процессам пользователя.

Именно эта архитектура дает QNX беспрецедентную расширяемость. Так как большинство сервисов ОС обеспечивается по стандартным процессам QNX, то чтобы непосредственно увеличить ОС, надо только записывать новые программы, чтобы обеспечить новые сервисы.

Фактически, граница между операционной системой и приложением может стать очень размытой. Единственное реальное различие между системными сервисами и приложениями — то, что сервисы ОС управляют ресурсами для клиентов.

Давайте предположим, что написан сервер базы данных. Как такой процесс должен быть классифицирован?

Как только файловая система принимает запрос (сообщения в QNX) на открытие файлов и чтение или запись данных, задействуется сервер базы данных. В то время как запросы на сервер базы данных могут быть более сложными, оба сервера предоставляют почти те же самые наборы примитивов (в соответствии с сообщениями), которые в свою очередь обеспечивают доступ к ресурсу. И независимые процессы, которые могут быть написаны конечным пользователем и начаты на необходимой основе.

Сервер базы данных можно рассматривать как системным процессом при одной инсталляции и приложением при другой. Это действительно не имеет значения. Важно то, что QNX позволяет таким процессам быть осуществленными, без модификации типовых элементов операционной системы.

Драйверы устройств

Драйверы устройств — процессы, которые ограждают операционную систему от воздействия со всеми деталями, требуемыми для поддержки определенных аппаратных средств.

Поскольку драйвера запускаются как стандартные процессы, добавление нового драйвера к QNX не затрагивает любую другую часть операционной системы. Единственное изменение, которое должен делать пользователь в среде QNX, это фактически запустить новый драйвер.

Как только они завершили свою инициализацию, драйверы могут делать следующее:

- Выбрать стандартный процесс, становясь расширением к тому, с которым они связаны;
- Сохранение их индивидуальных идентификаций как стандартных процессов.

Межпроцессорная связь (IPC)

Когда несколько процессов, выполняются одновременно, как в типичных многозадачных средах реального масштаба времени, операционная система должна обеспечить механизмы, позволяющие процессам связываться друг с другом.

IPC — ключ к проектированию приложений, представляющий набор "сотрудничающих" процессов, в котором каждый процесс обрабатывает одну определенную часть целого.

QNX обеспечивает простой, но мощный набор возможностей IPC, которые упрощают задание "развивающихся" приложений, составленных из "сотрудничающих" процессов.

3 QNX как операционная система передачи сообщений

QNX была первой коммерческой операционной системой использующей передачу сообщений как фундаментальное средство IPC. QNX заимствует многое из ее мощности, простоты и элегантности для выполнения интегрированных методов передачи сообщений на всю полноту системы.

В QNX сообщение — пакет байтов, проходящий от одного процесса к другому. QNX не придает никакого специального значения к содержанию сообщения — данные в сообщении имеют значение для отправителя сообщения и для его получателя, но ни для кого еще.

Передача сообщений не только позволяет процессам передавать данные друг другу, но также и обеспечивает средства синхронизации выполнения нескольких процессов. Когда процессы посылают, принимают и отвечают на сообщения, они подвергаются различным "изменениям состояния", которые действуют только во время их работы. Зная их состояние и приоритеты, Микроядро может распределить все процессы настолько эффективно насколько возможно максимально использовать доступные ресурсы центрального процессора.

Приложения реального времени и другие значимые приложения вообще требуют надежной формы IPC, потому что процессы, которые составляют такие приложения жестко взаимодействуют. Дисциплина, наложенная конструкцией передачи сообщений QNX приносит порядок и большую надежность приложений.

4 QNX как сеть

В самой простой форме, локальная сеть обеспечивается механизмом для совместного использования файлов и периферийных устройств среди нескольких связанных компьютеров. QNX идет далеко впереди этой простой концепции и интегрирует сеть в одиночный, гомогенный набор ресурсов.

Любой процесс на любой машине в сети может непосредственно использовать любой ресурс на любой другой машине. Не имеется никакого различия между локальным или удаленным ресурсом — никакие специальные средства не требуется встраивать в приложения, чтобы использовать отдаленные ресурсы. Пользователи могут обращаться к файлам отовсюду в сети, воспользоваться любым периферийным устройством и выполнять приложения на любой машине в сети (если они имеют соответствующие полномочия). Процессы могут связываться тем же самым способом повсюду в сети.

Модель одиночного компьютера

QNX разработан в основном как глобальная сетевая операционная система. Сеть QNX больше понимает похожие универсальные компьютеры, чем набор микрокоманд. Пользователи просто знают о большом наборе ресурсов, доступных для использования любым приложением. Но в отличие от универсальных ЭВМ, QNX обеспечивает высоко чувствительную среду, так как необходимое количество вычислительной мощности может быть доступно на каждом узле, чтобы выполнять потребности каждого пользователя.

В среде управления процессами, например, в PLC и в других устройствах ввода-вывода реального времени может требоваться больше количества ресурсов, чем в других, менее критичных, приложениях типа текстового процессора. Сеть QNX достаточно чувствительна, чтобы поддержать оба типа приложений, и в то же время QNX позволяет фокусировать вычислительную мощность на этаже предприятия, где и когда это необходимо, без того, чтобы жертвовать параллельной связностью с рабочим столом.

Гибкая организация сети

Сеть QNX может быть собрана используя различные аппаратные средства и протоколы промышленного стандарта. Так как они полностью прозрачны к прикладным программам и пользователям новые сетевые архитектуры могут быть представлены в любое время без возмущения операционной системы.

Каждому узлу в QNX сети назначен уникальный номер, который становится его идентификатором. Этот номер виден только для определения выполняется ли QNX как сетевая или как одиночная операционная система.

Эта степень проницаемости — другой пример отличительной мощности архитектуры передачи сообщений QNX. В многих системах важные функции, такие как организации сети, IPC, или четная передача сообщений сформированы на вершине ОС, интегрированы непосредственно в ее ядро. Результат — часто неуклюжий, неэффективный интерфейс "двойного стандарта".

QNX связан на принципе, что эффективная связь является ключом к эффективному действию. Передача сообщений таким образом формирует краеугольный камень архитектуры QNX и расширяет эффективность работы всех процессов внутри системы.

5.6 Тенденции развития ОСРВ

Количество операционных систем реального времени, несмотря на их специфику, очень велико. В последнем обзоре "Real-Time Magazine" было упомянуто около шестидесяти систем. Наверное, этих систем еще больше, если иметь в виду некоммерческие операционные системы реального времени. Однако сама специфика применения операционных систем реального времени

требует гарантий надежности, причем гарантий, в том числе и юридических - этим, видимо, можно объяснить тот факт, что среди некоммерческих систем реального времени нет сколько-нибудь популярных.

Среди коммерческих систем реального времени можно выделить группу ведущих систем - по объемам продаж и по популярности. Эти системы: VxWorks, OS9, pSOS, LynxOS, QNX, VRTX.

После того, как мы попытались навести порядок в мире операционных систем реального времени, предложив классификацию систем, придется этот порядок разрушить, рассказывая о том, как развивается мир систем реального времени. Один из видимых процессов - сближение операционных систем реального времени различных классов. Так во многих операционных систем реального времени класса "ядра реального времени" и "UNIX'ы реального времени появились в последнее время кроссовые системы разработки высокого качества, что раньше было характерно для операционных систем реального времени класса "Исполнительные системы реального времени". И это - общая тенденция. Резидентные средства разработки для операционных систем реального времени, поддерживающих многие целевые архитектуры, уже выглядят архаизмом. Так, например, мощные кроссовые системы разработки появились в таких операционных системах реального времени, как OS9 ("ядра реального времени") и "Lynx OS" (UNIX'ы реального времени). Что касается систем исполнения операционных систем реального времени, здесь наблюдается такая же картина: в системах класса "UNIX РВ" и "ядра РВ" появляются новые компактные варианты систем исполнения с маленьким временем переключения контекста (качество систем класса "исполнительные операционные системы реального времени"). Примеры: OS9, QNX, LynxOS. Еще одна тенденция, которую нельзя не заметить - это появление таких продуктов как Real-Time JAVA и Embedded JAVA во многих ОСРВ. Сейчас JAVA - один из обязательных атрибутов систем реального времени. Отмечу также интересную тенденцию, возникшую в последнее время, и связанную с тем, что в ряде операционных систем реального времени (QNX, LynxOS) появились дополнительные библиотеки, реализующие подмножества программного интерфейса WIN32. Аналогичные процессы происходили в недавнее время со стандартом POSIX 1003.1 (базовый программный интерфейс UNIX'а). В итоге многие операционных систем реального времени стали POSIX-совместимыми. Видимо в недалеком будущем многие операционные системы реального времени станут еще и WIN32-совместимыми. Говоря о тенденциях развития операционных систем реального времени, нельзя обойти вниманием вопрос, о котором сейчас много говорят и пишут - а именно вопрос об операционной системе Windows CE, о позиционировании этой системы по отношению к операционным системам реального времени. Версии Windows CE, появившиеся к настоящему моменту, никак не назовешь системами реального времени (версии 2.0 и 2.1). Эти продукты говорят скорее о поисках, в которых находятся специалисты "Microsoft", чем о какой-то магистральной линии развития. Так, программные интерфесы для драйверов в версиях 2.1 и 2.0 несовместимы друг с другом, причем в версии 2.1 появились существенные недостатки, следствие которых - большие задержки при операциях ввода-вывода. Появление версии 3.0, про которую уже известно, что она будет существенно отличаться от предыдущих, прояснит ситуацию и позволит понять, как будет развиваться Windows CE дальше. Несомненно одно: "Microsoft" открыл проект "CE" с целью выйти на рынок встроенных применений и, наверное, достигнет многого на этом рынке. Однако встроенные системы и системы реального времени - не одно и то же. Различные интеллектуальные калькуляторы, электронные записные книжки, переводчики, органайзеры, и пр. и пр. представляют из себя огромный рынок сбыта - именно там лежит область интересов "Microsoft" и требования именно этого рынка будут определять технические решения в области CE, а никак не требования жесткого реального времени. Создать же универсальную систему, удовлетворяющую противоречивым требованиям невозможно. Маловероятно поэтому, что Windows CE составит серьезную конкуренцию системам, о которых говорилось.