Оглавление

[1. Общее описание операционных систем реального времени 3](#_Toc252112632)

[1.1 Что такое система реального времени 4](#_Toc252112633)

[1.2 Основные требования к СРВ 5](#_Toc252112634)

[1.3 Общие характеристики СРВ 5](#_Toc252112635)

[1.4 Способы использования ОС 5](#_Toc252112636)

[1.5 Требования, предъявляемые ОС при проектировании ОСРВ 6](#_Toc252112637)

[1.5.1 Требование 1. ОС должна быть многонитевой (multi-threaded) и прерываемой 6](#_Toc252112638)

[1.5.2 Требование 2. Должно существовать понятие приоритета нити 6](#_Toc252112639)

[1.5.3 Требование 3. ОС должна обеспечивать предсказуемые механизмы синхронизации задач 6](#_Toc252112640)

[1.5.4 Требование 4. Должна существовать система наследования приоритетов 6](#_Toc252112641)

[1.5.5 Требование 5. Поведение ОС должно быть известно 6](#_Toc252112642)

[2. Обзор операционных систем реального времени 7](#_Toc252112643)

[2.1 QNX 7](#_Toc252112644)

[2.1.1 Профессиональный пакет 8](#_Toc252112645)

[2.1.2 Рабочая станция 9](#_Toc252112646)

[2.2 VxWorks/Tornado 10](#_Toc252112647)

[2.2.1 Базовые сетевые средства VxWorks: UNIX-networking, SNMP и STREAMS. 11](#_Toc252112648)

[2.2.2 Мониторинг и отладка в реальном масштабе времени: WindView. 11](#_Toc252112649)

[2.3 RTLinux 12](#_Toc252112650)

[2.3.1 Основные сложности при реализации систем реального времени в среде LINUX 12](#_Toc252112651)

[2.3.2 Организация RTLinux 13](#_Toc252112652)

[2.4 Контроль и управление в реальном времени с использованием OS9 14](#_Toc252112653)

[2.4.1 Введение 14](#_Toc252112654)

[2.4.2 Как развивалась эта прикладная система? 15](#_Toc252112655)

[2.4.3 Утилиты WINOTOOLS: 15](#_Toc252112656)

[2.5 Windows NT 16](#_Toc252112657)

[2.5.1 Возможность использования Windows NT в качестве ОС реального времени 16](#_Toc252112658)

[2.5.2 RTX - real-time extension для Windows NT от компании VenturCom 16](#_Toc252112659)

[2.5.3 Windows NT 4.0 как ОСРВ. Общие требования 17](#_Toc252112660)

[2.5.4 Расширения реального времени для Windows NT. Расширение функциональности 17](#_Toc252112661)

[2.5.4.1 Подсистема реального времени RTSS 17](#_Toc252112662)

[2.5.4.2 HAL реального времени 18](#_Toc252112663)

[2.5.4.3 Программный интерфейс реального времени RTAPI 18](#_Toc252112664)

[3. Литература 19](#_Toc252112665)

**1. Общее описание операционных систем реального времени**

Основой любого аппаратно-программного комплекса, в том числе работающего в режиме реального времени, является операционная система (ОС). Операционной системой называют комплекс программ, обеспечивающий управление ресурсами аппаратно-программного комплекса (вычислительной системы) и процессами, использующими эти ресурсы при вычислениях. Ресурсом в данном контексте является любой логический или физический (и в совокупности) компонент вычислительной системы или аппаратно-программного комплекса и предоставляемые им возможности.

Основными ресурсами являются процессор (процессорное время), оперативная память и периферийные устройства.

Управление ресурсами сводится к выполнению следующих задач: упрощение доступа к ресурсам, распределение их между процессами.

Решение первой задачи позволяет "спрятать" аппаратные особенности вычислительной системы, и тем самым предоставить в распоряжение пользователю или программисту виртуальную машину с существенно облегченным управлением. Таким образом, ОС поддерживает следующие интерфейсы: пользовательский (командный язык для управления функционированием системы и набор сервисных услуг); программный (набор услуг, освобождающий программиста от кодирования рутинных операций).Функция распределения ресурсов является одной из наиболее важных задач, решаемых ОС, однако она присуща не всем ОС, а только тем, которые обеспечивают одновременное выполнение нескольких программ (процессов).

Процессом называется последовательность действий, предписанных программой или ее логически законченной частью, а также данные, используемые при вычислениях. Процесс является минимальной единицей работы, для которой выделяются ресурсы.

В настоящее время существует большое разнообразие ОС, которые классифицируются по следующим признакам:

* количество пользователей, одновременно обслуживаемых системой;
* число процессов, которые могут одновременно выполняться под управлением ОС;
* тип доступа пользователя к системе;
* тип аппаратно-программного комплекса.

В соответствии с первым признаком различаются одно- и многопользовательские ОС.

Второй признак делит ОС на одно- и многозадачные.

В соответствии с третьим признаком ОС делятся на:

* системы с пакетной обработкой. В этом случае из программ, подлежащих выполнению, формируется пакет, который предъявляется системе для обработки. В этом случае пользователи непосредственно с ОС не взаимодействуют;
* системы разделения времени, обеспечивающие одновременный интерактивный доступ к вычислительной системе нескольких пользователей через терминалы. При этом ресурсы системы выделяются каждому пользователю "по очереди", в соответствии с той или иной дисциплиной обслуживания;
* системы реального времени, которые должны обеспечивать гарантированное время ответа на внешние события (более подробно см. ниже).

Четвертый признак делит ОС на одно- и многопроцессорные, сетевые и распределенные. Для многопользовательских и многозадачных ОС важным показателем является дисциплина обслуживания. В соответствии с этим различают вытесняющий и согласующий режимы многозадачной работы. При вытесняющей организации выделением задачам процессорного времени занимается только ОС (например, для каждой задачи процессор выделяется по очереди, причем на строго фиксированный промежуток времени, но возможно и приоритетное обслуживание). В случае согласующей организации каждая задача, получив управление, сама определяет, когда ей "отдать" процессор другой задаче. В общем случае согласование эффективнее и надежнее вытеснения, но определяющим фактором при реализации программ становится тот факт, что данная программа не должна монопольно использовать процессорное время.

**Система реального времени (СРВ)** - это система, правильность функционирования которой зависит не только от логической корректности вычислений, но и от времени, за которое эти вычисления производятся.

Для событий, происходящих в такой системе, важно время, когда эти события происходят, и их логическая корректность. Система работает в реальном времени, если ее быстродействие адекватно скорости протекания физических процессов на объектах контроля или управления (имеются в виду процессы, непосредственно связанные с функциями, выполняемыми конкретной системой реального времени). Система управления должна собрать данные, произвести их обработку по заданным алгоритмам и выдать управляющее воздействие за такой промежуток времени, который обеспечивает успешное выполнение поставленных задач.

# 1.1 Что такое система реального времени

В последнее время все чаще приходится сталкиваться с задачами, требующими управления сложными процессами или оборудованием при помощи ЭВМ. При этом все события в этих процессах происходят тогда, когда они происходят. Компьютер же может выполнять лишь конечное число операций в конечное время, поэтому возникает вопрос: а успеет ли компьютер с нужной скоростью обсчитать ситуацию и выдать конкретные управляющие действия, которые были бы адекватны именно в определенный момент времени. На мой взгляд, проблемы подобного рода возникли из-за использования очень больших скоростей в современном производстве. Ясно, что сигналы в природе распространяются с конечной скоростью, скорость работы тоже конечна, поэтому мгновенных действий (вызванных неким событием) от компьютера ожидать принципиально невозможно. Ведь каким бы современным (читай - мощным по производительности, т.е. высокой скоростью обработки команд и операций) компьютер бы ни был - ему физически нужны хотя бы доли секунды, чтобы выполнить небольшую простую группу команд, а иногда этого времени слишком много. Таким образом, время реакции системы на некоторое событие строго больше нуля. Реальные задачи допускают некоторого запаздывания действий, и если система имеет время реакции меньше, чем эта допустимая задержка, то ее справедливо называть системой реального времени. Так как в природе разные процессы протекают с разной скоростью, одна и таже система может укладываться в заданные рамки для одного процесса и не укладываться для другого. Таким образом, о системе реального времени имеет смысл говорить применительно к конкретной задаче. Например, чтобы построить зависимость средней температуры воздуха за день от дня недели в качестве системы реального времени сойдет практически любой компьютер с практически любым ПО. Если же мы управляем посадкой самолета, где существенную роль играют миллисекунды, было бы более правильно внимательно выбирать аппаратное и программное обеспечение.

Кроме рассмотренной задачи реагирования на некоторое событие, существуют еще другие классы задач реального времени. Одной из часто встречаемых является задача постоянного наблюдения или управления динамическим процессом, т.е. когда требуется непрерывно обмениваться сигналами с внешним миром. Компьютер - дискретная система, поэтому приходится осуществлять некоторые действия с некоторыми конечными промежутками времени, считая, что в эти малые промежутки времени внешний мир остается неизменным. Если наша система способна обрабатывать информацию и выдавать управляющие сигналы с требуемой частотой, то ее можно назвать системой реального времени. Нетрудно понять, что эту задачу легко свести к предыдущей, используя в качестве события начало очередного интервала времени. Время реакции должно быть меньше времени дискретизации процесса. Таким образом, описанная ранее задача является наиболее важной, когда речь идет о системах реального времени. Следует отметить, что неудовлетворительная по запаздыванию работа системы в некоторых задачах может привести к фатальным последствиям, а в некоторых не произойдет никаких внештатных и нежелательных ситуаций. Например: если система измерения температуры из описанного выше примера случайно опоздает на непозволительное время, то это значит, что мы просто изменили выборку точек съема температуры, и все равно получим правильный результат, если же на секунду случайно задержится автомат захода на посадку в пассажирском самолете при внезапном порыве ветра, самолет может не попасть на полосу и десятки людей погибнут. Таким образом, следует делить системы на системы жесткого и мягкого реального времени.

**Системой жесткого реального времени** называется система, где неспособность обеспечить реакцию на какие-либо события в заданное время является отказом и ведет к невозможности решения поставленной задачи. Время реакции в системах жесткого реального времени должно быть минимальным. Большинство систем жесткого реального времени являются системами контроля и управления. Такие СРВ сложны в реализации, так как к ним предъявляются особые требования в вопросах безопасности.

Точного определения **мягкого реального времени** не существует, поэтому можно отнести сюда все СРВ, не подпадающие под категорию жестких. Так, система мягкого реального времени может не успевать все делать в заданное время, поэтому возникает проблема определения критериев успешности (нормальности) ее функционирования.

Кроме того, СРВ можно разделить на системы специализированные и универсальные.

* **Специализированная СРВ** - система, где конкретные временные требования изначально определены. Такая система должна быть специально спроектирована для удовлетворения этих требований.
* **Универсальная СРВ** должна уметь выполнять произвольные (заранее неопределенные) временные задачи без применения специальной техники. Разработка таких систем является самой сложной задачей, хотя обычно, требования, предъявляемые к таким системам, мягче, чем требования к специализированным системам.

# 1.2 Основные требования к СРВ

* возможность параллельного выполнения нескольких задач;
* предсказуемость;
* важно максимальное (не среднее) время отклика на событие;
* особые требования в вопросах безопасности;
* возможность безотказной работы в течение длительного времени.

# 1.3 Общие характеристики СРВ

* большие и сложные системы;
* распределенные системы;
* жесткое взаимодействие с аппаратурой;
* выполнение задач зависит от времени;
* сложность тестирования.

СРВ должны реагировать на различные типы внутренних и внешних событий (периодических и непериодических). Необходимо отметить, что принадлежность системы к классу СРВ никак не связана с ее быстродействием. Исходные требования к времени реакции системы и другим временным параметрам определяются или техническим заданием на систему, или просто логикой ее функционирования. Интуитивно понятно, что быстродействие СРВ должно быть тем больше, чем больше скорость протекания процессов на объекте контроля и управления.

# 1.4 Способы использования ОС

Классическая ОС. Win NT, Linux и Unix.

Классическая ОС с расширением РВ (Win NT - RTX, RT Linux, RT Unix)

Коммерческая ОС. Примером могут служить такие системы как VxWorks, OS9 и т.п. Необходимо отметить, что такие системы очень дорогие. Например, стоимость полного пакета ОС VxWorks (Tornado 1.0) в 2002 году составляла около 15 000 долларов США. Впрочем, с годами такая система значительно дешевеет - сегодня ее стоимость составляет около 10 000 долларов (Tornado версии 2.0 и выше - полная стоимость зависит от выбираемых компонентов).

Для более детального рассмотрения возможностей ОСРВ представлены ориентировочные цифры, дающее представление о порядке времени реакции и подходящих операционных системах. Данная таблица сформирована на основании экспериментальных данных, полученных на базе вычислительных комплексов, построенных на основе процессоров Intel 80486DX. Безусловно, данный процессор на сегодняшний день является устаревшим, но можно сделать выводы об уровне реакции на внешние события различных систем

|  |  |
| --- | --- |
| РВ.Время реакции | Использованные ОС |
| Менее 10 мкс | Только ОСРВ, но даже они могут быть бессильны |
| 10 - 100 мкс | Операционные системы реального времени |
| 100 мкс - 1 мс | ОСРВ, RTAI, RT- UNIX и LINUX, расширения реального времени для Windows NT, CE |
| 1 мс | Можно пытаться делать что-то с Linux и Windows NT, но не для систем, где опоздания реакции могут привести к тяжелым последствиям |

Из таблицы видно, что временные рамки ОСРВ достаточно жесткие. Среди современных операционных систем есть класс продуктов, разработанных специально для построения систем жесткого реального времени - VxWorks, OS9, QNX, LynxOS, OSE и другие. Эти системы содержат необходимый набор инструментов, и в некоторых случаях являются единственным выбором - на него приходится идти, невзирая на затраты. Однако достаточно часто требования к реальному времени (полная предсказуемость времени реакции) становятся менее жесткими, например, необходимо добиться только нужной средней производительности.

Иногда достаточно жестко контролировать только одно из событий, допуская при этом задержки реакций на остальные. В подобных случаях возможности выбора расширяются, и желаемых результатов можно достичь, используя такие широко распространенные операционные системы как LINUX, Windows NT, Windows CE, дополняя их расширениями реального времени (RTAI, RT LINUX, RTX).

# 1.5 Требования, предъявляемые ОС при проектировании ОСРВ

# 1.5.1 Требование 1. ОС должна быть многонитевой (multi-threaded) и прерываемой

Как указывалось выше, ОСРВ должна быть предсказуемой, что означает максимальное время выполнения того или иного действия, которое должно быть известно заранее и должно соответствовать требованиям приложения.

Первое требование состоит в том, что ОС должна быть многонитевой по принципу абсолютного приоритета (прерываемой). Планировщик должен иметь возможность прервать любую нить и предоставить ресурс той нити, которой он более необходим. ОС (и аппаратура) должны также обеспечивать прерывания на уровне обработки прерываний.

# 1.5.2 Требование 2. Должно существовать понятие приоритета нити

Проблема в том, чтобы определить, какой задаче требуется ресурс. В идеальной ситуации ОСРВ отдает ресурс нити или драйверу с ближайшим крайним сроком (так называемые ОС, управляемые временным ограничением (deadline driven OS)).

Чтобы реализовать это, ОС должна знать время, требуемое каждой из выполняющихся нитей для завершения (до сих пор не существует ОС, построенной по этому принципу, так как он слишком сложен для реализации), поэтому разработчики ОС принимают иную точку зрения: вводится понятие уровня приоритета задачи, и временные ограничения сводят к приоритетам. Так как умозрительные решения чреваты ошибками, показатели СРВ при этом снижаются. Чтобы более эффективно осуществить указанное преобразование ограничений, проектировщик может воспользоваться теорией расписаний или имитационным моделированием, хотя и это может оказаться бесполезным. На сегодняшний день не имеется иного решения, поэтому понятие приоритета нити необходимо.

# 1.5.3 Требование 3. ОС должна обеспечивать предсказуемые механизмы синхронизации задач

Задачи разделяют данные (ресурсы) и должны сообщаться друг с другом, следовательно, должны существовать механизмы блокирования и коммуникации.

# 1.5.4 Требование 4. Должна существовать система наследования приоритетов

На самом деле именно этот механизм синхронизации и тот факт, что различные нити используют одно и то же пространство памяти, отличают нити от процессов. Процессы не разделяют одно и то же пространство памяти. Так, например, старые версии UNIX не являлись многонитевыми. Старый UNIX - многозадачная ОС, где задачами являются процессы, которые сообщаются через потоки (pipes) и разделяемую память. Оба эти механизма используют файловую систему, а ее поведение непредсказуемо.

Комбинация приоритета нити и разделение ресурсов между ними приводит к другому явлению: классической проблеме инверсии приоритетов. Это можно проиллюстрировать примером, где есть как минимум три нити. Когда нить низшего приоритета заняла ресурс, разделяемый с нитью высшего приоритета, а сначала выполняется нить среднего приоритета, выполнение нити высшего приоритета будет приостановлено, пока не освободится ресурс и не отработает нить среднего приоритета. В этой ситуации время, необходимое для завершения нити высшего приоритета, зависит от нижних приоритетных уровней - это и есть инверсия приоритетов. Ясно, что в такой ситуации трудно выдержать ограничение на время исполнения.

Чтобы устранить такие инверсии, ОСРВ должна допускать наследование приоритета, т. е. повышение приоритета до уровня вызывающей нити. Наследование означает, что блокирующая ресурс нить наследует приоритет блокируемой нити (справедливо лишь в том случае, если блокируемая нить имеет более высокий приоритет). Иногда утверждают, что в грамотно спроектированной системе такая проблема не возникает. В случае сложных систем с этим нельзя согласиться. Единственный способ решения этой проблемы состоит в увеличении приоритета нити вручную прежде, чем ресурс окажется заблокированным - это возможно в случае, когда две нити разных приоритетов претендуют на один ресурс. В общем случае решения не существует.

# 1.5.5 Требование 5. Поведение ОС должно быть известно

Наконец, следует рассмотреть временные ограничения. Время выполнения системных вызовов и временные характеристики поведения системы в различных обстоятельствах должны быть известны разработчику, поэтому производитель ОСРВ должен приводить следующие характеристики:

* латентную задержку прерывания (т. е. время от момента прерывания до момента запуска задачи): она должна быть предсказуема и согласована с требованиями приложения. Эта величина зависит от числа одновременно "висящих" прерываний;
* максимальное время выполнения каждого системного вызова (должно быть предсказуемым и независимым от числа объектов в системе);
* максимальное время маскирования прерываний драйверами и ОС.
* системные уровни прерываний;
* уровни прерываний драйверов устройств, их временные характеристики и т. д.

# 2. Обзор операционных систем реального времени

На сегодняшний день существует более 100 коммерческих ОСРВ. Есть множество бесплатных (или условно бесплатных) СРВ и систем, имеющих статус исследовательских или университетских проектов. Сначала рассмотрим краткое описание некоторых систем реального времени, а затем более подробно остановимся на СРВ Win NT RTX, как наиболее перспективной системе.

# 2.1 QNX

Операционная система QNX является разработкой канадской компании QNX Software System Ltd (1981).

Операционная система QNX представляет собой гибрид 16/32-битовой операционной системы, которую пользователь может конфигурировать по своему усмотрению. Наиболее часто она применяется для создания систем, работающих в реальном масштабе времени. Время, необходимое для полной инсталляции системы, включая сетевые средства, составляет всего 10-15 мин, после чего можно начинать работу. Нетребовательность системы к ресурсам проявляется уже в том, что система с необходимой и достаточной средой разработки в виде компилятора Watcom C/C++ (основной компилятор для QNX) умещается на 10 Мб.

**QNX** - первая коммерческая ОС, построенная на принципах микроядра и обмена сообщениями. Система реализована в виде совокупности независимых (но взаимодействующих через обмен сообщениями) процессов различного уровня (менеджеры и драйверы), каждый из которых реализует определенный вид сервиса.

Эти идеи позволили добиться нескольких важнейших преимуществ:

* предсказуемость, означающая ее применимость к задачам жесткого реального времени. Ни одна версия UNIX не может достичь подобного качества, поскольку код ядра слишком велик. Любой системный вызов из обработчика прерывания в UNIX может привести к непредсказуемой задержке (как и в Windows NT);
* масштабируемость и эффективность, достигаемые оптимальным использованием ресурсов и означающие ее применимость для встроенных (embedded) систем. В каталоге dev присутствуют только необходимые для поставленных задач файлы, соответствующие нужным драйверам. Драйверы и менеджеры можно запускать и удалять (кроме файловой системы) динамически, просто из командной строки. Возможна также покупка только тех модулей, которые реально необходимы для обеспечения нужных функций;
* расширяемость и надежность одновременно, поскольку написанный драйвер не нужно компилировать в ядро, рискуя вызвать нестабильность системы.

Система построена по технологии **FLEET** [Fault-tolerance (отказоустойчивая), **Load-bаlаncing** (регулирующая нагрузку), **Еffiсiеnt** (эффективная), **Ехtеnsible** (расширяемая), **Тгаnsparent** (прозрачная)], которая характеризуетмя следующим. QNX является ОСРВ на основе микроядра (размером около 10 Кб). В качестве основного средства взаимодействия между процессами система использует передачу сообщений. Благодаря этому в 32-битовой среде возможно взаимодействие процессов с 32 и 16-битовыми кодами, причем сообщения передаются между любыми процессами, независимо от того, находятся ли процессы на одном компьютере или на разных узлах сети. Пользователь, работая на одном из узлов сети, может иметь доступ к любым ресурсам остальных узлов, включая порты, файловую систему и задачи. Пользователю нет необходимости вникать в сетевой протокол, который, кстати, не является тайной, вплоть до его структуры. Он содержит пакеты, которые применяются и для передачи сообщений. Сетевой администратор распознает эти пакеты и переправляет микроядру, которое, в свою очередь, переправляет их в шину локальных сообщений. QNX распознает не только пакеты сообщений QNX-процессов. Можно также легко обращаться к сетевому администратору для передачи таких пакетных протоколов, как TCP/IP, 8MB и др. Возможно обращение к различным сетевым администраторам через один кабель.

Операционная система QNX объединяет всю сеть ПК в единый набор ресурсов с абсолютной прозрачностью доступа к ним. Узлы могут добавляться и исключаться из сети, не влияя на целостность системы. Сетевая обработка данных в QNX является настолько гибкой, что можно объединить в одну сеть любой разнородный набор Intel совместимых компьютеров, соединенных через Arcnet, Ethernet, Token Ring или через последовательный порт, к которому также может быть подключен модем. Кроме того, возможно участие компьютера одновременно в нескольких сетях, и если одна из них окажется перегруженной или выйдет из строя, то QNX автоматически будет использовать другие доступные сети без потери информации.

QNX имеет некоторые ограничения, связанные с ориентацией системы на рынок встроенных систем реального времени:

* нет поддержки SMP;
* отсутствует запись виртуальной памяти на диск;
* неэффективная и нестандартная поддержка нитей;
* неполноценная реализация отображения файлов в памяти;
* нет поддержки UNIX-domain sockets;
* слабые средства безопасности в рамках собственного сетевого протокола.

Несмотря на присущие минусы, для QNX разработано множество пользовательских программ, например, базы данных, которые по производительности часто превосходят аналоги под управлением других операционных систем.

# 2.1.1 Профессиональный пакет

 В состав QNX входят графическая оболочка Photon microGUI — полнофункциональная и в то же время встраиваемая оконная среда с расширяемой подсистемой мультимедиа и поддержкой аппаратно независимого многослойного интерфейса. В её состав также включены масштабируемые шрифты и встроенная поддержка Unicode; широкий спектр файловых систем, включая образную файловую систему, файловую систему в ОЗУ или ППЗУ Flash, файловые системы QNX, Linux, DOS, CD - ROM, DVD, NFS, CIFS, пакетную файловую систему и файловую систему со сжатием; ядро с поддержкой SMP — стандартное микроядро можно заменить на SMP совместимое, предоставляющее настоящую поддержку тесно связанного SMP для многопроцессорных плат на основе MIPS, PowerPC и x 86; администратор систем высокой готовности поддерживает квитанции работоспособности для ранней диагностики отказов и предоставляет интеллектуальные механизмы перезапуска отказавших компонентов; поддержка Java две среды исполнения, оптимизированные для QNX: WebSphere Embedded Environment (стандарт Java Powered ) и WebSphere Custom Environment; POSIX API поддержка POSIX 1003.1-2003, включая многопоточность, расширения реального времени и множество других опций; стеки TCP / IP — встраиваемый стек, стек NetBSD или расширенный стек NetBSD с поддержкой IPSec и IPv6; отказоустойчивый сетевой интерфейс в QNX приложениях могут прозрачно взаимодействовать по резервированным сетевым соединениям. Если одно соединение нарушается, ОС автоматически перенаправит сетевой трафик по одному или нескольким альтернативным маршрутам. Поддерживается также балансировка нагрузки между всеми доступными соединениями для увеличения пропускной способности. Функции реального времени QNX обеспечивает быстрое, предсказуемое время реакции за счёт вытесняющего планирования, сверхмалых задержек обработки прерываний, распределённого наследования приоритетов и многих других реализованных в ней современных механизмов. В поставку профессионального пакета QNX Momentics входит всё, что необходимо на каждом этапе разработки системы, от встраивания на процессорную плату до системного анализа. Помимо графической IDE разработчики могут использовать и обычный командностроковый инструментарий, получая абсолютно такие же бинарные модули и контекст. QNX Momentics обеспечивает разработчикам свободу выбора инструментария разработки, так как его интегрированная среда основана на Eclipse — открытой платформе, поддерживаемой большим и постоянно расширяющимся сообществом компаний производителей. Eclipse также предоставляет расширяемую архитектуру подключаемых модулей, позволяющую QNX Momentics работать с практически любым типом информационного содержания. Например, в состав QNX Momentics входит множество подключаемых модулей для разработки и анализа встраиваемых образов, исходных текстов на C/C++ и прочих объектов, характерных для встраиваемых систем. Поддержка множества языков программирования ( C, C++, встраиваемый C++ или Java ), инструментальных ОС ( Windows, Solaris или QNX ) и целевых процессоров (ARM, MIPS, PowerPC, SH -4, StrongARM, XScale или x86) позволяет существенно сократить время разработки проекта независимо от его масштаба и сложности.

Средства разработки кода QNX Momentics включают в себя:

* + «мастера» проектов,
	+ редакторы кода,
	+ средства управления исходными текстами,
	+ средства построения проектов.

QNX Momentics поддерживает большое количество широко распространённых библиотек, включая

* + ANSI C,
	+ POSIX,
	+ Dinkum C++, полная версия,
	+ Dinkum C++, встраиваемая версия с сокращенной STL ,
	+ GNU C++ (только для x86),
	+ сжатие,
	+ сеть,
	+ графика,
	+ виджеты,
	+ XML.

Использование стандартных инструментов включает в себя:

* + GCC v2.95x, GDB v5.x,
	+ LD v2.10.x, поддерживается эмуляция: i386nto, armnto, elf32bmpinto, elf32ppcnto, shielfnto,
	+ MAKE v3.79x,
	+ JDK 1.3 совместимый Java компилятор (с поддержкой инкрементной компиляции для увеличения производительности больших проектов).

IDE включает в себя встроенную поддержку протокола управления исходными текстами CVS включая поддержку удалённого сервера и доступ к защищенным репозитариям посредством sSh. Также поддерживается система управления исходными текстами ClearCase, поставляемая компанией Rational Software в виде подключаемого модуля для Eclipse.

Возможности:

* + локальное управление версиями;
	+ распределённое управление версиями;
	+ поддержка журнала изменения файлов (кем и какие изменения внесены);
	+ визуальное сравнение версий;
	+ интерактивное слияние изменений в ситуации, когда несколько разработчиков изменяют один и тот же файл.

Используя PhAB™, визуальное средство разработки приложений QNX Photon microGUI®, можно создавать полнофункциональные пользовательские интерфейсы с простотой щелчка мыши. Возможности PhAB™ включают в себя:

* + готовые шаблоны PhAB;
	+ обширная палитра доступных элементов управления (виджетов);
	+ привязка к виджетам диалоговых окон или вызовов функций (предопределенные, по «горячей» клавише или по низкоуровневым событиям);
	+ полная поддержка со стороны интегрированной среды QNX Momentics;
	+ минимизация кода;
	+ многоязычная поддержка.

# 2.1.2 Рабочая станция

Системный профайлер позволяет разрешать конфликты синхронизации, определять ситуации взаимных блокировок, выявлять корни семантических ошибок, находить скрытые неполадки в программном и аппаратном обеспечении и оптимизировать производительность приложения, причём как для однопроцессорных, так и для многопроцессорных целевых систем. Характеристики системы могут быть проанализированы в реальном времени, в моменты возникновения событий. Удобный инструмент поиска позволяет проанализировать детали по каждому событию, включая время возникновения, владельца и тип. Системный профайлер может отображать огромные объёмы информации, включая информацию о вызовах ядра, аппаратных прерываниях, состоянии потоков, обмене сообщениями и событиях планировщика. Сложные комбинации условий могут быть отслежены благодаря развитой системе динамических фильтров, определяемых пользователем. В приложения могут быть встроены средства генерации специализированных сообщений для подсистемы трассировки, оказывающей упреждающее воздействие на процесс записи событий. Профайлер приложений предоставляет информацию об использовании процессорного времени каждым потоком и отображает её одновременно как в виде абсолютных значений, так и в виде процентной доли от общего времени с возможностью сортировки. Профайлер может анализировать динамически загружаемые разделяемые библиотеки, отвечая тем самым на вопрос, где кроется причина снижения производительности в коде приложения или в библиотеке, которую оно вызывает. ОС QNX предоставляет пользователю библиотеку распределения памяти, содержащую реализацию большинства типовых операций над строками и памятью. Эти функции перед выполнением операции проверяют корректность использования указанной области памяти, позволяя выявлять ошибки типа переполнения, выборки из пустого буфера, некорректного использования памяти и повторного освобождения одной и той же области. «Интеллектуальный» механизм отслеживает ошибки работы с памятью. При возникновении ошибки соответствующий фрагмент исходного текста будет помечен предупреждением, при этом можно:

* + продолжить выполнение программы;
	+ завершить программу и сохранить её образ в дамп файле;
	+ остановить программу и немедленно переключиться в отладчик для локализации проблемы.

Анализатор ОЗУ помогает визуально представлять использование памяти программами и может быстро выявлять переполнения буферов, некорректные освобождения памяти и множество других типовых ошибочных ситуаций. Анализатор ОЗУ предоставляет:

* + информацию на уровне процесса, которая позволяет быстро оценить карту памяти программы;
	+ специализированную статистику распределения памяти для выявления проблем с использованием «кучи». Статистика включает в себя суммарное количество свободных, распределённых и используемых байтов и блоков; динамический журнал использования памяти для оценки динамики ситуации. Комплекты разработки драйверов (DDK) позволяют быстро создавать драйверы для нестандартного оборудования — аудио, графических и сетевых адаптеров, устройств ввода, принтеров, символьных и USB устройств. Комплекты включают в себя исходные тексты, детальную документацию и готовый программный каркас, в котором весь высокоуровневый аппаратно независимый код уже реализован в виде библиотек.

Поскольку в QNX драйверы выполняются как обычные пользовательские процессы, их можно отлаживать и оптимизировать при помощи того же интегрированного инструментария, который QNX Momentics предоставляет для отладки обычных приложений. Нет никакой необходимости в использовании отладчиков на уровне ядра, которые могут застопорить работу всей целевой системы, в результате скрывая ошибки. Микроядерная архитектура QNX позволяет тестировать изменения в коде драйверов без перезагрузки системы и даже без перезапуска сеанса отладки необходимо просто перекомпилировать и перезапустить драйвер. Наличие встроенного эмулятора позволяет тестировать и отлаживать драйверы непосредственно на инструментальном компьютере, не теряя времени на ожидание целевой аппаратуры. QNX Momentics предоставляет полный набор инструментария для начальной загрузки и взаимодействия с целевым оборудованием. В этот инструментарий входят пакеты поддержки (BSP) для широкого спектра процессорных плат, построитель встраиваемых систем, позволяющий быстро формировать и настраивать целевые образы, и уникальный целевой агент, динамически загружающий сервисные модули по мере необходимости. QNX Momentics также включает в себя навигатор целевых систем, который позволяет привязывать проекты к IP адресам или именам хостов, однозначно определяя программно аппаратную конфигурацию целевых систем. Эти конфигурации впоследствии могут использоваться при работе с другими инструментальными средствами. Навигатор целевых систем также обеспечивает интерактивное отображение выполняющихся процессов, позволяя просмотреть следующую информацию о целевой системе: использование ресурсов, атрибуты потоков, файловые дескрипторы и т.д. QNX Momentics предоставляет богатый выбор готовых пакетов поддержки процессорных плат (BSP) на основе процессоров ARM, MIPS, x86, PowerPC, SH-4, StrongARM и XScale. Каждый BSP снабжён детальной документацией и исходными текстами всех бинарных модулей, включая начальный загрузчик, стартовый код и драйверы устройств.

# 2.2 VxWorks/Tornado

Операционная система реального времени VxWorks и инструментальная среда Tornado фирмы Wind River Systems предназначены для разработки ПО встроенных компьютеров, работающих в системах жесткого реального времени. Операционная система VxWorks является системой с кросс-средствами разработки прикладного программного обеспечения. Разработка ведется на инструментальном компьютере (host) в среде Tornado для последующего исполнения на целевой машине (target) под управлением VxWorks.

VxWorks поддерживает целевые архитектуры (targets):

* Motorola 680x0 и CPU32, PowerPC;
* Intel 386/486/Pentium, Intel 960;
* Spare, Mips R3000/4000;
* AMD 29K, Motorola 88110;
* HP PA-RISC;
* Hitachi SH7600;
* DEC Alpha.

Инструментальные платформы, поддерживаемые для Tornado (hosts):

* Sun SPARCstation (SunOS и Solaris);
* HP 9000/400,700 (HP-UX);
* IBM RS6000 (AIX);
* Silicon Graphics (IRIX);
* DEC Alpha (OSF/1);
* PC (Windows).

Поддерживаемые интерфейсы host-target:

* host-target Ethernet;
* RS232;
* внутрисхемный эмулятор ICE (In-Circuit Emulator);
* кросс-шина (backplane).

Операционная система VxWorks построена, как и положено ОС жесткого реального времени, по технологии микроядра, т. е. на нижнем непрерываемом уровне ядра выполняются только базовые функции планирования задач и их управления коммуникацией/синхронизацией. Все остальные функции операционной системы более высокого уровня (управление памятью, вводом/выводом, сетевые средства и т. д.) базируются на простых функциях нижнего уровня, что позволяет обеспечить быстродействие и предсказуемость ядра, а также легко построить необходимую конфигурацию операционной системы.

В многозадачном ядре wind применен алгоритм планирования задач, учитывающий приоритеты и включающийся по прерываниям. В качестве основного средства синхронизации задач и взаимоисключающего доступа к общим ресурсам в ядре wind применены семафоры. Имеется несколько видов семафоров, ориентированных на различные прикладные задачи: двоичные, целочисленные, взаимного исключения и POSIX.

Все аппаратно-зависимые части VxWorks вынесены в отдельные модули для того, чтобы разработчик встроенной системы мог сам портировать VxWorks на свою нестандартную целевую машину. Этот комплект конфигурационных и инициализационных модулей называется BSP (Board Support Package) и поставляется для стандартных компьютеров (VME-процессор, PC или Sparcstation) в исходных текстах. Разработчик нестандартной машины может взять за образец BSP наиболее близкий по архитектуре стандартный компьютер и перенести VxWorks на свою машину путем разработки собственного BSP с помощью BSP Porting Kit.

# 2.2.1 Базовые сетевые средства VxWorks: UNIX-networking, SNMP и STREAMS.

VxWorks была первой операционной системой реального времени, в которой реализован протокол TCP/IP с учетом требований реального времени. С тех пор VxWorks поддерживает все сетевые средства, стандартные для UNIX: TCP/UDP/ICMP/IP/ARP, Sockets, SLIP/CSLIP/PPP, telnet/rlogin/rpc/rsh, ftp/tftp/bootp, NFS (клиент и сервер).

Wind River Systems анонсировала (1994) программу WindNet, по которой ведущие фирмы-производители программных средств в области коммуникаций интегрировали свои программные продукты с VxWorks.

На сегодняшний день - это сетевые протоколы Х.25, ISDN, ATM, SS7, Frame Relay и OSI; CASE-средства разработки распределенных систем на базе стандартов ROOM (Real-Time Object Oriented Modelling) и CORBA (Common Object Request Broker Architecture); менеджмент сетей по технологиям MBD (Management By Delegation) и CMIP/GDMO (Common Management Information Protocol/Guidelines for Definition of Managed Objects).

# 2.2.2 Мониторинг и отладка в реальном масштабе времени: WindView.

Обычные отладчики, позволяющие исследовать состояние программ и данных в точках останова, являются статическими средствами отладки. Возможности исследования динамики исполнения программ и изменения данных предоставляют специальные средства отладки в реальном масштабе времени, которые трассируют интересующие пользователя события и накапливают их в буфере для последующего анализа.

Трассировку системных событий (переключения задач, запись в очередь сообщений, установка семафора и т. д.) позволяет вести динамический анализатор WindView, который отображает накопленные в буфере события на временной диаграмме. В последнее время высокопроизводительные микропроцессоры, а с ними и операционные системы реального времени, все чаще используются в так называемых "глубоко встроенных" (deeply embedded) применениях (автомобильная электроника, офисная и бытовая техника, измерительные и медицинские приборы и др.). К таким компьютерным системам предъявляются два основных требования: малые габариты и низкая стоимость, поэтому глубоко встроенные микропроцессорные системы ставят две проблемы на пути применения серийных ОСРВ: небольшие объемы используемой памяти и отсутствие "лишних" интерфейсов, по которым можно было бы связать целевую и инструментальную машины на этапе разработки встроенного ПО.

Специально для систем с сильно ограниченным объемом памяти компания Wind River Systems разработала редуцированное ядро WindStream, которое требует для работы не более 8 Кб ПЗУ и 2 Кб ОЗУ. При этом для WindStream применим весь спектр инструментальных средств VxWorks, включая WindView.

Инструментальная среда Tornado имеет открытую архитектуру, что позволяет другим фирмам-производителям инструментальных средств разработки ПО реального времени интегрировать свои программные продукты с Tornado. Пользователь может подключать к Tornado свои собственные специализированные средства разработки, а также расширять возможности инструментальных средств фирмы Wind River Systems.

**В стандартную конфигурацию Tornado** входят ядро VxWorks и системные библиотеки, GNU C/C++ Toolkit, дистанционный отладчик уровня исходного языка CrossWind, оболочка WindSh, конфигуратор BSP WindConfig и др.

# 2.3 RTLinux

# 2.3.1 Основные сложности при реализации систем реального времени в среде LINUX

Как было сказано выше, основной задачей является реагирование на некоторое внешнее событие в заданный промежуток времени. Внешнее событие обычно, с точки зрения программиста, выглядит как аппаратное прерывание. В современных многозадачных операционных системах первым на аппаратное прерывание реагирует ядро. Затем это прерывание через драйвера устройств каким-то образом может попасть и к прикладной задаче. Но в многозадачной системе должны одновременно работать сразу несколько задач и для того, чтобы доставить прерывание, ядро должно перевести процесс, выполняющийся в данный момент в состояние сна, пробудить нужный процесс и передать ему прерывание. Для этого нужно переключать контексты, что требует много времени, поэтому прерывание будет доставлено процессу со значительным опозданием. Кроме того, после получения прерывания процессом нельзя быть уверенным, что обработка информации будет завершена в минимальные сроки, т.к. если компьютер оснащен всего одним процессором, а в системе запущено больше одной задачи - то в любой момент может произойти переключение задач с очередным переключением контекстов. В результате - время реакции может получиться неоправданно большим (на достаточно производительном компьютере).

**Linux** - современная POSIX-совместимая и Unix-подобная операционная система для ПК и рабочих станций, т. е. многопользовательская сетевая операционная система. ОС Linux поддерживает стандарты открытых систем и протоколы сети Internet. Все компоненты системы, включая исходные тексты, распространяются с лицензией на свободное копирование и установку для неограниченного числа пользователей.

Характерные особенности Linux как ОС:

* многозадачность (является обязательным условием);
* многопользовательский режим;
* защищенный режим процессора (386 protected mode);
* защита памяти процесса (сбой программы не может вызвать зависания системы);
* экономная загрузка: Linux считывает с диска только те части программы, которые действительно используются для выполнения;
* разделение страниц по записи между экземплярами выполняемой программы. Это значит, что процессы-экземпляры программы могут использовать при выполнении одну и ту же память. Когда такой процесс пытается произвести запись в память, то 4-килобайтная страница, в которую идет запись, копируется на свободное место. Это свойство увеличивает быстродействие и экономит память;
* виртуальная память со страничной организацией (т. е. на диск из памяти вытесняется не весь неактивный процесс, а только требуемая страница); виртуальная память в самостоятельных разделах диска и/или файлах файловой системы; объем виртуальной памяти до 2 Гб; изменение размера виртуальной памяти во время выполнения программ;
* общая память программ и дискового кэша: вся свободная память используется для буферизации обмена с диском;
* динамические загружаемые разделяемые библиотеки;
* дамп программы для пост-мортем анализа: позволяет анализировать отладчиком не только выполняющуюся, но и завершившуюся аварийно программу;
* сертификация по стандарту POSIX.1, совместимость со стандартами System V и BSD на уровне исходных текстов;
* через iВS2-согласованный эмулятор совместимость с SCO, SVR3, SVR4 по загружаемым программам;
* наличие исходного текста всех программ, включая тексты ядра, драйверов, средств разработки и приложений. Эти тексты свободно распространяются. В настоящее время некоторыми фирмами для Linux поставляется ряд коммерческих программ без исходных текстов, но все, что было свободным так и остается свободным;
* управление заданиями в стандарте POSIX;
* эмуляция сопроцессора в ядре, поэтому приложение может не заботиться об эмуляции сопроцессора. Конечно, если сопроцессор имеется в наличии, то он и используется;
* множественные виртуальные консоли: на одном дисплее несколько одновременно независимых сеансов работы, переключаемых с клавиатуры;
* поддержка ряда распространенных файловых систем (MINIX, Xenix, файловые системы System V); наличие собственной передовой файловой системы объемом до 4 Тб и с именами файлов до 255 знаков;
* прозрачный доступ к разделам DOS (или OS/2 FAT): раздел DOS выглядит как часть файловой системы Linux; поддержка VFAT (WNT, Windows 95);
* доступ (только чтение) к файловой системе HPFS-2 OS/2 2.1;
* поддержка всех стандартных форматов CD ROM;
* поддержка сети TCP/IP, включая ftp, telnet, NFS и т. д.

Рост популярности Linux побуждает разработчиков внимательнее присмотреться к этой операционной системе. В данный момент эта ОС готова к стабильной работе, а открытость ее исходных текстов и архитектуры наряду с растущей популярностью заставляет программистов переносить свои наработки на многие аппаратные платформы: SGI, IBM, Intel, Motorola и т. д.

Для задач РВ сообщество разработчиков Linux активно применяет специальные расширения - RTLinux, KURT и UTIME, позволяющие получить устойчивую среду реального времени. RTLinux представляет собой систему "жесткого" реального времени, a KURT (KU Real Time Linux) относится к системам "мягкого" реального времени. Linux-расширение UTIME, входящее в состав KURT, позволяет добиться увеличения частоты системных часов, что приводит к более быстрому переключению контекста задач.

**RTLinux** - это операционная система, в которой небольшое ядро реального времени сосуществует с Posix-like ядром Linux. Основная цель - сделать доступными сложные службы и оптимизированное поведение системы в стандартных ситуациях для системы с разделением времени, и, в то же время, выполнять задачи реального времени. В прошлом операционные системы реального времени примитивны - простые программы, которые предлагали пользователю чуть больше, чем просто библиотека основных функций. Но в наше время пользователи требуют доступ к TCP/IP, графическому дисплею и системе окон, базам данных и другим службам, которые не являются ни примитивными, ни простыми. Одно из решений - добавить non-real-time службы к базовому ядру реального времени, что и было проделано в VXworks и, немного по-другому, в микроядре QNX. Вторая возможность - модифицировать стандартное ядро и сделать его полностью прерываемым.

# 2.3.2 Организация RTLinux

RTLinux организован третьим способом, в котором простое ядро реального времени запускает обычное ядро как одну из задач реального времени с самым низким приоритетом, используя виртуальную машину для того, чтобы сделать стандартное ядро полностью прерываемым.

В RTLinux все прерывания обслуживаются ядром реального времени, а затем передаются стандартному ядру, но только в том случае, если нет необходимости запускать одну из задач реального времени. Для того чтобы минимизировать количество изменений в стандартном ядре, этот механизм реализован при помощи эмулирования ICH (Interrupt Control Hardware). Ядро реального времени и пользовательские задачи Linux могут обмениваться данными через неблокируемые очереди и сегменты разделяемой памяти.

С точки зрения программиста очереди выглядят как стандартные последовательные устройства UNIX, доступ к которым возможен при помощи системных вызовов POSIX read/write/open/ioctl. Разделяемая память доступна через системный вызов mmap.

RTLinux использует Linux для загрузки, доступа к большинству устройств, работы с сетью, файловыми системами, управлением процессами Linux и загрузки модулей ядра, что дает возможность легко модифицировать систему реального времени.

Программа реального времени состоит из двух частей: задачи, которая представляет собой модуль ядра, и обыкновенный UNIX/Linux процесс и заботится об обработке данных, доступу к дисплею и сети и о любых других функциях, не требующих таких жестких временных рамок.

На практике оказалось, что идея RTLinux очень удачна. В самом худшем случае запаздывание прерываний на 486/33Mhz PC оказалось менее 30 мкс, что близко к аппаратному пределу. Для прикладных задач симбиоз систем реального времени и оптимизированной для "общего случая" оказался очень удачным. Наиболее часто используемая конфигурация RTLinux - примитивные задачи реального времени со статически распределяемой памятью без ее защиты, простым планировщиком с фиксированными приоритетами без защиты от нереализуемых планов, аппаратным запрещением прерываний, разделяемая память - единственный механизм синхронизации задач реального времени и ограниченный набором операций над FIFO-очередями, подсоединенными к обычным процессам Linux.

Ядро Linux позволяет в динамике загружать и выгружать модули ядра. Представив отдельные части ядра реального времени в виде модулей, легко изменять ядро реального времени. Уже написаны альтернативные планировщики и модуль семафоров. Во время работы системы можно загрузить модуль с задачами реального времени, затем выгрузить стандартный планировщик и загрузить, например, EDF планировщик. Можно пробовать разные комбинации модулей, пока не будет найдена оптимальная.

Этот вариант Linux позволяет выполнять задачи в реальном времени, что достигается путем вставки ядра реального времени между стандартным ядром Linux и аппаратными прерываниями и позволяет избавиться от главной причины непригодности Linux для задач реального времени - большого запаздывания прерываний.

С точки зрения RTLinux, Linux - одна из задач реального времени, имеющая самый низкий приоритет, может быть прервана, когда нужно. Такая структура накладывает некоторые ограничения на задачи реального времени. Они не могут легко использовать различные драйверы Linux, не имеют доступ к сети и т. д., но зато могут обмениваться данными с стандартными задачами Linux.

Простые очереди FIFO реализованы для обмена данными между процессами реального времени и процессами Linux. Типичное приложение состоит из двух частей - задачи реального времени, непосредственно работающей с аппаратурой, и, обычно, задачи Linux, которая выполняет остальные операции, такие как сохранение данных на диск, пересылка их по сети, работа с пользователем (GUI) и т. д.

Самый короткий период для периодически вызываемых задач реального времени в RTLinux на Pentium 120 - менее 150 мкс. Задачи, вызываемые по прерыванию, могут иметь намного меньший период.

Ядро реального времени не защищает от перегрузок. Если одна из задач реального времени полностью утилизирует процессор, ядро Linux, имея самый низкий приоритет, не получит управления и система повиснет. Задачи реального времени запускаются в адресном пространстве ядра с привилегиями ядра и могут быть реализованы, например, при помощи модулей Linux.

Необходимо отметить, что компания LynuxWorks начала поставки (17.05.2002) встраиваемой ОС BlueCat Linux для комплекта разработчика ПО Intel Internet Exchange Architecture Software Developers Kit (Intel IXA SDK) 2.0, предназначенного для семейства сетевых процессоров Intel IXP1200. ОС BlueCat Linux распространяется бесплатно совместно с Intel IXA SDK 2.0.

VxWorks давно стала де-факто стандартом для подавляющего большинства систем, использующих встроенные ОС. Флэш-память вычислительной системы IXP1200 содержит загрузчик ядра VxWorks. Для разработчиков это упрощает задачу написания новых программ. Кроме того, уже реализована возможность работы сетевого процессора под управлением ОС Linux (с расширениями реального времени). Осуществляется программная поддержка некоторыми производителями ОС Linux (например,LynuxWorks и т. д.).

# 2.4 Контроль и управление в реальном времени с использованием OS9

# 2.4.1 Введение

Для управления из единого центра одной или многими удалёнными установками и проверки их состояния фирма CS (Франция) разработала систему контроля. Она может применяться для управления энергетическими объектами, в системах кондиционирования воздуха, системах безопасности, технологических установках, сетях, промышленных конвейерах и уже внедрена на многих гражданских и военных объектах.

Система предоставляет аппаратное и программное решение всех аспектов контроля. Широкие возможности сопряжения обеспечивают сбор данных с наружных датчиков, оконечных устройств, программируемых блоков ввода/вывода и с хост-систем.

В системе контроля можно выделить два уровня:

* Уровень 1 для сбора и обработки локальных данных;
* Уровень 2 для сбора и анализа данных с удаленных объектов.

Задачи на уровне 1 выполняются блоком GESCAP (например, рабочей станцией на процессоре 68010 фирмы Motorola с ОЗУ 1 Мбайт), а на уровне 2 - блоком GESVA (например, рабочей станцией на процессоре 68030 фирмы Motorola с ОЗУ 4 Мбайт). В блоках GESVA и GESCAP используется одна и та же операционная система (OS-9) и прикладное программное обеспечение. Различны только видеотерминалы и функциональные возможности. Благодаря этой однородности облегчается установка, использование и поддержание работоспособности системы.

Поскольку пользовательский интерфейс GESVA и GESCAP соответствует интуитивным представлениям, работа по конфигурированию не вызывает трудностей и не требует никаких особых познаний в информатике. Пользователь имеет дело с всплывающими меню и диалоговыми окнами, обеспечивающими доступ к следующим стандартным функциям.

Конфигурирование аналоговых или цифровых блоков ввода/вывода (аварийные сигналы, сообщения, приоритеты, и т.д.):

* логические операции (И, ИЛИ ...) над входными данными;
* установление соответствия разъёмов каждому устройству ввода/вывода;
* перегруппировка устройств ввода/вывода;
* графическое взаимодействие с элементами анимации.

Функции отображения состояния:

состояние блоков ввода/вывода, устройств, соединений, аварийных сигналов, бортовой журнал реального времени.

Графические функции:

* автоматическое расположение аварийных сигналов в изображениях;
* перемещение по изображениям источников данных;
* отображение кривых в реальном времени;
* видеоотображение кривых в реальном времени.

Функции связи:

* протокол связи JBUS/MODBUS (RS-232/RS-422);
* связь с системой помощи в локализации и устранении неисправностей (GESDOC).

Функции выдачи на печать:

* распечатка структуры системы;
* распечатка списка сигналов тревоги и событий.

Функции различного назначения:

интеграция новых задач с программным обеспечением, не требующая никаких изменений на уровне исходного кода.

# 2.4.2 Как развивалась эта прикладная система?

Основными требованиями к программному обеспечению были:

* один и тот же состав прикладных программ и единый пользовательский интерфейс на различных платах и типах отображения;
* быстрая развитая многооконная среда, расходующая как можно меньше памяти;
* многозадачная, прошиваемая в ПЗУ операционная система реального времени с небольшими требованиями к объему памяти.

В результате выбрана операционная система OS9, наилучшим образом отвечающая перечисленным требованиям.

В качестве многооконной среды разработки выбран пакет WINOTOOLS. С помощью утилит WINOTOOLS можно без программирования разрабатывать все элементы пользовательского интерфейса (окна, средства графического ввода данных, кнопки, предупреждающие сообщения и т.п.).

Благодаря скромным требованиям к объему памяти и малому времени реакции, WINOTOOLS в сочетании с характеристиками OS9 является идеальной средой разработки разнообразных прикладных программ реального времени, работающих в графическом режиме.

# 2.4.3 Утилиты WINOTOOLS:

* **PAINT** - объектно-ориентированный графический редактор для разработки пользовательских интерфейсов, пультов управления и диаграмм. PAINT разработан таким образом, чтобы его можно было настраивать и интегрировать в пользовательскую прикладную программу. Все файлы графических данных, созданные с помощью PAINT, могут целиком модифицироваться без обращения к исходным кодам (импортируются файлы в формате GIF).
* **SCRIPT** - язык, позволяющий моделировать все диалоги с пользовательским интерфейсом до написания строки исходного кода. Эта утилита позволяет производить быструю проверку правильности пользовательского интерфейса.
* **DATABASE** (БАЗА ДАННЫХ) - библиотека С-функций, обеспечивающая управление базой данных (соответствие данных и индексных файлов базы данных).
* **БИБЛИОТЕКА ВВОДА/ВЫВОДА** - библиотека С-функций, обеспечивающая сопряжение пользовательского интерфейса с устройствами ввода/вывода.

# 2.5 Windows NT

# 2.5.1 Возможность использования Windows NT в качестве ОС реального времени

В последнее время приобретают популярность расширения реального времени для Windows NT. Это обусловлено с одной стороны - расширением областей применения компьютерного управления, с другой стороны - сравнительно малой известностью и высокой стоимостью специализированных операционных систем реального времени. Но даже если бы это было не так и о других системах было бы известно не меньше - все равно Win NT RTX была бы наиболее популярна. Ведь не даром соотношение пользователей ОС семейства Windows к пользователям Linux/Unix систем - 1000 к 1. Вполне логично, что человек, работая дома в одной системе, хочет видеть такую же, понятную ему удобную систему и на работе. Интерфейс Win32 является стандартным и хорошо знакомым большому числу программистов и пользователей. Под NT существует огромное число готовых приложений (в том числе коммуникационных), а также популярные средства разработки. К сожалению, Windows NT "в чистом виде" нельзя отнести к операционным системам реального времени. Обсуждению причин этого посвящены статьи Martin Timmerman и Jean-Christophe Monfret в Real-Time Magazine Q21997.

Вот некоторые из них:

* недостаточное количество real-time приоритетов;
* отсутствие наследования приоритетов, как средства борьбы с инверсией приоритетов;
* неподходящая для RTOS система обработки прерываний.

В Windows NT доступ к прерываниям осуществляется из драйвера ядра,а сами прерывания обрабатываются в два этапа: сначала вызывается очень короткая Interrupt Service Routine (ISR), осуществляющая критическую обработку, основная обработка прерывания происходит в Deferred Procedure Call (DPC). Все DPC выполняются с одинаковым уровнем приоритета в порядке поступления (FIFO).

Таким образом, время окончания обработки вашей DPC оказывается непредсказуемым образом зависимым от наличия в системе других драйверов и их активности. Для систем с жестким детерминизмом необходимо точно знать максимальное время от момента возникновения прерывания до входа в процедуру обработки и гарантировать его непревышение.

# 2.5.2 RTX - real-time extension для Windows NT от компании VenturCom

Одним из возможных решений является использование совместно с Windows NT подсистемы реального времени, исполняющейся на том же процессоре (если процессор один) или на выделенном процессоре(-ах) (если их несколько). Этот подход использован фирмой VenturCom в продукте RTX. Сущность подхода заключается в использовании модифицированного HAL (Hardware Abstraction Level). Изменять kernel Microsoft не разрешает, а исходный код HAL предоставляет своим партнерам, одним из которых является VenturCom.

После установки RTX стандартная NT Workstation или Server превращается в операционную систему реального времени с жестким детерминизмом (hard real-time). Сама NT об этом, правда, не подозревает. Ни ядро, ни исполняющая подсистема NT не были изменены. Подсистема реального времени видна из Windows NT как еще один драйвер устройства.

Операционная система Windows NT первоначально разрабатывалась как система общего назначения. Однако, на нынешнем рынке специализированных систем с целью обеспечения открытости на каждом системном уровне существует тенденция использования операционных систем Microsoft Windows. Это обусловлено следующими причинами:

* прикладной программный интерфейс (API) Win32 на данный момент является для программистов стандартом де-факто;
* графический интерфейс пользователя (GUI-интерфейс) популярен настолько, что GUI-интерфейсы других операционных систем начинают на него походить все больше;
* разработано огромное количество драйверов независимых производителей;
* разработано множество очень мощных интегрированных инструментальных пакетов (IDE).

# 2.5.3 Windows NT 4.0 как ОСРВ. Общие требования

Чтобы называться ОСРВ, операционная система должна удовлетворять некоторому минимальному набору требований, которые необходимы, но недостаточны. В самом начале своей работы я это указывал, но повторимся еще раз:

* операционная система должна поддерживать многопоточность (многонитиевость - multithreaded) и вытеснение задач по приоритетам (preemptable);
* должно существовать понятие приоритета потока (нити);
* операционная система должна поддерживать механизмы синхронизации исполнения потоков (нитей) с предсказуемыми характеристиками;
* должен иметься механизм наследования приоритетов;
* поведение операционной системы должно быть известным и предсказуемым (внутренние задержки прерываний, задержки переключения задач, задержки драйверов и т.д.).

Первому требованию Windows NT явно удовлетворяет. Второму - тоже, но для режима реального времени уровней приоритета маловато. Практически невозможно спроектировать хорошую систему реального времени, например, с диспетчеризацией в зависимости от частоты (rate-monotonic scheduling), поскольку доступного количества приоритетных уровней исполнения потоков не хватит. Кроме того, в NT отсутствует механизм наследования приоритетов.

Для обработки прерываний с целью минимизации затрат времени на подпрограммы обслуживания прерываний (ISR-подпрограммы) в NT была введена концепция отложенных вызовов процедур (deferred procedure calls - DPC). Хотя приоритет этих вызовов выше, чем приоритет пользовательских и системных потоков, все они находятся на одном и том же уровне. Это означает, что все DPC-вызовы выстраиваются в FIFO-очередь и что прерывание высокого уровня будет обслужено только тогда, когда свое исполнение завершат все предшествующие ему DPC-вызовы. Вследствие этого, время отклика системы становится непредсказуемым, что противоречит пятому требованию.

В NT в основе управления памятью лежит механизм виртуальной памяти. Это влечет за собой защиту памяти, преобразование адресов и подкачку (свопинг). Для приложений РВ свопинг неприемлем. Страницы памяти могут быть заблокированы в физической памяти. Однако Джеффри Рихтер (Jeffrey Richter) в своей книге [Richter95] утверждает, что, если процесс не активен, NT может разблокировать страницы процесса и переписать их из физической памяти на диск.ПРИМЕЧАНИЕ

Windows NT в "чистом виде" подходит разве что для систем мягкого РВ для применения в приложениях реального времени.

# 2.5.4 Расширения реального времени для Windows NT. Расширение функциональности

Расширения реального времени добавляют к Windows NT специфическую для реального времени функциональность:

Появляются процессы реального времени, управляемые собственным планировщиком. Этот планировщик работает уже по всем правилам планировщиков реального времени и использует алгоритм вытеснения по приоритетам. Кроме того, процессы реального времени имеют преимущество перед стандартными процессами Win32, вытесняя их. Процессы реального времени имеют совсем иную, по сравнению со стандартными процессами Windows NT, степень надежности и специфическую функциональность. Процессы реального времени и стандартные процессы Win32 имеют средства взаимодействия друг с другом. Процессы реального времени имеют свой собственный программный интерфейс RTAPI, реализующий развитый набор средств, характерный для программных интерфейсов (API) операционных систем реального времени. Одно и то же приложение может использовать как стандартные функции Win32, так и специфические функции API реального времени (RTAPI), что позволяет выделять критические участки кода приложений WindowsNT и контролировать время и надежность их выполнения. Появляется возможность контроля за работоспособностью и временами реакции системы. "Зависания" стандартных приложений Windows NT или "крах" системы не приводят к "зависанию" приложений реального времени. Появляется возможность работы с быстрыми часами и таймерами высокого разрешения. Появляется возможность прямого доступа к памяти и физическим устройствам

# 2.5.4.1 Подсистема реального времени RTSS

Подсистема реального времени RTSS обеспечивает исполнение большинства функций и управление ресурсами расширений реального времени. С точки зрения реализации, RTSS выглядит как драйвер Windows NT и выполняется в режиме ядра. Это позволяет достаточно простым способом устроить взаимодействие между процессами реального времени и процессами Windows NT. RTSS обеспечивает исполнение функций RTAPI и содержит планировщик нитей реального времени со 128-ю фиксированными приоритетами. RTSS содержит также менеджер объектов, предоставляющий унифицированные механизмы использования системных ресурсов. Управление объектами RTSS: Предоставляет возможности унифицированного управления объектами RTSS (создание,закрытие,доступ). Объектами RTSS являются: таймеры, обработчики прерываний и исключительных ситуаций (startup, shutdown, blue screen), нити, процессы, семафоры, мьютексы, разделяемая память, почтовые ящики, консольный и файловый ввод-вывод, регистры.

# 2.5.4.2 HAL реального времени

HAL является программным компонентом самого низкого уровня при взаимодействии драйверов ядра с аппаратурой. В частности, именно на уровне HAL происходит первоначальная обработка прерываний от таймера. Важной особенностью реализации Real-Time HAL является то, что он полностью совместим со стандартным HAL, и, более того, время исполнения кодов Real-Time HAL и стандартного HAL совпадают в случае, если подсистема реального времени не используется.

# 2.5.4.3 Программный интерфейс реального времени RTAPI

Программный интерфейс реального времени RTAPI является расширением Win32 и содержит, прежде всего, набор функций, необходимых для управления устройствами. RTAPI реализован в двух видах - как подмножество подсистемы реального времени (RTSS) и как динамическая библиотека (DLL), которая может вызываться из Win32-приложений. RTAPI содержит следующие группы функций: Управление процессами и нитями: Предоставляет Win32-совместимый интерфейс для управления, создания, изменения приоритетов, профилирования и завершения нитей реального времени. Взаимодействие между процессами: RTAPI использует семафоры, мьютексы и разделяемую память для взаимодействия как нитей реального времени между собой, так и для взаимодействия процессов реального времени с процессами WIN32. Управление памятью: Позволяет фиксировать приложения в памяти, запрещая их выгрузку в файл подкачки. Доступ к физической памяти: Приложение пользователя получает возможность доступа к данным по физическим адресам памяти. Управление прерываниями: Содержит функции, позволяющие назначать и запрещать обработчики прерываний, разрешать и запрещать прерывания. Часы и таймеры: Содержит функции управления часами и таймерами (создание, удаление, отмена, инициализация таймеров, назначение обработчиков прерываний) Управление вводом-выводом: RTAPI предоставляет два способа управления устройствами ввода-вывода. Во-первых, приложения пользователя получают возможность непосредственного доступа к адресам портов ввода-вывода, что позволяет программировать работу устройств напрямую. Кроме того, внешнее устройство может управляться специальными (легко разрабатываемыми) драйверами, для работы с которыми RTAPI предоставляет специальный интерфейс.

# Литература

1. Часть 1 «Программируемые контроллеры». 40 стр.
2. Часть 2 «Операционные системы реального времени. OS-9. Промышленные сети». 156 стр.
3. Часть 3 «Инструментальная система программирования логических контроллеров ISaGRAF». 228 стр.
4. QNX4 User's Guide
5. Журнал "СТА" (Современные Технологии Автоматизации), 96, 97 г.
6. Журнал "Открытые системы", 97 г. ( <http://www.osp.ru/os/index> )